man pages section 1M: System Administration Commands

Part No: E29031
October 2012

ORACLE

121203@25097

Contents

Preface

Both novice users and those familar with the SunOS operating system can use online man pages to obtain information about the system and its features. A man page is intended to answer concisely the question "What does it do?" The man pages in general comprise a reference manual. They are not intended to be a tutorial.

Overview

The following contains a brief description of each man page section and the information it references:

- Section 1 describes, in alphabetical order, commands available with the operating system.
- Section 1M describes, in alphabetical order, commands that are used chiefly for system maintenance and administration purposes.
- Section 2 describes all of the system calls. Most of these calls have one or more error returns. An error condition is indicated by an otherwise impossible returned value.
- Section 3 describes functions found in various libraries, other than those functions that directly invoke UNIX system primitives, which are described in Section 2.
- Section 4 outlines the formats of various files. The C structure declarations for the file formats are given where applicable.
- Section 5 contains miscellaneous documentation such as character-set tables.
- Section 7 describes various special files that refer to specific hardware peripherals and device drivers. STREAMS software drivers, modules and the STREAMS-generic set of system calls are also described.
- Section 9E describes the DDI (Device Driver Interface)/DKI (Driver/Kernel Interface), DDI-only, and DKI-only entry-point routines a developer can include in a device driver.
- Section 9F describes the kernel functions available for use by device drivers.
- Section 9S describes the data structures used by drivers to share information between the driver and the kernel.

Below is a generic format for man pages. The man pages of each manual section generally follow this order, but include only needed headings. For example, if there are no bugs to report,

there is no BUGS section. See the `intro` pages for more information and detail about each section, and `man(1)` for more information about man pages in general.

NAME — This section gives the names of the commands or functions documented, followed by a brief description of what they do.

SYNOPSIS — This section shows the syntax of commands or functions. When a command or file does not exist in the standard path, its full path name is shown. Options and arguments are alphabetized, with single letter arguments first, and options with arguments next, unless a different argument order is required.

The following special characters are used in this section:

- [ ] — Brackets. The option or argument enclosed in these brackets is optional. If the brackets are omitted, the argument must be specified.
- . . . — Ellipses. Several values can be provided for the previous argument, or the previous argument can be specified multiple times, for example, "`filename...`".
- | — Separator. Only one of the arguments separated by this character can be specified at a time.
- { } — Braces. The options and/or arguments enclosed within braces are interdependent, such that everything enclosed must be treated as a unit.

PROTOCOL — This section occurs only in subsection 3R to indicate the protocol description file.

DESCRIPTION — This section defines the functionality and behavior of the service. Thus it describes concisely what the command does. It does not discuss OPTIONS or cite EXAMPLES. Interactive commands, subcommands, requests, macros, and functions are described under USAGE.

IOCTL — This section appears on pages in Section 7 only. Only the device class that supplies appropriate parameters to the `ioctl(2)` system call is called `ioctl` and generates its own heading. `ioctl` calls for a specific device are listed alphabetically (on the man page for that specific device).

	`ioctl` calls are used for a particular class of devices all of which have an `io` ending, such as `mtio(7I)`.
OPTIONS	This section lists the command options with a concise summary of what each option does. The options are listed literally and in the order they appear in the SYNOPSIS section. Possible arguments to options are discussed under the option, and where appropriate, default values are supplied.
OPERANDS	This section lists the command operands and describes how they affect the actions of the command.
OUTPUT	This section describes the output – standard output, standard error, or output files – generated by the command.
RETURN VALUES	If the man page documents functions that return values, this section lists these values and describes the conditions under which they are returned. If a function can return only constant values, such as 0 or –1, these values are listed in tagged paragraphs. Otherwise, a single paragraph describes the return values of each function. Functions declared void do not return values, so they are not discussed in RETURN VALUES.
ERRORS	On failure, most functions place an error code in the global variable `errno` indicating why they failed. This section lists alphabetically all error codes a function can generate and describes the conditions that cause each error. When more than one condition can cause the same error, each condition is described in a separate paragraph under the error code.
USAGE	This section lists special rules, features, and commands that require in-depth explanations. The subsections listed here are used to explain built-in functionality: Commands Modifiers Variables Expressions Input Grammar
EXAMPLES	This section provides examples of usage or of how to use a command or function. Wherever possible a complete

	example including command-line entry and machine response is shown. Whenever an example is given, the prompt is shown as `example%`, or if the user must be superuser, `example#`. Examples are followed by explanations, variable substitution rules, or returned values. Most examples illustrate concepts from the SYNOPSIS, DESCRIPTION, OPTIONS, and USAGE sections.
ENVIRONMENT VARIABLES	This section lists any environment variables that the command or function affects, followed by a brief description of the effect.
EXIT STATUS	This section lists the values the command returns to the calling program or shell and the conditions that cause these values to be returned. Usually, zero is returned for successful completion, and values other than zero for various error conditions.
FILES	This section lists all file names referred to by the man page, files of interest, and files created or required by commands. Each is followed by a descriptive summary or explanation.
ATTRIBUTES	This section lists characteristics of commands, utilities, and device drivers by defining the attribute type and its corresponding value. See `attributes(5)` for more information.
SEE ALSO	This section lists references to other man pages, in-house documentation, and outside publications.
DIAGNOSTICS	This section lists diagnostic messages with a brief explanation of the condition causing the error.
WARNINGS	This section lists warnings about special conditions which could seriously affect your working conditions. This is not a list of diagnostics.
NOTES	This section lists additional information that does not belong anywhere else on the page. It takes the form of an aside to the user, covering points of special interest. Critical information is never covered here.
BUGS	This section describes known bugs and, wherever possible, suggests workarounds.

REFERENCE

Introduction

Name Intro – introduction to maintenance commands and application programs

Description This section describes, in alphabetical order, commands that are used chiefly for system maintenance and administration purposes.

Because of command restructuring for the Virtual File System architecture, there are several instances of multiple manual pages that begin with the same name. For example, the `mount`, pages – `mount(1M)`, `mount_hsfs(1M)`, `mount_nfs(1M)`, `mount_tmpfs(1M)`, and `mount_ufs(1M)`. In each such case the first of the multiple pages describes the syntax and options of the generic command, that is, those options applicable to all FSTypes (file system types). The succeeding pages describe the functionality of the FSType-specific modules of the command. These pages list the command followed by an underscore (_) and the FSType to which they pertain. Note that the administrator should not attempt to call these modules directly. The generic command provides a common interface to all of them. Thus the FSType-specific manual pages should not be viewed as describing distinct commands, but rather as detailing those aspects of a command that are specific to a particular FSType.

Command Syntax Unless otherwise noted, commands described in this section accept options and other arguments according to the following syntax:

name [*option*(*s*)] [*cmdarg*(*s*)]

where:

name	The name of an executable file.
option	– *noargletter*(*s*) or, – *argletter*< >*optarg* where < > is optional white space.
noargletter	A single letter representing an option without an argument.
argletter	A single letter representing an option requiring an argument.
optarg	Argument (character string) satisfying preceding *argletter*.
cmdarg	Pathname (or other command argument) *not* beginning with – or, – by itself indicating the standard input.

Attributes See `attributes(5)` for a discussion of the attributes listed in this section.

Acknowledgments Oracle America, Inc. gratefully acknowledges The Open Group for permission to reproduce portions of its copyrighted documentation. Original documentation from The Open Group can be obtained online at `http://www.opengroup.org/bookstore/`.

The Institute of Electrical and Electronics Engineers and The Open Group, have given us permission to reprint portions of their documentation.

In the following statement, the phrase ‘‘this text" refers to portions of the system documentation.

Portions of this text are reprinted and reproduced in electronic form in the SunOS Reference Manual, from IEEE Std 1003.1, 2004 Edition, Standard for Information Technology -- Portable Operating System Interface (POSIX), The Open Group Base Specifications Issue 6, Copyright (C) 2001-2004 by the Institute of Electrical and Electronics Engineers, Inc and The Open Group. In the event of any discrepancy between these versions and the original IEEE and The Open Group Standard, the original IEEE and The Open Group Standard is the referee document. The original Standard can be obtained online at `http://www.opengroup.org/unix/online.html`.

This notice shall appear on any product containing this material.

See Also `getopt(1)`, `getopt(3C)`, `attributes(5)`

Diagnostics Upon termination, each command returns 0 for normal termination and non-zero to indicate troubles such as erroneous parameters, bad or inaccessible data, or other inability to cope with the task at hand. It is called variously ‘‘exit code,'' ‘‘exit status,'' or ‘‘return code,'' and is described only where special conventions are involved.

Notes Unfortunately, not all commands adhere to the standard syntax.

REFERENCE

System Administration Commands - Part 1

Name 6to4relay – administer configuration for 6to4 relay router communication

Synopsis

```
/usr/sbin/6to4relay

/usr/sbin/6to4relay [-e] [-a addr]

/usr/sbin/6to4relay [-d]

/usr/sbin/6to4relay [-h]
```

Description The `6to4relay` command is used to configure 6to4 relay router communication. Relay router communication support is enabled by setting the value of a variable that stores an IPv4 address within the `tun` module. This variable is global to all tunnels and defines the policy for communication with relay routers. By default, the address is set to `INADDR_ANY` (`0.0.0.0`), and the kernel interprets the value to indicate that support for relay router communication is disabled. Otherwise, support is enabled, and the specified address is used as the IPv4 destination address when packets destined for native IPv6 (non-6to4) hosts are sent through the 6to4 tunnel interface. The `6to4relay` command uses a project private ioctl to set the variable.

`6to4relay` used without any options outputs the current, in-kernel, configuration status. Use the `-a` option to send packets to a specific relay router's `unicast` address instead of the default `anycast` address. The address specified with the `-a` option does not specify the policy for receiving traffic from relay routers. The source relay router on a received packet is non-deterministic, since a different relay router may be chosen for each sending native IPv6 end-point.

Configuration changes made by using the `6to4relay` are not persistent across reboot. The changes will persist in the kernel only until you take the tunnel down

Options The `6to4relay` command supports the following options:

`-a` *addr* Use the specified address, *addr*.

`-e` Enable support for relay router. Use `-a` *addr* if it is specified. Otherwise, use the default `anycast` address, 192.88.99.1.

`-d` Disable support for the relay router.

`-h` Help

Operands The following operands are supported:

addr A specific relay router's unicast address. *addr* must be specified as a dotted decimal representation of an IPv4 address. Otherwise, an error will occur, and the command will fail.

Examples **EXAMPLE 1** Printing the In-Kernel Configuration Status

Use `/usr/sbin/6to4relay` without any options to print the in-kernel configuration status.

```
example# /usr/sbin/6to4relay
```

EXAMPLE 1 Printing the In-Kernel Configuration Status *(Continued)*

If 6to4 relay router communication is disabled, the administrator will see the following message:

```
6to4relay: 6to4 Relay Router communication support is disabled.
```

If 6to4 router communication is enabled, the user will see this message:

```
6to4relay: 6to4 Relay Router communication support is enabled.
IPv4 destination address of Relay Router = 192.88.99.1
```

Exit Status The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

Files `/usr/sbin/6to4relay` The default installation root

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os
Interface Stability	Committed

See Also `ifconfig(1M)`, `attributes(5)`

Huitema, C. *RFC 3068, An Anycast Prefix for 6to4 Relay Routers*. Network Working Group. June, 2001.

Carpenter, B. and Moore, K. *RFC 3056, Connection of IPv6 Domains via IPv4 Clouds*. Network Working Group. February, 2001.

Diagnostics The `6to4relay` reports the following messages:

```
6to4relay: input (0.0.0.0) is not a valid IPv4 unicast address
```

Example: The following example provides an incorrect unicast address.

```
example# 6to4relay -e -a 0.0.0.0
```

Description: The address specified with the `-a` option must be a valid unicast address.

```
6to4relay: option requires an argument —a
usage:
    6to4relay
    6to4relay -e [-a <addr>]
```

```
    6to4relay -d
    6to4relay -h
```

Example: The following example does not include an argument for the `-a` option.

```
example# 6to4relay -e -a
```

Description: The `-a` option requires an argument.

```
usage:
    6to4relay
    6to4relay -e [-a <addr>]
    6to4relay -d
    6to4relay -h
```

Example: The following example specifies options that are not permitted.

```
example# 6to4relay -e -d
```

Description: The options specified are not permitted. A usage message is output to the screen.

```
usage:
    6to4relay
    6to4relay -e [-a <addr>]
    6to4relay -d
    6to4relay -h
```

Example: The following example specifies the `-a` option without specifying the `-e` option.

```
example# 6to4relay -a 1.2.3.4
```

Description: The `-e` option is required in conjunction with the `-a` option. A usage message is output to the screen.

```
6to4relay: ioctl (I_STR) : Invalid argument
```

Example: The following example specifies an invalid address.

```
example# 6to4relay -e -a 239.255.255.255
```

Description: The address specified with the `-a` option must not be a class d *addr*.

Name acct, acctdisk, acctdusg, accton, acctwtmp, closewtmp, utmp2wtmp – overview of accounting and miscellaneous accounting commands

Synopsis

```
/usr/lib/acct/acctdisk

/usr/lib/acct/acctdusg [-u filename] [-p filename]

/usr/lib/acct/accton [filename]

/usr/lib/acct/acctwtmp reason filename

/usr/lib/acct/closewtmp

/usr/lib/acct/utmp2wtmp
```

Description Accounting software is structured as a set of tools (consisting of both C programs and shell procedures) that can be used to build accounting systems. acctsh(1M) describes the set of shell procedures built on top of the C programs.

Connect time accounting is handled by various programs that write records into /var/adm/wtmpx, as described in utmpx(4). The programs described in acctcon(1M) convert this file into session and charging records, which are then summarized by acctmerg(1M).

Process accounting is performed by the system kernel. Upon termination of a process, one record per process is written to a file (normally /var/adm/pacct). The programs in acctprc(1M) summarize this data for charging purposes; acctcms(1M) is used to summarize command usage. Current process data may be examined using acctcom(1).

Process accounting records and connect time accounting records (or any accounting records in the tacct format described in acct.h(3HEAD)) can be merged and summarized into total accounting records by acctmerg (see tacct format in acct.h(3HEAD)). prtacct (see acctsh(1M)) is used to format any or all accounting records.

acctdisk reads lines that contain user ID, login name, and number of disk blocks and converts them to total accounting records that can be merged with other accounting records. acctdisk returns an error if the input file is corrupt or improperly formatted.

acctdusg reads its standard input (usually from find / -print) and computes disk resource consumption (including indirect blocks) by login.

accton without arguments turns process accounting off. If *filename* is given, it must be the name of an existing file, to which the kernel appends process accounting records (see acct(2) and acct.h(3HEAD)).

acctwtmp writes a utmpx(4) record to *filename*. The record contains the current time and a string of characters that describe the *reason*. A record type of ACCOUNTING is assigned (see utmpx(4)) *reason* must be a string of 11 or fewer characters, numbers, $, or spaces. For example, the following are suggestions for use in reboot and shutdown procedures, respectively:

```
acctwtmp "acctg on" /var/adm/wtmpx
acctwtmp "acctg off" /var/adm/wtmpx
```

For each user currently logged on, `closewtmp` puts a false `DEAD_PROCESS` record in the `/var/adm/wtmpx` file. `runacct` (see `runacct(1M)`) uses this false `DEAD_PROCESS` record so that the connect accounting procedures can track the time used by users logged on before `runacct` was invoked.

For each user currently logged on, `runacct` uses `utmp2wtmp` to create an entry in the file `/var/adm/wtmpx`, created by `runacct`. Entries in `/var/adm/wtmpx` enable subsequent invocations of `runacct` to account for connect times of users currently logged in.

Options The following options are supported:

`-u` *filename* Places in *filename* records consisting of those filenames for which `acctdusg` charges no one (a potential source for finding users trying to avoid disk charges).

`-p` *filename* Specifies a password file, *filename*. This option is not needed if the password file is `/etc/passwd`.

Environment Variables If any of the `LC_*` variables (`LC_TYPE`, `LC_MESSAGES`, `LC_TIME`, `LC_COLLATE`, `LC_NUMERIC`, and `LC_MONETARY`) (see `environ(5)`) are not set in the environment, the operational behavior of `acct` for each corresponding locale category is determined by the value of the `LANG` environment variable. If `LC_ALL` is set, its contents are used to override both the `LANG` and the other `LC_*` variables. If none of the above variables are set in the environment, the "C" (U.S. style) locale determines how `acct` behaves.

`LC_CTYPE` Determines how `acct` handles characters. When `LC_CTYPE` is set to a valid value, `acct` can display and handle text and filenames containing valid characters for that locale. `acct` can display and handle Extended Unix Code (EUC) characters where any character can be 1, 2, or 3 bytes wide. `acct` can also handle EUC characters of 1, 2, or more column widths. In the "C" locale, only characters from ISO 8859-1 are valid.

`LC_TIME` Determines how `acct` handles date and time formats. In the "C" locale, date and time handling follows the U.S. rules.

Files

`/etc/passwd` Used for login name to user ID conversions.

`/usr/lib/acct` Holds all accounting commands listed in sub-class 1M of this manual.

`/var/adm/pacct` Current process accounting file.

`/var/adm/wtmpx` History of user access and administration information..

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/accounting/legacy-accounting

See Also acctcom(1), acctcms(1M), acctcon(1M), acctmerg(1M), acctprc(1M), acctsh(1M), fwtmp(1M), runacct(1M), acct(2), acct.h(3HEAD), passwd(4), utmpx(4), attributes(5), environ(5)

Oracle Solaris Administration: Common Tasks

Name acctadm – configure extended accounting facility

Synopsis

```
/usr/sbin/acctadm [-DEsrux] [-d resource_list]
     [-e resource_list] [-f filename]
     [task | process | flow | net]
```

Description `acctadm` configures various attributes of the extended accounting facility. Without arguments, `acctadm` displays the current status of the extended accounting facility.

Options The following options are supported:

`-d` *resource_list*
Disable reporting of resource usage for resource. Specify *resource_list* as a comma-separated list of resources or resource groups.

This option requires an operand. See `OPERANDS`.

`-D`
Disable accounting of the given operand type without closing the accounting file. This option can be used to temporarily stop writing accounting records to the accounting file without closing it. To close the file use the `-x` option. See `-x`.

`-e` *resource_list*
Enable reporting of resource usage for resource. Specify *resource_list* as a comma-separated list of resources or resource groups.

This option requires an operand. See `OPERANDS`.

`-E`
Enable accounting of the given operand type without sending the accounting output to a file. This option requires an operand. See `OPERANDS`.

`-f` *filename*
Send the accounting output for the given operand type to *filename*. If *filename* exists, its contents must be of the given accounting type.

This option requires an operand. See `OPERANDS`.

`-r`
Display available resource groups.

When this option is used with an operand, it displays resource groups available for a given accounting type. When no operand is specified, this option displays resource groups for all available accounting types. See OPERANDS.

`-s`
Start method for the `smf(5)` instance. This option is used to restore the extended accounting configuration at boot.

`-x`
Deactivate accounting of the given operand type. This option also closes the accounting file for the given accounting type if it is currently open.

This option requires an operand. See `OPERANDS`.

Operands The `-d`, `-D`, `-e`, `-E`, `-f`, and `-x` options require an operand.

The following operands are supported:

`process`
Run `acctadm` on the process accounting components of the extended accounting facility.

`task`
Run `acctadm` on the task accounting components of the extended accounting facility.

`flow`
Run `acctadm` on the IPQoS accounting components of the extended accounting facility.

`net`
Run `acctadm` on links and flows administered by `dladm(1M)` and `flowadm(1M)`, respectively. Basic network accounting relates only to links, while extended network accounting includes both link and flow accounting.

The optional final parameter to `acctadm` represents whether the command should act on the process, system task, IPQoS, or network accounting components of the extended accounting facility.

Examples **EXAMPLE 1** Displaying the Current Status

The following command displays the current status. In this example, system task accounting is active and tracking only CPU resources. Process and flow accounting are not active.

```
$ acctadm
            Task accounting: active
       Task accounting file: /var/adm/exacct/task
     Tracked task resources: extended
   Untracked task resources: host
         Process accounting: inactive
    Process accounting file: none
  Tracked process resources: none
Untracked process resources: extended,host
            Flow accounting: inactive
       Flow accounting file: none
     Tracked flow resources: none
   Untracked flow resources: extended
             Net accounting: inactive
        Net accounting file: none
      Tracked Net resources: none
    Untracked Net resources: extended
```

EXAMPLE 2 Activating Basic Process Accounting

The following command activates basic process accounting:

```
$ acctadm -e basic -f /var/adm/exacct/proc process
```

EXAMPLE 3 Displaying Available Resource Groups

The following command displays available resource groups:

```
$ acctadm -r
  process:
  extended pid,uid,gid,cpu,time,command,tty,projid, \
  taskid,ancpid,wait-status,zone,flag,memory,mstate
  basic    pid,uid,gid,cpu,time,command,tty,flag
  task:
  extended taskid,projid,cpu,time,host,mstate,anctaskid,zone
  basic    taskid,projid,cpu,time
  flow:
  extended saddr,daddr,sport,dport,proto,dsfield,nbytes,npkts, \
  action,ctime,lseen,projid,uid
  basic    saddr,daddr,sport,dport,proto,nbytes,npkts,action
  net:
  extended name,devname,edest,vlan_tpid,vlan_tci,sap,cpuid, \
  priority,bwlimit,curtime,ibytes,obytes,ipkts,opks,ierrpkts \
  oerrpkts,saddr,daddr,sport,dport,protocol,dsfield
  basic    name,devname,edest,vlan_tpid,vlan_tci,sap,cpuid, \
  priority,bwlimit,curtime,ibytes,obytes,ipkts,opks,ierrpkts \
  oerrpkts
```

In the output above, the lines beginning with `extended` are shown with a backslash character. In actual `acctadm` output, these lines are displayed as unbroken, long lines.

EXAMPLE 4 Displaying Resource Groups for Task Accounting

The following command displays resource groups for task accounting:

```
$ acctadm -r task
  extended taskid,projid,cpu,time,host,mstate,anctaskid,zone
  basic    taskid,projid,cpu,time
```

Exit Status

The following exit values are returned:

0

Successful completion.

The modifications to the current configuration were valid and made successfully.

1

An error occurred.

A fatal error occured either in obtaining or modifying the accounting configuration.

2
Invalid command line options were specified.

95
A fatal, non-configuration error occurred during the start of the smf(5) service instance.

96
A fatal configuration error occurred during the start of the smf(5) service instance.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os
Interface Stability	Committed

See Also dladm(1M), flowadm(1M), acct(2), attributes(5), smf(5), ipqos(7ipp)

Notes Both extended accounting and regular accounting can be active.

Available resources can vary from system to system, and from platform to platform.

Extended accounting configuration is stored in the service management facility (smf(5)) repository. The configuration is restored at boot by a transient service instance, one per accounting type:

```
svc:/system/extended-accounting:flow        Flow accounting
svc:/system/extended-accounting:process     Process accounting
svc:/system/extended-accounting:task        Task accounting
svc:/system/extended-accounting:net         Network accounting
```

The instances are enabled or disabled by `acctadm` as needed. Configuration changes are made using `acctadm`; service properties should not be modified directly using svccfg(1M).

Users can manage extended accounting (start accounting, stop accounting, change accounting configuration parameters) if they have the appropriate RBAC Rights profile for the accounting type to be managed:

- Extended Accounting Flow Management
- Extended Accounting Process Management
- Extended Accounting Task Management
- Extended Accounting Network Management

The preceding profiles are for, respectively, flow accounting, process accounting, task accounting, and network accounting.

Name acctcms – command summary from process accounting records

Synopsis

```
/usr/lib/acct/acctcms [-a [-o] [-p]] [-c] [-j] [-n] [-s]
     [-t] filename...
```

Description `acctcms` reads one or more *filename*s, normally in the form described in `acct.h(3HEAD)`. It adds all records for processes that executed identically named commands, sorts them, and writes them to the standard output, normally using an internal summary format.

Options

`-a` Print output in ASCII rather than in the internal summary format. The output includes command name, number of times executed, total kcore-minutes, total CPU minutes, total real minutes, mean size (in K), mean CPU minutes per invocation, "hog factor," characters transferred, and blocks read and written, as in `acctcom(1)`. Output is normally sorted by total kcore-minutes.

Use the following options only with the `-a` option:

`-o` Output a (non-prime) offshift-time-only command summary.

`-p` Output a prime-time-only command summary.

When `-o` and `-p` are used together, a combination prime-time and non-prime-time report is produced. All the output summaries are total usage except number of times executed, CPU minutes, and real minutes, which are split into prime and non-prime.

`-c` Sort by total CPU time, rather than total kcore-minutes.

`-j` Combine all commands invoked only once under "***other".

`-n` Sort by number of command invocations.

`-s` Any file names encountered hereafter are already in internal summary format.

`-t` Process all records as total accounting records. The default internal summary format splits each field into prime and non-prime-time parts. This option combines the prime and non-prime time parts into a single field that is the total of both, and provides upward compatibility with old style `acctcms` internal summary format records.

Examples **EXAMPLE 1** Using the `acctcms` command.

A typical sequence for performing daily command accounting and for maintaining a running total is:

```
example% acctcms filename ... > today
example% cp total previoustotal
example% acctcms -s today previoustotal > total
example% acctcms -a -s today
```

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/accounting/legacy-accounting

See Also acctcom(1), acct(1M), acctcon(1M), acctmerg(1M), acctprc(1M), acctsh(1M), fwtmp(1M), runacct(1M), acct(2), acct.h(3HEAD), utmpx(4), attributes(5)

Notes Unpredictable output results if -t is used on new style internal summary format files, or if it is not used with old style internal summary format files.

Name acctcon, acctcon1, acctcon2 – connect-time accounting

Synopsis

```
/usr/lib/acct/acctcon [-l lineuse] [-o reboot]

/usr/lib/acct/acctcon1 [-p] [-t] [-l lineuse] [-o reboot]

/usr/lib/acct/acctcon2
```

Description `acctcon` converts a sequence of login/logoff records to total accounting records (see the `tacct` format in `acct.h`(3HEAD)). The login/logoff records are read from standard input. The file `/var/adm/wtmpx` is usually the source of the login/logoff records; however, because it might contain corrupted records or system date changes, it should first be fixed using `wtmpfix`. The fixed version of file `/var/adm/wtmpx` can then be redirected to `acctcon`. The `tacct` records are written to standard output.

`acctcon` is a combination of the programs `acctcon1` and `acctcon2`. `acctcon1` converts login/logoff records, taken from the fixed `/var/adm/wtmpx` file, to ASCII output. `acctcon2` reads the ASCII records produced by `acctcon1` and converts them to `tacct` records. `acctcon1` can be used with the `-l` and `-o` options, described below, as well as with the `-p` and `-t` options.

Options

`-p` Print input only, showing line name, login name, and time (in both numeric and date/time formats).

`-t` `acctcon1` maintains a list of lines on which users are logged in. When it reaches the end of its input, it emits a session record for each line that still appears to be active. It normally assumes that its input is a current file, so that it uses the current time as the ending time for each session still in progress. The `-t` flag causes it to use, instead, the last time found in its input, thus assuring reasonable and repeatable numbers for non-current files.

`-l` *lineuse* *lineuse* is created to contain a summary of line usage showing line name, number of minutes used, percentage of total elapsed time used, number of sessions charged, number of logins, and number of logoffs. This file helps track line usage, identify bad lines, and find software and hardware oddities. Hangup, termination of `login`(1) and termination of the login shell each generate logoff records, so that the number of logoffs is often three to four times the number of sessions. See `init`(1M) and `utmpx`(4).

`-o` *reboot* `reboot` is filled with an overall record for the accounting period, giving starting time, ending time, number of reboots, and number of date changes.

Examples **EXAMPLE 1** Using the `acctcon` command.

The `acctcon` command is typically used as follows:

```
example% acctcon -l lineuse -o reboots < tmpwtmp > ctacct
```

The `acctcon1` and `acctcon2` commands are typically used as follows:

EXAMPLE 1 Using the `acctcon` command. *(Continued)*

```
example% acctcon1 -l lineuse -o reboots < tmpwtmp | sort +1n +2 > ctmp
example% acctcon2 < ctmp > ctacct
```

Files `/var/adm/wtmpx` History of user access and administration information

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/accounting/legacy-accounting

See Also `acctcom(1)`, `login(1)`, `acct(1M)`, `acctcms(1M)`, `acctmerg(1M)`, `acctprc(1M)`, `acctsh(1M)`, `fwtmp(1M)`, `init(1M)`, `runacct(1M)`, `acct(2)`, `acct.h(3HEAD)`, `utmpx(4)`, `attributes(5)`

Oracle Solaris Administration: Common Tasks

Notes The line usage report is confused by date changes. Use `wtmpfix` (see `fwtmp(1M)`), with the `/var/adm/wtmpx` file as an argument, to correct this situation.

During a single invocation of any given command, the `acctcon`, `acctcon1`, and `acctcon2` commands can process a maximum of:

- 6000 distinct session
- 1000 distinct terminal lines
- 2000 distinct login names

If at some point the actual number of any one of these items exceeds the maximum, the command will not succeed.

Name acctmerg – merge or add total accounting files

Synopsis

```
/usr/lib/acct/acctmerg [-a] [-i] [-p] [-t] [-u] [-v]
     [filename] ...
```

Description `acctmerg` reads its standard input and up to nine additional files, all in the `tacct` format (see `acct.h`(3HEAD)) or an ASCII version thereof. It merges these inputs by adding records whose keys (normally user ID and name) are identical, and expects the inputs to be sorted on those keys.

Options

`-a` Produce output in ASCII version of `tacct`.

`-i` Produce input in ASCII version of `tacct`.

`-p` Print input with no processing.

`-t` Produce a single record that totals all input.

`-u` Summarize by user ID, rather than by user ID and name.

`-v` Produce output in verbose ASCII format, with more precise notation for floating-point numbers.

Examples **EXAMPLE 1** Using the `acctmerg` command.

The following sequence is useful for making "repairs" to any file kept in this format:

```
example% acctmerg   -v  <filename1 >filename2
```

Edit *filename2* as you want:

```
example% acctmerg   -i  <filename2 >filename1
```

Attributes See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/accounting/legacy-accounting

See Also `acctcom`(1), `acct`(1M), `acctcms`(1M), `acctcon`(1M), `acctprc`(1M), `acctsh`(1M), `fwtmp`(1M), `runacct`(1M), `acct`(2), `acct.h`(3HEAD), `utmpx`(4), `attributes`(5)

Oracle Solaris Administration: Common Tasks

Name acctprc, acctprc1, acctprc2 – process accounting

Synopsis

```
/usr/lib/acct/acctprc

/usr/lib/acct/acctprc1 [ctmp]

/usr/lib/acct/acctprc2
```

Description `acctprc` reads the standard input and converts it to total accounting records (see the `tacct` record in `acct.h`(3HEAD)). `acctprc` divides CPU time into prime time and non-prime time and determines mean memory size (in memory segment units). `acctprc` then summarizes the `tacct` records, according to user IDs, and adds login names corresponding to the user IDs. The summarized records are then written to the standard output. `acctprc1` reads input in the form described by `acct.h`(3HEAD), adds login names corresponding to user IDs, then writes for each process an ASCII line giving user ID, login name, prime CPU time (tics), non-prime CPU time (tics), and mean memory size (in memory segment units). If *ctmp* is given, it should contain a list of login sessions sorted by user ID and login name. If this file is not supplied, it obtains login names from the password file, just as `acctprc` does. The information in *ctmp* helps it distinguish between different login names that share the same user ID.

From the standard input, `acctprc2` reads records in the form written by `acctprc1`, summarizes them according to user ID and name, then writes the sorted summaries to the standard output as total accounting records.

Examples **EXAMPLE 1** Examples of `acctprc`.

The `acctprc` command is typically used as shown below:

```
example% acctprc  < /var/adm/pacct  > ptacct
```

The `acctprc1` and `acctprc2s` commands are typically used as shown below:

```
example% acctprc1  ctmp  </var/adm/pacct
example% acctprc2 > ptacct
```

Files `/etc/passwd` system password file

Attributes See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/accounting/legacy-accounting

See Also `acctcom`(1), `acct`(1M), `acctcms`(1M), `acctcon`(1M), `acctmerg`(1M), `acctsh`(1M), `cron`(1M), `fwtmp`(1M), `runacct`(1M), `acct`(2), `acct.h`(3HEAD), `utmpx`(4), `attributes`(5)

Notes Although it is possible for `acctprc1` to distinguish among login names that share user IDs for commands run from a command line, it is difficult for `acctprc1` to make this distinction for commands invoked in other ways. A command run from `cron`(1M) is an example of where

`acctprc1` might have difficulty. A more precise conversion can be done using the `acctwtmp` program in `acct(1M)`. `acctprc` does not distinguish between users with identical user IDs.

A memory segment of the mean memory size is a unit of measure for the number of bytes in a logical memory segment on a particular processor.

During a single invocation of any given command, the `acctprc`, `acctprc1`, and `acctprc2` commands can process a maximum of

- 6000 distinct sessions
- 1000 distinct terminal lines
- 2000 distinct login names

If at some point the actual number of any one of these items exceeds the maximum, the command will not succeed.

Name acctsh, chargefee, ckpacct, dodisk, lastlogin, monacct, nulladm, prctmp, prdaily, prtacct, shutacct, startup, turnacct – shell procedures for accounting

Synopsis

```
/usr/lib/acct/chargefee login-name number

/usr/lib/acct/ckpacct [blocks]

/usr/lib/acct/dodisk [-o] [filename]...

/usr/lib/acct/lastlogin

/usr/lib/acct/monacct number

/usr/lib/acct/nulladm filename...

/usr/lib/acct/prctmp filename

/usr/lib/acct/prdaily [-c] [-l] [mmdd]

/usr/lib/acct/prtacct filename ['' heading '']

/usr/lib/acct/shutacct ['' reason '']

/usr/lib/acct/startup

/usr/lib/acct/turnacct on | off | switch
```

Description

chargefee Command — `chargefee` can be invoked to charge a *number* of units to *login-name*. A record is written to `/var/adm/fee`, to be merged with other accounting records by `runacct(1M)`.

ckpacct Command — `ckpacct` should be initiated using `cron(1M)` to periodically check the size of `/var/adm/pacct`. If the size exceeds *blocks*, `500` by default, `turnacct` will be invoked with argument `switch`. To avoid a conflict with `turnacct switch` execution in `runacct`, do not run `ckpacct` and `runacct` simultaneously. If the number of free disk blocks in the `/var` file system falls below `500`, `ckpacct` will automatically turn off the collection of process accounting records via the `off` argument to `turnacct`. When at least `500` blocks are restored, the accounting will be activated again on the next invocation of `ckpacct`. This feature is sensitive to the frequency at which `ckpacct` is executed, usually by the `cron(1M)` command.

dodisk Command — `dodisk` should be invoked by `cron`(1M) to perform the disk accounting functions.

lastlogin Command — `lastlogin` is invoked by `runacct(1M)` to update `/var/adm/acct/sum/loginlog`, which shows the last date on which each person logged in.

monacct Command — `monacct` should be invoked once each month or each accounting period. *number* indicates which month or period it is. If *number* is not given, it defaults to the current month (01–12). This default is useful if `monacct` is to executed using `cron(1M)` on the first day of each month. `monacct` creates summary files in `/var/adm/acct/fiscal` and restarts the summary files in `/var/adm/acct/sum`.

nulladm Command — `nulladm` creates *filename* with mode 664 and ensures that owner and group are `adm`. It is called by various accounting shell procedures.

prctmp Command — `prctmp` can be used to print the session record file (normally `/var/adm/acct/nite/ctmp` created by `acctcon1` (see `acctcon(1M)`).

prdaily Command — `prdaily` is invoked by `runacct(1M)` to format a report of the previous day's accounting data. The report resides in `/var/adm/acct/sum/rprt/mmdd` where *mmdd* is the month and day of the report. The current daily accounting reports may be printed by typing `prdaily`. Previous days' accounting reports can be printed by using the *mmdd* option and specifying the exact report date desired.

prtacct Command — `prtacct` can be used to format and print any total accounting (`tacct`)file.

shutacct Command — `shutacct` is invoked during a system shutdown to turn process accounting off and append a *reason* record to `/var/adm/wtmpx`.

startup Command — `startup` can be invoked when the system is brought to a multi-user state to turn process accounting on.

turnacct Command — `turnacct` is an interface to `accton` (see `acct(1M)`) to turn process accounting `on` or `off`. The `switch` argument moves the current `/var/adm/pacct` to the next free name in `/var/adm/pacct.`*incr* (where *incr* is a number starting with `0` and incrementing by one for each additional `pacct` file), then turns accounting back on again. This procedure is called by `ckpacct` and thus can be taken care of by the `cron(1M)` command and used to keep `pacct` to a reasonable size. `shutacct` uses `turnacct` to stop process accounting. `startup` uses `turnacct` to start process accounting.

Options The following options are supported:

`-c` This option prints a report of exceptional resource usage by command, and may be used on current day's accounting data only.

`-l` This option prints a report of exceptional usage by login id for the specified date. Previous daily reports are cleaned up and therefore inaccessible after each invocation of `monacct`.

`-o` This option uses `acctdusg` (see `acct(1M)`) to do a slower version of disk accounting by login directory. *filename*s specifies the one or more filesystem names where disk accounting will be done. If *filename*s are used, disk accounting will be done on these filesystems only. If the `-o` option is used, *filename*s should be mount points of mounted filesystems. If the `-o` option is omitted, *filename*s should be the special file names of mountable filesystems.

Files

`/etc/logadm.conf`	Configuration file for the `logadm(1M)` command
`/usr/lib/acct`	Holds all accounting commands listed in section 1M of this manual

`/usr/lib/acct/ptecms.awk`	Contains the limits for exceptional usage by command name
`/usr/lib/acct/ptelus.awk`	Contains the limits for exceptional usage by login ID
`/var/adm/acct/fiscal`	Fiscal reports directory
`/var/adm/acct/nite`	Working directory
`/var/adm/acct/sum`	Summary directory that contains information for `monacct`
`/var/adm/acct/sum/loginlog`	File updated by last login
`/var/adm/fee`	Accumulator for fees
`/var/adm/pacct`	Current file for per-process accounting
`/var/adm/pacct`*incr*	Used if `pacct` gets large and during execution of daily accounting procedure
`/var/adm/wtmpx`	History of user access and administration information

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/accounting/legacy-accounting

See Also `acctcom(1)`, `acct(1M)`, `acctcms(1M)`, `acctcon(1M)`, `acctmerg(1M)`, `acctprc(1M)`, `cron(1M)`, `fwtmp(1M)`, `logadm(1M)`, `runacct(1M)`, `acct(2)`, `acct.h(3HEAD)`, `utmpx(4)`, `attributes(5)`

Notes See `runacct(1M)` for the main daily accounting shell script, which performs the accumulation of connect, process, fee, and disk accounting on a daily basis. It also creates summaries of command usage.

Name acpihpd – ACPI hot plug daemon

Synopsis

```
/usr/platform/i86pc/lib/acpihpd
```

Description The ACPI hot plug daemon, `acpihpd`, is a daemon process that runs on x86 platforms. The daemon is started by the service management facility. It communicates with the `syseventd(1M)` and `cfgadm(1M)` subsystems to handle ACPI events from hotpluggable devices.

The service FMRI for `acpihpd` is:

```
svc:/platform/i86pc/acpihpd:default
```

Note that `acpihpd` is a private interface.

Options The `acpihpd` daemon does not support any options.

Errors `acpihpd` uses `syslog(3C)` to report status and error messages. All of the messages are logged with the `LOG_DAEMON` facility.

Error messages are logged with the `LOG_ERR` and `LOG_NOTICE` priorities, and informational messages are logged with the `LOG_DEBUG` priority. The default entries in the `/etc/syslog.conf` file log all of the error messages to the `/var/adm/messages` log.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/kernel/dynamic-reconfiguration/i86pc
Interface Stability	Private

See Also `svcs(1)`, `cfgadm(1M)`, `inetadm(1M)`, `svcadm(1M)`, `syseventd(1M)`, `syslog(3C)`, `syslog.conf(4)`, `attributes(5)`, `smf(5)`

Notes The `acpihpd` service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/platform/i86pc/acpihpd:default
```

Name adbgen – generate adb script

Synopsis `/usr/lib/adb/adbgen [-m` *model*`]` *filename*`.adb . . .`

Description `adbgen` makes it possible to write `adb(1)` scripts that do not contain hard-coded dependencies on structure member offsets. The input to `adbgen` is a file named *filename*`.adb` that contains header information, then a null line, then the name of a structure, and finally an `adb` script. `adbgen` only deals with one structure per file; all member names are assumed to be in this structure. The output of `adbgen` is an `adb` script in *filename*. `adbgen` operates by generating a C program which determines structure member offsets and sizes, which in turn generate the `adb` script.

The header lines, up to the null line, are copied verbatim into the generated C program. Typically, these are `#include` statements, which include the headers containing the relevant structure declarations.

The `adb` script part may contain any valid `adb` commands (see `adb(1)`), and may also contain `adbgen` requests, each enclosed in braces (`{ }`). Request types are:

- Print a structure member. The request form is {*member*`,` *format*`}.` *member* is a member name of the *structure* given earlier, and `format` is any valid `adb` format request or any of the `adbgen` format specifiers (such as `{POINTER}`) listed below. For example, to print the `p_pid` field of the *proc* structure as a decimal number, you would write `{p_pid,d}`.
- Print the appropriate `adb` format character for the given `adbgen` format specifier. This action takes the data model into consideration. The request form is {*format specifier*`}.` The valid `adbgen` format specifiers are:

`{POINTER}`	pointer value in hexadecimal
`{LONGDEC}`	long value in decimal
`{ULONGDEC}`	unsigned long value in decimal
`{ULONGHEX}`	unsigned long value in hexadecimal
`{LONGOCT}`	long value in octal
`{ULONGOCT}`	unsigned long value in octal

- Reference a structure member. The request form is {`*`*member*`,` *base*`}.` *member* is the member name whose value is desired, and *base* is an `adb` register name which contains the base address of the structure. For example, to get the `p_pid` field of the *proc* structure, you would get the *proc* structure address in an `adb` register, for example `<f`, and write `{*p_pid,<f}`.
- Tell `adbgen` that the offset is valid. The request form is `{OFFSETOK}`. This is useful after invoking another `adb` script which moves the `adb` *dot*.
- Get the size of the *structure*. The request form is `{SIZEOF}`. `adbgen` replaces this request with the size of the structure. This is useful in incrementing a pointer to step through an array of structures.

- Calculate an arbitrary C expression. The request form is {EXPR, *expression*}. adbgen replaces this request with the value of the expression. This is useful when more than one structure is involved in the script.
- Get the offset to the end of the structure. The request form is {END}. This is useful at the end of the structure to get adb to align the *dot* for printing the next structure member.

adbgen keeps track of the movement of the adb *dot* and generates adb code to move forward or backward as necessary before printing any structure member in a script. adbgen's model of the behavior of adb's *dot* is simple: it is assumed that the first line of the script is of the form *struct_address*/*adb text* and that subsequent lines are of the form +/*adb text*. The adb *dot* then moves in a sane fashion. adbgen does not check the script to ensure that these limitations are met. adbgen also checks the size of the structure member against the size of the adb format code and warns if they are not equal.

Options The following option is supported:

-m *model* Specifies the data type model to be used by adbgen for the macro. This affects the outcome of the {*format specifier*} requests described under DESCRIPTION and the offsets and sizes of data types. *model* can be ilp32 or lp64. If the -m option is not given, the data type model defaults to ilp32.

Operands The following operand is supported:

filename.adb Input file that contains header information, followed by a null line, the name of the structure, and finally an adb script.

Examples **EXAMPLE 1** A sample adbgen file.

For an include file x.h which contained

```
struct x {
        char    *x_cp;
        char    x_c;
        int    x_i;
};
```

then , an adbgen file (call it script.adb) to print the file x.h would be:

```
#include "x.h"
x
./"x_cp"16t"x_c"8t"x_i"n{x_cp,{POINTER}}{x_c,C}{x_i,D}
```

After running adbgen as follows,

```
% /usr/lib/adb/adbgen  script.adb
```

the output file script contains:

```
./"x_cp"16t"x_c"8t"x_i"nXC3+D
```

EXAMPLE 1 A sample adbgen file. *(Continued)*

For a macro generated for a 64-bit program using the `lp64` data model as follows,

```
% /usr/lib/adb/adbgen/ -m lp64  script.adb
```

the output file `script` would contain:

```
./"x_cp"16t"x_c"8t"x_i"nJC3+D
```

To invoke the script, type:

```
example% adb program
x$<script
```

Files

`/usr/platform/`*platform-name*`/lib/adb/*`
platform-specific `adb` scripts for debugging the 32-bit kernel

`/usr/platform/`*platform-name*`/lib/adb/sparcv9/*`
platform-specific `adb` scripts for debugging the 64-bit SPARC V9 kernel

`/usr/lib/adb/*`
`adb` scripts for debugging the 32-bit kernel

`/usr/lib/adb/sparcv9/*`
`adb` scripts for debugging the 64-bit SPARC V9 kernel

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	developer/debug/mdb

See Also

`adb(1)`, `kmdb(1)`, `uname(1)`, `attributes(5)`

Diagnostics

Warnings are given about structure member sizes not equal to `adb` format items and about badly formatted requests. The C compiler complains if a structure member that does not exist is referenced. It also complains about an ampersand before array names; these complaints may be ignored.

Notes

platform-name can be found using the `-i` option of `uname(1)`.

Bugs

`adb` syntax is ugly; there should be a higher level interface for generating scripts.

Structure members which are bit fields cannot be handled because C will not give the address of a bit field. The address is needed to determine the offset.

Name add_allocatable – add entries to allocation databases

Synopsis

```
/usr/sbin/add_allocatable [-f] [-s] [-d] -n name -t type -l device-list
    [-a authorization] [-c clean] [-o key=value]
```

Description

`add_allocatable` creates new entries for user allocatable devices that are to be managed by the device allocation mechanism. `add_allocatable` can also be used to update existing entries of such devices.

`add_allocatable` can also create and update entries for non-allocatable devices, such as printers, whose label range is managed by the device allocation mechanism.

`add_allocatable` can be used in shell scripts, such as installation scripts for driver packages, to automate the administrative work of setting up a new device.

Use `list_devices(1)` to see the names and types of allocatable devices, their attributes, and device paths.

Options

`-f`	Force an update of an already-existing entry with the specified information. `add_allocatable` exits with an error if this option is not specified when an entry with the specified device name already exists.
`-s`	Turn on silent mode. `add_allocatable` does not print any error or warning messages.
`-d`	If this option is present, `add_allocatable` updates the system-supplied default attributes of the device type specified with `-t`.
`-n` *name*	Adds or updates an entry for device that is specified by *name*.
`-t` *type*	Adds or updates device entries that are of a type that are specified by *type*.
`-l` *device-list*	Adds or updates device paths to the device that is specified with `-n`. Multiple paths in *device-list* must be separated by white spaces and the list must be quoted.
`-a` *authorization*	Adds or updates the authorization that is associated with either the device that is specified with `-n` or with devices of the type that is specified with `-t`. When more than one authorization is specified, the list must be separated by commas and must be quoted. When the device is not allocatable, *authorization* is specified with an asterisk (*) and must be quoted. When the device is allocatable by any user, *authorization* is specified with the at sign (@) and must be quoted. Default authorization is '@'.
`-c` *clean*	Specifies the `device_clean(5)` program *clean* to be used with the device that is specified with `-n` or with devices of the type that is specified with `-t`. The default clean program is `/bin/true`.

`-o` *key=value* Accepts a string of colon-separated *key=value* pairs for a device that is specified with `-n` or with devices of the type that is specified with `-t`. The following keys are currently interpreted by the system:

`minlabel` The minimum label at which the device can be used.

`maxlabel` The maximum label at which the device can be used.

`class` Specifies a logical grouping of devices. For example, all Sun Ray devices of all device types is a logical grouping. The `class` keyword has no default value.

`xdpy` Specifies the display name of the X session. This keyword is used to identify devices that are associated with the X session. The `xdpy` keyword has no default value.

Exit Status When successful, `add_allocate` returns an exit status of `0` (true). `add_allocate` returns a nonzero exit status in the event of an error. The exit codes are as follows:

`1` Invocation syntax error

`2` Unknown system error

`3` An entry already exists for the specified device. This error occurs only when the `-f` option is not specified.

`4` Permission denied. User does not have DAC or MAC access record updates.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/trusted
Interface Stability	See below.

The invocation is Uncommitted. The options are Uncommitted. The output is Not-an-Interface.

See Also `allocate(1)`, `deallocate(1)`, `list_devices(1)`, `remove_allocatable(1M)`, `attributes(5)`, `device_clean(5)`

Notes The functionality described on this manual page is available only if the system is configured with Trusted Extensions.

Name addbadsec – map out defective disk blocks

Synopsis `addbadsec [-p] [-a` *blkno* `[`*blkno*`]...] [-f` *filename*`]` *raw_device*

Description `addbadsec` is used by the system administrator to map out bad disk blocks. Normally, these blocks are identified during surface analysis, but occasionally the disk subsystem reports unrecoverable data errors indicating a bad block. A block number reported in this way can be fed directly into `addbadsec`, and the block will be remapped. `addbadsec` will first attempt hardware remapping. This is supported on SCSI drives and takes place at the disk hardware level. If the target is an IDE drive, then software remapping is used. In order for software remapping to succeed, the partition must contain an alternate slice and there must be room in this slice to perform the mapping.

It should be understood that bad blocks lead to data loss. Remapping a defective block does not repair a damaged file. If a bad block occurs to a disk-resident file system structure such as a superblock, the entire slice might have to be recovered from a backup.

Options The following options are supported:

`-a` Adds the specified blocks to the hardware or software map. If more than one block number is specified, the entire list should be quoted and block numbers should be separated by white space.

`-f` Adds the specified blocks to the hardware or software map. The bad blocks are listed, one per line, in the specified file.

`-p` Causes `addbadsec` to print the current software map. The output shows the defective block and the assigned alternate. This option cannot be used to print the hardware map.

Operands The following operand is supported:

raw_device The address of the disk drive (see `FILES`).

Files The raw device should be `/dev/rdsk/c?[t?]d?p0`. See `disks(1M)` for an explanation of SCSI and IDE device naming conventions.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	x86
Availability	system/core-os

See Also `disks(1M)`, `diskscan(1M)`, `fdisk(1M)`, `fmthard(1M)`, `format(1M)`, `attributes(5)`

Notes The `format(1M)` utility is available to format, label, analyze, and repair SCSI disks. This utility is included with the `addbadsec`, `diskscan(1M)`, `fdisk(1M)`, and `fmthard(1M)` commands available for x86. To format an IDE disk, use the DOS "format" utility; however, to label, analyze, or repair IDE disks on x86 systems, use the Solaris `format(1M)` utility.

Name add_drv – add a new device driver to the system

Synopsis

```
add_drv [-b basedir] [-c class_name]
     [-i 'identify_name...'] [-m 'permission','...']
     [-p 'policy'] [-P privilege] [-n] [-f] [-u] [-v] device_driver
```

Description The `add_drv` command is used to inform the system about newly installed device drivers.

Each device on the system has a name associated with it. This name is represented by the `name` property for the device. Similarly, the device may also have a list of driver names associated with it. This list is represented by the `compatible` property for the device.

The system determines which devices will be managed by the driver being added by examining the contents of the `name` property and the `compatible` property (if it exists) on each device. If the value in the `name` property does not match the driver being added, each entry in the `compatible` property is tried, in order, until either a match occurs or there are no more entries in the `compatible` property.

In some cases, adding a new driver may require a reconfiguration boot. See the `NOTES` section.

Aliases might require quoting (with double-quotes) if they contain numbers. See `EXAMPLES`.

The `/etc/minor_perm` File `add_drv` and `update_drv(1M)` read the `/etc/minor_perm` file to obtain permission information. The permission specified is applied to matching minor nodes created when a device bound to the driver is attached. A minor node's permission may be manually changed by `chmod(1)`. For such nodes, the specified permissions apply, overriding the default permissions specified via `add_drv` or `update_drv(1M)`.

The format of the `/etc/minor_perm` file is as follows:

name:*minor_name permissions owner group*

minor_name may be the actual name of the minor node, or contain shell metacharacters to represent several minor nodes (see `sh(1)`).

For example:

```
sd:* 0640 root sys
zs:[a-z],cu 0600 uucp uucp
mm:kmem 0640 root bin
```

The first line sets all devices exported by the `sd` node to `0640` permissions, owned by `root`, with group `sys`. In the second line, devices such as `a,cu` and `z,cu` exported by the `zs` driver are set to `0600` permission, owned by `uucp`, with group `uucp`. In the third line the `kmem` device exported by the `mm` driver is set to `0640` permission, owned by `root`, with group `bin`.

Running `add_drv` from a postinstall Script When running `add_drv` from within the context of a package's postinstall script, you must consider whether the package is being added to a system image or to a running system. When a package is being installed on a system image, the `BASEDIR` variable refers to the image's base

directory. In this situation, `add_drv` should be invoked with `-b $BASEDIR`. This causes `add_drv` only to update the image's system files; a reboot of the system or client would be required to make the driver operational.

When a package is being installed on the running system itself, the system files need to be updated, as in the case above. However, the running kernel can be informed of the existence of the new driver without requiring a reboot. To accomplish this, the postinstall script must invoke `add_drv` without the `-b` option. Accordingly, `postinstall` scripts invoking `add_drv` should be written thusly:

```
if [ "${BASEDIR:=/}" = "/" ]
then
        ADD_DRV="add_drv"
else
        ADD_DRV="add_drv -b ${BASEDIR}"
fi
$ADD_DRV [<options>] <driver>
```

...or, alternatively:

```
if [ "${BASEDIR:=/}" != "/" ]
then
         BASEDIR_OPT="-b $BASEDIR"
fi
         add_drv $BASEDIR_OPT [<options>] <driver>
```

The `-b` option is described below.

Options `-b` *basedir*
Installs the driver on the system with a root directory of *basedir* rather than installing on the system executing `add_drv`. This option is typically used in package post-installation scripts when the package is not being installed on the system executing the `pkgadd` command. The system using *basedir* as its root directory must reboot to complete the driver installation.

Note – The root file system of any non-global zones must not be referenced with the `-b` option. Doing so might damage the global zone's file system, might compromise the security of the global zone, and might damage the non-global zone's file system. See `zones(5)`.

`-c` *class_name*
The driver being added to the system exports the class *class_name*.

`-f`
Normally if a reconfiguration boot is required to complete the configuration of the driver into the system, `add_drv` will not add the driver. The force flag forces `add_drv` to add the driver even if a reconfiguration boot is required. See the `-v` flag.

`-i` *'identify_name'*
A white-space separated list of aliases for the driver *device_driver*.

`-m` *'permission'*
Specify the file system permissions for device nodes created by the system on behalf of *device_driver*.

`-n`
Do not try to load and attach *device_driver*, just modify the system configuration files for the *device_driver*.

`-p` *'policy'*
Specify an additional device security policy.

The device security policy constists of several whitespace separated tokens:

`{`*minorspec* `{token=value}+}+`

minorspec is a simple wildcard pattern for a minor device. A single * matches all minor devices. Only one * is allowed in the pattern.

Patterns are matched in the following order:

- entries without a wildcard
- entries with wildcards, longest wildcard first

The following tokens are defined: `read_priv_set` and `write_priv_set`. `read_priv_set` defines the privileges that need to be asserted in the effective set of the calling process when opening a device for reading. `write_priv_set` defines the privileges that need to be asserted in the effective set of the calling process when opening a device for writing. See `privileges(5)`.

A missing minor spec is interpreted as a *.

`-P` *'privilege'*
Specify additional, comma separated, privileges used by the driver. You can also use specific privileges in the device's policy.

`-u`
Add the driver to the system, leaving it in an inactive state for later configuration with `devfsadm(1M)` `-u`. The `-u` behavior differs from `-n` in that `-n` only updates the system files, requiring a reboot to attach the driver. Drivers added with `-u` can be attached by running `devfsadm -u` without rebooting. Driver writers should verify their driver with this behavior. See `NOTES` for additional considerations. The `-u` option cannot be used together with `-n` or `-b`.

`-v`
The verbose flag causes `add_drv` to provide additional information regarding the success or failure of a driver's configuration into the system. See the `EXAMPLES` section.

Examples

EXAMPLE 1 Adding SUNW Example Driver to the System

The following example adds the `SUNW,example` driver to a 32–bit system, with an alias name of `SUNW,alias`. It assumes the driver has already been copied to `/usr/kernel/drv`.

```
example# add_drv -m '* 0666 bin bin','a 0644 root sys' \
      -p 'a write_priv_set=sys_config  * write_priv_set=none' \
      -i 'SUNW,alias' SUNW,example
```

Every minor node created by the system for the `SUNW,example` driver will have the permission `0666`, and be owned by user `bin` in the group `bin`, except for the minor device `a`, which will be owned by `root`, group `sys`, and have a permission of `0644`. The specified device policy requires no additional privileges to open all minor nodes, except minor device `a`, which requires the `sys_config` privilege when opening the device for writing.

EXAMPLE 2 Adding Driver to the Client /export/root/sun1

The following example adds the driver to the client `/export/root/sun1`. The driver is installed and loaded when the client machine, `sun1`, is rebooted. This second example produces the same result as the first, except the changes are on the diskless client, `sun1`, and the client must be rebooted for the driver to be installed.

```
example# add_drv -m '* 0666 bin bin','a 0644 root sys' \
        -i 'SUNW,alias' -b /export/root/sun1 \
    SUNW,example
```

See the note in the description of the `-b` option, above, specifying the caveat regarding the use of this option with the Solaris zones feature.

EXAMPLE 3 Adding Driver for a Device Already Managed by an Existing Driver

The following example illustrates the case where a new driver is added for a device that is already managed by an existing driver. Consider a device that is currently managed by the driver `dumb_framebuffer`. The `name` and `compatible` properties for this device are as follows:

```
name="display"
compatible="whizzy_framebuffer", "dumb_framebuffer"
```

If `add_drv` is used to add the `whizzy_framebuffer` driver, the following will result.

```
example# add_drv whizzy_framebuffer
Error: Could not install driver (whizzy_framebuffer)
Device managed by another driver.
```

If the `-v` flag is specified, the following will result.

```
example# add_drv -v whizzy_framebuffer
Error: Could not install driver (whizzy_framebuffer)
Device managed by another driver.
Driver installation failed because the following
entries in /devices would be affected:
```

EXAMPLE 3 Adding Driver for a Device Already Managed by an Existing Driver *(Continued)*

```
        /devices/iommu@f,e0000000/sbus@f,e0001000/display[:*]
        (Device currently managed by driver "dumb_framebuffer")

The following entries in /dev would be affected:

        /dev/fbs/dumb_framebuffer0
```

If the `-v` and `-f` flags are specified, the driver will be added resulting in the following.

```
example# add_drv -vf whizzy_framebuffer
A reconfiguration boot must be performed to complete the
installation of this driver.

The following entries in /devices will be affected:

        /devices/iommu@f,e0000000/sbus@f,e0001000/display[:*]
        (Device currently managed by driver "dumb_framebuffer"

The following entries in /dev will be affected:

        /dev/fbs/dumb_framebuffer0
```

The above example is currently only relevant to devices exporting a generic device name.

EXAMPLE 4 Use of Double Quotes in Specifying Driver Alias

The following example shows the use of double quotes in specifying a driver alias that contains numbers.

```
example# add_drv -i '"pci10c5,25"' smc
```

Exit Status

`add_drv` returns `0` on success and `1` on failure.

Files

`/kernel/drv`
: 32–bit boot device drivers

`/kernel/drv/sparcv9`
: 64–bit SPARC boot device drivers

`/kernel/drv/amd64`
: 64–bit x86 boot device drivers

`/usr/kernel/drv`
: other 32–bit drivers that could potentially be shared between platforms

`/usr/kernel/drv/sparcv9`
: other 64–bit SPARC drivers that could potentially be shared between platforms

`/usr/kernel/drv/amd64`
other 64–bit x86 drivers that could potentially be shared between platforms

`/platform/‘uname -i‘/kernel/drv`
32–bit platform-dependent drivers

`/platform/‘uname -i‘/kernel/drv/sparcv9`
64–bit SPARC platform-dependent drivers

`/platform/‘uname -i‘/kernel/drv/amd64`
64–bit x86 platform-dependent drivers

`/etc/driver_aliases`
driver aliases file

`/etc/driver_classes`
driver classes file

`/etc/minor_perm`
minor node permissions

`/etc/name_to_major`
major number binding

`/etc/security/device_policy`
device policy

`/etc/security/extra_privs`
device privileges

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also `boot(1M)`, `chmod(1)`, `devfsadm(1M)`, `kernel(1M)`, `modinfo(1M)`, `rem_drv(1M)`, `update_drv(1M)`, `driver.conf(4)`, `system(4)`, `attributes(5)`, `privileges(5)`, `devfs(7FS)`, `ddi_create_minor_node(9F)`

Writing Device Drivers

Notes It is possible to add a driver for a device already being managed by a different driver, where the driver being added appears in the device's `compatible` list before the current driver. In such cases, a reconfiguration boot is required (see `boot(1M)` and `kernel(1M)`). After the reconfiguration boot, device links in `/dev` and references to these files may no longer be valid (see the `-v` flag). If a reconfiguration boot would be required to complete the driver installation, `add_drv` will fail unless the `-f` option is specified. See `Example 3` in the `EXAMPLES` section.

With the introduction of the device policy several drivers have had their minor permissions changed and a device policy instated. The typical network driver should use the following device policy:

```
add_drv -p 'read_priv_set=net_rawaccess\
   write_priv_set=net_rawaccess' -m '* 666 root sys'\
   mynet
```

This document does not constitute an API. `/etc/minor_perm`, `/etc/name_to_major`, `/etc/driver_classes`, and `/devices` may not exist or may have different contents or interpretations in a future release. The existence of this notice does not imply that any other documentation that lacks this notice constitutes an API.

`/etc/minor_perm` can only be updated by `add_drv`(1M), `rem_drv(1M)` or `update_drv(1M)`.

In the current version of `add_drv`, the use of double quotes to specify an alias is optional when used from the command line. However, when using `add_drv` from packaging scripts, you should continue to use double quotes to specify an alias.

Some drivers should not be added and configured on the system directly, but should only be configured as the system boots. The reasons for this restriction include, but are not limited to, a driver dependency on configuration early during boot or a dependency on some kernel component being installed or updated at the same time as the driver is being added. Such drivers should only be added to the system with the `-n` flag, so the driver is only loaded and configured when the system is rebooted, thus assuring an environment in which the driver can be configured properly.

Bugs Previous versions of `add_drv` accepted a pathname for *device_driver*. This feature is no longer supported and results in failure.

Name aimanifest – Modify an XML file used by Automated Installer (AI)

Synopsis

```
/usr/bin/aimanifest [-h]

aimanifest add [-r] path value

aimanifest get [-r] path

aimanifest set [-r] path value

aimanifest load [-i] filename

aimanifest validate
```

Description The `aimanifest` command creates a new XML manifest or modifies an existing one. While `aimanifest` can be used with any XML file that contains a valid `!DOCTYPE` reference to a DTD definition, it is intended for creating derived manifests used by the Automated Installer (AI). See *Installing Oracle Solaris 11 Systems* for information about AI derived manifests.

The `aimanifest` command can be invoked multiple times to develop a manifest. The `AIM_MANIFEST` environment variable specifies the location of the manifest for `aimanifest` to modify. `AIM_MANIFEST` must be set. Each invocation of the `aimanifest` command with the `load`, `add`, or `set` subcommand opens, modifies, and saves the `AIM_MANIFEST` file.

The minimum `AIM_MANIFEST` file that the `aimanifest` command can modify must contain both of the following pieces:

- A `!DOCTYPE` reference to a DTD that is valid for the XML manifest being developed.
- The root element for this manifest.

If you start with an empty `AIM_MANIFEST` file, as when AI is executing a derived manifests script, then the first `aimanifest` command must specify the `load` subcommand to load at least the minimum required `AIM_MANIFEST` file. Subsequent `aimanifest` commands that modify the manifest use the DTD to determine where to add elements in the developing manifest.

To save error and informational messages to a file in addition to displaying messages to `stdout` and `stderr`, set the `AIM_LOGFILE` environment variable to a log file location. Information is appended to the log file. The log file is not cleared.

Options The `aimanifest` command has the following option:

`-h`, `--help`
Show the usage help message.

The `add`, `get`, and `set` subcommands of the `aimanifest` command have the following option:

`-r`, `--return-path`
Return the path of the XML element that this `aimanifest` command creates or operates on. This returned path is a chain of node IDs. You can save this returned path value to use in subsequent calls to `aimanifest`. Using the path returned by the `-r` option is more reliable than specifying the path using XML element and attribute values, since the values can

change as the AI manifest is being built. See the "Return Paths" section for more information about the path returned by the `-r` option.

The `load` subcommand of the `aimanifest` command has the following option:

`-i`, `--incremental`
Do not clear the `AIM_MANIFEST` data before adding new data.

Sub-commands The following subcommands are supported:

`aimanifest add [-r | --return-path]` *path value*
Add a new element to an XML manifest. Add the new element at *path* and with value *value*. See the "Operands" section for more information about *path*. If *path* ends in an attribute (`@`*attr*), then the new element has the *attr* attribute, and *value* is the value of the attribute.

No validation is performed except to examine parent/child relationships in *path*.

The `-r` option returns a path to the newly-added node. See the "Return Paths" section for more information.

If the parent path matches an element in the `AIM_MANIFEST` file, it must match only one element. The new element is created as a child of the matching parent element. The path can specify element and attribute values to match a unique parent element, as shown in "Example 2: Path With a Value" in this section.

If the parent path does not match an element in the `AIM_MANIFEST` file, new elements are created as necessary, and the new child element is added to the new parent. The path to an added element is split off from the preexisting elements according to the following rules:

- The split occurs after all parts of the path that specify a value.
- The split occurs at the first place where multiple relevant same-tagged elements are allowed by the DTD, after all parts of the path that specify a value.

Use this XML manifest schema to analyze the following examples:

- The manifest begins with a single A node.
- The A node can have only one B node child.
- The B node can have multiple C node children.
- A C node can have multiple D node children.

Example 1: Simple Path. The AI manifest has one A node, one B node, and one C node: `/A/B/C`. An `add` subcommand is issued with a *path* of `/A/B/C/D`. In this case, a new C node is created because C nodes are the first nodes along the path that can have same-tagged siblings. A new D node is added as a child to the new C node. The resulting manifest has the structure `/A/B/{C,C/D}`. Issuing the same command for a different value of D results in three C nodes: `/A/B/{C,C/D,C/D}`.

Example 2: Path With a Value. The AI manifest has one A node, one B node, and two C nodes. Only one of the C nodes has a value of 1 so that the manifest has the structure `/A/B/{C,C=1}`. An `add` subcommand is issued with a *path* of `/A/B/C=1/D` and a *value* of 10.

In this case, no new C node is added because specifying the value of 1 for C identifies a unique node, and the path cannot be split at or before a branch where a value is specified. The first place where this path can be split is at D. A new D node with a value of 10 is added as a child of the C node that has a value of 1. The resulting manifest has the structure `/A/B/{C,C=1/D=10}`. Issuing the same command with a value of 20 for D results in `/A/B/{C,C=1/{D=10,D=20}}`.

`aimanifest get [-r | --return-path]` *path*
Retrieve an element or attribute value. An empty string ("") is displayed for empty element or attribute values. The *path* must match a unique existing element or attribute. See the "Operands" section for more information about *path*.

The `-r` option returns a path to the accessed node as a second returned string. See the "Return Paths" section for more information.

`aimanifest set [-r | --return-path]` *path value*
Change the value of an existing element or attribute, or create a new attribute of an existing element. No validation is performed.

When changing the value of an existing element, *path* must match a unique existing element. If the element has same-tagged siblings, use an element value or attribute, or a child element of the target element to make the path unique. See "The Path Operand" section.

When setting the value of an attribute, the attribute does not need to exist, but the element to which the attribute belongs must exist.

The `-r` option returns a path to the changed element. See the "Return Paths" section for more information.

`aimanifest load [-i | --incremental]` *filename*
Load an XML manifest or partial XML manifest from the file *filename*. No validation is performed except to examine parent/child relationships of elements.

When the `-i` option is not specified, overwrite any existing XML data. All data in the `AIM_MANIFEST` file is replaced with the contents of the *filename* file. The *filename* file must include a `!DOCTYPE` reference to a DTD so that subsequent `aimanifest` commands can modify the file.

When the `-i` option is specified, do not clear the `AIM_MANIFEST` data before adding new data. Instead, incrementally insert or merge the new data with the existing XML data. The DTD given by the `!DOCTYPE` reference in `AIM_MANIFEST` is used to determine how and where to merge the *filename* data. If the `!DOCTYPE` reference is missing, the AI manifest DTD at `/usr/share/install/ai.dtd` is used. If the data in *filename* cannot be reconciled with the DTD, a non-zero error status is returned.

The following considerations affect where new data is inserted into the AIM_MANIFEST manifest:

- To what extent the tags of elements near the beginning of the AIM_MANIFEST data paths and *filename* data paths match
- What child elements are allowed under those AIM_MANIFEST data elements
- Where same-tagged sibling elements are allowed
- Where childless AIM_MANIFEST data nodes are located

As each element of *filename* data is processed, if all of the following conditions are true, then in general a new node is not created for this element in the AIM_MANIFEST data. Instead, an existing node is replaced with the new data.

- Both sets of data contain a node with the same tag and same location.
- The DTD given by the !DOCTYPE reference in AIM_MANIFEST does not allow both of these nodes to exist together as same-tagged sibling elements.
- The *filename* data element has children.

When an element from *filename* is inserted, the split where new nodes start to be created is done as close as possible to the AIM_MANIFEST data root. The first new node of the split is created at the earliest point where same-tagged sibling elements are allowed, or at the earliest appropriate point when no same-tagged element exists in AIM_MANIFEST.

Use this XML manifest schema to analyze the following examples:

- The manifest begins with a single A node.
- The A node can have only one B node child.
- The B node can have multiple C node children.
- The B node can have only one E node child.

Example 1: Inserting Same-Tagged Elements. If the content of AIM_MANIFEST is /A/B/C1/D1 and the content of *filename* is /A/B/C2/D2, then after the load -i command, the content of the AIM_MANIFEST file is /A/B/{C1/D1,C2/D2}. The C node is the first place where new nodes can be added. The C node from the *filename* data is added after the existing C node in the AIM_MANIFEST data. If the two A elements have different values or if the two B elements have different values, the value of the *filename* element replaces the value of the AIM_MANIFEST element. If the two A elements have different attributes, or if the two B elements have different attributes, the attribute values are merged.

- Attributes of A and B that exist in both the AIM_MANIFEST file and the *filename* file have the values from the *filename* file in the merged file.
- Attributes of A and B that exist in either the AIM_MANIFEST file or the *filename* file but not in both files are all retained in the merged file.

Example 2: Inserting Differently Tagged Elements. If the content of AIM_MANIFEST is /A/B/C/D and the content of *filename* is /A/B/E/F, then after the load -i command, the

content of the AIM_MANIFEST file is /A/B/{E/F,C/D}. The E node is added at the first location where it is allowed by the DTD. The values of elements A and B are the values from *filename*, and the attributes of A and B are merged from *filename* to AIM_MANIFEST as described in Example 1 above.

Sometimes the correct merge location cannot be determined. This can happen if a sibling that is required to follow a node to be merged has not yet been added. To avoid this issue, add multiple nodes or subtrees to a common parent node in the order mandated by the DTD. A node is placed at the end of its list of new siblings if its proper place among them cannot be determined.

`aimanifest validate`
Validates the AIM_MANIFEST manifest against the DTD referenced in the !DOCTYPE statement. Errors are printed to stderr. A non-zero status is returned if validation fails.

Operands The following operands are required.

The Filename Operand The load subcommand requires the *filename* operand, which is the name of a full or partial manifest to load to the AIM_MANIFEST manifest.

The Value Operand The add and set subcommands require the *value* operand. The *value* operand is a valid value of the element or attribute specified by the *path* operand.

The Path Operand The add, get, and set subcommands of the aimanifest command require the *path* operand. The path defines a node in an XML hierarchy of elements and attributes.

The XML element hierarchy structure is also called an XML tree. In the following partial AI manifest, the auto_install element is the root of the tree, and the ai_instance and software elements are branches or the roots of subtrees.

```
<auto_install>
  <ai_instance>
    <software type="IPS"/>
  </ai_instance>
</auto_install>
```

In aimanifest path syntax, use forward slash characters (/) to indicate branches in the tree structure. In the current example, the path to the software element is /auto_install/ai_instance/software.

Attributes are bound to an element. In aimanifest path syntax, use an at symbol (@) to identify an attribute name. The path to the type attribute of the software element is /auto_install/ai_instance/software@type.

An aimanifest *path* operand must correspond to a single element. Include element and attribute values as necessary to make the path unique. For example, to specify a size for the second slice defined in the following partial AI manifest, you could use the path /auto_install/ai_instance/target/disk/slice[@name="4"]/size@val to identify which slice you are specifying the size for.

```
<auto_install>
  <ai_instance>
    <target>
        <disk>
          <slice name="0"/>
          <slice name="4"/>
        </disk>
    </target>
  </ai_instance>
</auto_install>
```

Relative paths are permitted. The `slice` path shown in the previous paragraph could be specified starting at `ai_instance`, `target`, `disk`, or `slice` , since there is only one `slice` with a `name` attribute value of 4. For example, you could use the path `slice[@name="4"]/size@val`.

If a *value* within a *path* contains forward slash characters, then that value must be enclosed in single or double quotation marks, as in `/name="pkg:/entire"`.

When the `aimanifest` call is in a shell script, values that contain quotation marks might require additional special treatment. Within a shell script, quotation marks in `aimanifest` path values might need to be escaped with a preceding backslash character (\) so that the shell does not remove or interpret the quotation marks. Check the rules of the shell you are using. The following example shows a value with a forward slash character in a `ksh93` script:

```
/usr/bin/aimanifest get software_data[name=\"pkg:/entire\"]@action
```

Most examples in this man page omit backslash escape characters because this man page does not assume that `aimanifest` is being called in a script or in a particular shell. See *Installing Oracle Solaris 11 Systems* for information about AI derived manifests scripts.

The following forms of branches show how to construct a path to an element or element attribute.

`/A`
: `A` is the tag name of an element, as in `/auto_install`. This branch specification is also called a simple branch. Paths with only simple branches are called simple paths.

`/A=`*value*
: `A` is the tag name of an element, and *value* is the value of that element, as in `/name="pkg:/entire"`.

`/A[B/C=`*value*`]`
: `A` is an element, `B` is an element that is a child of `A`, `C` is an element that is a child of `B`, and *value* is the value of the `C` element. This path form specifies the `A` element that has a grandchild element `C` that has value *value*. For example, if your AI manifest has more than one software section, you could use this form to operate on the software section that installs package `pkg:/entire`, as in the following path:

  ```
  software[software_data/name="pkg:/entire"]
  ```

`/A[@Aattr=`*value*`]`
A is an element, `Aattr` is an attribute of `A`, and *value* is the value of the `Aattr` attribute. This path form specifies the `A` element that has attribute `Aattr` with value *value*. For example, if your AI manifest defines more than one slice, you could use this form to operate on the slice that has a `name` value of 4, as in `slice[@name="4"]`

`/A[B/C@Cattr=`*value*`]`
`A` is an element, `B` is a child of `A`, `C` is a child of `B`, `Cattr` is an attribute of `C`, and *value* is the value of the `Cattr` attribute. This path form specifies the `A` element that has a grandchild element `C` that has attribute `Cattr` with value *value*. For example, if your AI manifest has more than one software section, you could use this form to operate on the software section that has a publisher section with a name value of `solaris`, as in the path `software[source/publisher@name="solaris"]`.

`/A[1]`
`/A[1]` specifies the first instance of an `A` element in the manifest. For example, if your AI manifest has more than one software section, you could use this form to operate on the second software section, as in `/auto_install[1]/ai_instance[1]/software[2]`.

This is the form of path that is returned by the `-r` option. See the "Return Paths" section.

`/A@Aattr`
This path specifies the `Aattr` attribute of the `A` element. This path does not specify the `A` element but rather the `Aattr` attribute. Use this form to set or get the `Aattr` attribute.

`/A[B/C=`*value*`]@Aattr`
This path specifies the `Aattr` attribute of the `A` element that has a grandchild element `C` that has value *value*.

`/A[B/C@Cattr=`*value*`]@Aattr`
This path specifies the `Aattr` attribute of the `A` element that has a grandchild element `C` that has attribute `Cattr` with value *value*.

`/A/B=`*value*`@Battr`
This path specifies the `Battr` attribute of the `B` element with value *value*. The `B` element is a child of the `A` element.

Return Paths With the `-r` option, the `add`, `get`, and `set` subcommands return the address of the element that was created or accessed by the subcommand. This returned address is in the form of a chain of node IDs. This returned address can be used to access the same element again, even if values associated with that element have changed.

The following examples show that the address returned by the `-r` option can be much easier to use than a path that specifies element and attribute values. Start with the following node tree:

```
auto_install
     |
ai_instance
     |
```

```
          target
            |
          disk
        attribute: whole_disk=true
            |
        disk_name
        attribute: name=data1
        attribute: name_type=volid
```

Add a new `disk` node with `name` attribute value `data2` and `name_type` attribute value `volid`:

```
        auto_install
             |
        ai_instance
             |
          target
            |
  |----------|----------|
disk                  disk
whole_disk=true       whole_disk=true
    |                     |
disk_name             disk_name
name=data1            name=data2
name_type=volid       name_type=volid
```

A new `disk_name` element with one attribute can be added easily with a single command. To add the second and third attributes, you must specify which `disk_name` element to change. Compare the following two methods for accessing the same node multiple times.

Specifying Paths By Using Values

The commands in this example specify paths using values. Note that you must assign a unique value in the first command so that you can use that value to specify a unique path in the subsequent commands. This method could yield an incorrect result if the values are changed.

```
$ aimanifest add target/disk/disk_name@name data2
$ aimanifest set \
> target/disk/disk_name[@name=data2]@name_type volid
$ aimanifest set \
> target/disk[disk_name@name=data2]@whole_disk true
```

Specifying Paths By Using Returned Paths

The most reliable way to access the same node multiple times is to save the path to the new `disk_name` element, and then use that saved path for subsequent accesses.

```
$ NewDisk=$(aimanifest add -r target/disk@whole_disk true)
$ aimanifest add ${NewDisk}/disk_name@name data2
$ aimanifest add ${NewDisk}/disk_name@name_type volid
```

The path that is returned to `$NewDisk` through the `-r` option expresses the node in terms of IDs and is free of values:

```
$ aimanifest add -r target/disk/@whole_disk true
/auto_install[1]/ai_instance[1]/target[1]/disk[2]
```

Examples

To try these examples, you need to set AIM_MANIFEST.

```
$ export AIM_MANIFEST=/tmp/aimtest.xml
```

The minimum AIM_MANIFEST file that the aimanifest command can modify must contain both of the following pieces:

- A !DOCTYPE reference to a DTD that is valid for the XML manifest being developed.
- The root element for this manifest.

The following example shows the minimum AIM_MANIFEST manifest file for an AI manifest:

```
<!DOCTYPE auto_install SYSTEM "file:///usr/share/install/ai.dtd">
<auto_install/>
```

Usually, you will use the aimanifest command in a derived manifests script that operates on an existing valid AI manifest. To try these examples, you can copy /usr/share/auto_install/manifest/default.xml and then define AIM_MANIFEST to refer to this copy. Make sure the copy is writable.

EXAMPLE 1 Set the auto_reboot Attribute

```
$ aimanifest set /auto_install/ai_instance@auto_reboot false
```

EXAMPLE 2 Get the auto_reboot Value

```
$ aimanifest get /auto_install/ai_instance@auto_reboot
false
```

EXAMPLE 3 Add a Publisher by Using Values Paths

The package repository in this example is a file repository at file:///net/host2/export/extras_repo. The publisher is extras. Since a software element can have only one source element, this example adds the new publisher element to the source element that contains the solaris publisher.

```
$ aimanifest add \
> software[@type=IPS]/source[publisher@name=solaris]/publisher@name \
extras
$ aimanifest add \
> publisher[@name=extras]/origin@name \
> file:///net/host2/export/extras_repo
```

These aimanifest commands result in the following AI manifest entries if you started with the default.xml AI manifest. The destination and software_data elements are omitted for brevity.

```
<software type="IPS">
  <source>
    <publisher name="solaris">
      <origin name="http://pkg.oracle.com/solaris/release"/>
```

EXAMPLE 3 Add a Publisher by Using Values Paths *(Continued)*

```
    </publisher>
    <publisher name="extras">
      <origin name="file:///net/host2/export/extras_repo"/>
    </publisher>
  </source>
</software>
```

EXAMPLE 4 Add a Publisher by Using Returned Paths

This example is the same as the previous example but uses a different method to achieve the same result.

```
$ NEW_PUB=$(aimanifest add -r \
> software[@type=IPS]/source[publisher@name=solaris]/publisher@name \
extras)
$ echo $NEW_PUB
/auto_install[1]/ai_instance[1]/software[1]/source[1]/publisher[2]
$ aimanifest add ${NEW_PUB}/origin@name \
file:///net/host2/export/extras_repo
```

EXAMPLE 5 Add a Publisher By Adding a Manifest Fragment

This example adds the `extras` publisher by loading a file that contains a partial AI manifest. In this case, the result is a separate, additional `software` element of type IPS with the `extras` publisher defined. This new `software` element is inserted after the original IPS `software` element that defines the `solaris` publisher. Packages named in `software_data` elements within this new `software` element are only searched for from the `extras` publisher or other publishers defined in this new `software` element. This manifest fragment also defines a package to install, since a `software` element with no software to install is not useful.

Create a file named `extras.xml` with the following content:

```
<auto_install>
  <ai_instance>
    <software type="IPS">
      <source>
        <publisher name="extras">
          <origin name="file:///net/host2/export/extras_repo"/>
        </publisher>
      </source>
      <software_data action="install">
        <name>pkg:/package/from/extras_repo</name>
      </software_data>
    </software>
  </ai_instance>
</auto_install>
```

EXAMPLE 5 Add a Publisher By Adding a Manifest Fragment *(Continued)*

Even though you only want the `software` section, you must include the `auto_install` and `ai_instance` elements as well. If the loaded file specifies attributes for the `auto_install` or `ai_instance` elements, then those attribute values replace existing values or are added.

Use the following command to add this `software` section to the `AIM_MANIFEST` manifest:

```
$ aimanifest load -i extras.xml
```

EXAMPLE 6 Add a Package by Using a Values Path

This example adds a package to the `software` element that has a `publisher` element with name `solaris` by specifying the publisher name as a value in the path.

```
$ aimanifest add \
> software[source/publisher@name=solaris]/software_data/name \
> pkg:/system/utils
```

This `aimanifest` command adds the second `software_data` element shown below if you started with the `default.xml` AI manifest.

```
<software_data action="install">
  <name>pkg:/entire@latest</name>
  <name>pkg:/group/system/solaris-large-server</name>
</software_data>
<software_data>
  <name>pkg:/system/utils</name>
</software_data>
```

EXAMPLE 7 Add a Package by Using a Returned Path

This example is the same as the previous example but uses a different method to achieve the same result. This example uses the `get` subcommand with the returned path option to add the package to the `software` element where the `solaris` publisher is defined.

```
$ NEW_PKG=$(aimanifest get -r \
software[source/publisher@name=solaris] | awk '{print $2 }')
$ echo $NEW_PKG
/auto_install[1]/ai_instance[1]/software[1]
$ aimanifest add ${NEW_PKG}/software_data/name \
pkg:/system/utils
```

EXAMPLE 8 Add a Package By Adding a Manifest Fragment

This example adds the package by loading a file that contains a partial AI manifest. In this case, the result is a separate, additional `software` element of type IPS inserted after the original IPS `software` element. This new `software` element contains only a `software_data` element; no `source` element is specified. Packages named in `software_data` elements within this new `software` element are searched for from publishers defined in the preceding `software` element.

EXAMPLE 8 Add a Package By Adding a Manifest Fragment *(Continued)*

Create a file named `newpkg.xml` with the following content:

```
<auto_install>
  <ai_instance>
    <software type="IPS">
      <software_data>
        <name>pkg:/system/utils</name>
      </software_data>
    </software>
  </ai_instance>
</auto_install>
```

Even though you only want the `software` section, you must include the `auto_install` and `ai_instance` elements as well. If the loaded file specifies attributes for the `auto_install` or `ai_instance` elements, then those attribute values replace existing values or are added.

Use the following command to add this `software` section to the `AIM_MANIFEST` manifest:

```
$ aimanifest load -i newpkg.xml
```

EXAMPLE 9 Validate a Manifest

Validate the `AIM_MANIFEST` manifest.

```
$ aimanifest validate
```

Exit Status

The following exit values are returned:

`0` The command was processed successfully.

`>0` An error occurred.

Files

`AIM_MANIFEST`
The value of this environment variable is the location of the AI manifest that is being built.

`AIM_LOGFILE`
The value of this environment variable is the location of the log file of `aimanifest` operations.

Attributes

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	`system/install/auto-install/auto-install-common`
Interface Stability	Uncommitted

See Also `installadm`(1M)

Part III, "Installing Using an Install Server," in *Installing Oracle Solaris 11.1 Systems*

Name arp – address resolution display and control

Synopsis

```
arp hostname

arp -a [-n]

arp -d hostname

arp -f filename

arp -s hostname ether_address [temp] [pub] [trail]
     [permanent]
```

Description The `arp` program displays and modifies the Internet-to-MAC address translation tables used by the address resolution protocol (see `arp(7P)`).

With no flags, the program displays the current ARP entry for *hostname*. The host may be specified by name or by number, using Internet dot notation.

Options that modify the ARP translation tables (`-d`, `-f`, and `-s`) can be used only when the invoked command is granted the `sys_net_config` privilege. See `privileges(5)`.

Options

`-a` Display all of the current ARP entries. The definition for the flags in the table are:

- `d` Unverified; this is a local IP address that is currently undergoing Duplicate Address Detection. ARP will not respond to requests for this address until Duplicate Address Detection completes.
- `o` Old; this entry is aging away. If IP requests it again, a new ARP query will be generated. This state is used for detecting peer address changes.
- `y` Delayed; periodic address defense and conflict detection was unable to send a packet due to internal network use limits for non-traffic-related messages (100 packets per hour per interface). This occurs only on interfaces with very large numbers of aliases.
- `A` Authority; this machine is authoritative for this IP address. ARP will not accept updates from other machines for this entry.
- `L` Local; this is a local IP address configured on one of the machine's logical interfaces. ARP will defend this address if another node attempts to claim it.
- `M` Mapping; only used for the multicast entry for `224.0.0.0`
- `P` Publish; includes IP address for the machine and the addresses that have explicitly been added by the `-s` option. ARP will respond to ARP requests for this address.
- `S` Static; entry cannot be changed by learned information. This indicates that the `permanent` flag was used when creating the entry.
- `U` Unresolved; waiting for ARP response.

You can use the `-n` option with the `-a` option to disable the automatic numeric IP address-to-name translation. Use `arp -an` or `arp -na` to display numeric IP addresses. The `arp -a` option is equivalent to:

```
# netstat -p -f inet
```

...and `-an` and `-na` are equivalent to:

```
# netstat -pn -f inet
```

`-d` Delete an entry for the host called *hostname*.

Note that ARP entries for IPMP (IP Network Multipathing) data and test addresses are managed by the kernel and thus cannot be deleted.

`-f` Read the file named *filename* and set multiple entries in the ARP tables. Entries in the file should be of the form:

hostname MACaddress `[temp] [pub] [trail] [permanent]`

See the `-s` option for argument definitions.

`-s` Create an ARP entry for the host called *hostname* with the MAC address *MACaddress*. For example, an Ethernet address is given as six hexadecimal bytes separated by colons. The entry will not be subject to deletion by aging unless the word `temp` is specified in the command. If the word `pub` is specified, the entry will be published, which means that this system will respond to ARP requests for *hostname* even though the *hostname* is not its own. The word `permanent` indicates that the system will not accept MAC address changes for *hostname* from the network.

Solaris does not implement trailer encapsulation, and the word `trail` is accepted on entries for compatibility only.

`arp -s` can be used for a limited form of proxy ARP when a host on one of the directly attached networks is not physically present on a subnet. Another machine can then be configured to respond to ARP requests using `arp -s`. This is useful in certain SLIP configurations.

Non-temporary proxy ARP entries for an IPMP (IP Network Multipathing) group are automatically managed by the kernel. Specifically, if the hardware address in an entry matches the hardware address of an IP interface in an IPMP group, and the IP address is not local to the system, this will be regarded as an IPMP proxy ARP entry. This entry will have its hardware address automatically adjusted in order to keep the IP address reachable so long as the IPMP group has not entirely failed.

ARP entries must be consistent across an IPMP group. Therefore, ARP entries cannot be associated with individual underlying IP interfaces in an IPMP group, and must instead be associated with the corresponding IPMP IP interface.

Note that ARP entries for IPMP data and test addresses are managed by the kernel and thus cannot be changed.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also ifconfig(1M), netstat(1M), attributes(5), privileges(5), arp(7P)

Name asradm – Auto Service Request registration utility

Synopsis

```
asradm list

asradm send [-n] [activate|audit|deactivate|fault|heartbeat|test]

asradm register -u username [-p password-file] [-e endpoint_URL]

asradm set-proxy [-i] | [-h host[:port]] [-u username]
     [-p password-file]

asradm unregister

asradm authenticate -u username -p password-file [-n]
     [-e endpoint_URL]
```

Description The `asradm` utility is used by a privileged system administrator to register hosts for enrollment in the Oracle Auto Service Request (ASR) for Oracle Sun systems. Using auto-case generation improves system availability and expedites the diagnostic process when specific hardware faults occur.

Once the system has been registered with a valid My Oracle Support (MOS) account the `smf(5)` service, `asr-notify`, will start to send HTTPS/XML telemetry either directly to the configured endpoint or through an optional HTTP proxy.

The `asradm` utility can also list the current registration state as seen from the registered server. This only confirms that messages are being sent to the Oracle ASR service and does not mean the system is actively being monitored. The user must log into their MOS account and activate the service for the system. An email should be sent to the user after registration that will describe the specific details needed to complete enrollment.

ASR messages can be sent manually using the `send` subcommand. The content of ASR messages can also be viewed without actually sending the messages as well. The generation of messages can be done even if the system has not been registered.

The system can also be unregistered from the ASR service, which will remove the system from being monitored and will disable all telemetry sent by the system.

Options The following options are supported:

`-e` *endpoint_URL*
: Sets the endpoint URL used for registration and all message telemetry. The default value is `transport.sun.com` sends all telemetry directly to Oracle service. A different URL can be used to support a local instance of an ASR Manager solution. The local ASR Manager can be used to aggregate telemetry from many hosts instances.

`-h` *host*[:*port*]
: Sets the HTTPS proxy host and optional port number to use for connecting to the internet. If a port is not specified, the default port value of `80` is used.

`-i`
Sets the HTTPS connection to be a direct internet connection and thus not use any proxy host. This option will clear any previously set HTTPS proxy information.

`-n`
Do a dry run of sending an event, which displays the message data to stdout that would have been sent.

`-u` *username*
When used with the `register` subcommand, this specifies the MOS user name to be associated with registrations of products on this system. When used with the `set-proxy` subcommand this specifies the HTTPS proxy user name used for message transport.

`-p` *password-file*
This should be a single-line file containing a password value. It can be immediately removed after running this command. When used with the `register` subcommand, this specifies the password associated with the MOS user name. When used with the `set-proxy` subcommand, this option specifies the password associated with the HTTPS proxy used to connect to the internet. If the `-p` option is not entered and the `-u` option was set, the user will be prompted for the password.

Sub-commands The `asradm` subcommands are described as follows.

`authenticate`
Authenticates MOS credentials with the Oracle ASR service and prints out `sysconfig` properties that can be used in conjunction with the automated installer to populate the `asr-notify` service so that it can automatically register the with the Oracle ASR service.

The command requires the `-u` user and `-p` option and also takes the optional endpoint argument to define an ASR Manager endpoint location.

If the `-n` option is supplied then no network connection will be made and the properties required to authenticate later will be printed.

`register`
Registers the system with MOS using the supplied authentication credentials. The MOS password will be used only for initial registration, to obtain a token used for all future telemetry and will not be stored anywhere on the system.

If the host system is behind a firewall, then the HTTP proxy settings must be set using the set-proxy command before registering the system.

Once registered, an ASR activation message will be sent that will request support for automated support call generation. Subsequently, an email will be sent to the registered user giving the status of the service request.

`list`
Shows the current authenticated MOS user name and the network connectivity information needed for HTTPS communication with MOS. If no registration has yet been done (by means of the `register` command), then the status of `Unregistered` is displayed.

`set-proxy`
Sets up the HTTPS connection information to be used for sending all ASR messages. Either a direct connection can be made or an HTTPS proxy can be defined.

`unregister`
Sends an ASR deactivation event and removes all configured registration information. No further telemetry will be sent and the Oracle ASR service will no longer generate any automated support calls.

`send`
Manually sends a specified ASR message to the Oracle ASR service or, with the dry run option (`-n`), display ASR messages.

Examples

EXAMPLE 1 Setup Internet Connection to Use an HTTPS Proxy

The following command will route all messages through an HTTPS proxy host `webproxy.example.com` on port 8080.

```
# asradm set-proxy -h webproxy.example.com:8080
```

EXAMPLE 2 Registering an MOS ID

The following command is used to interactively authenticate and register this system with the given MOS ID for use with ASR. Following this, you will be prompted for your support user name and password.

```
# asradm register
```

EXAMPLE 3 Authenticating Non-interactively

This is similar to `set–proxy` example, above. The difference is that the MOS user name and password are specified by means of the command line.

```
# asradm register -u joe.admin@example.com -p mypassword
```

EXAMPLE 4 Viewing Contents of Audit Message

The command below will display an audit message without sending the event. This will work even if the ASR service has not been registered.

```
# asradm send -n audit
```

Exit Status

The following exit values are returned:

`0`
Command completed with no errors.

1
Command failed to complete due to system error.

2
Command line usage is incorrect.

3
Connection configuration is not valid.

4
Authentication error.

5
Network connection error.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/fault-management/asr-notify
Interface Stability	Uncommitted

See Also svcs(1), asr-notify(1M), svcadm(1M), svccfg(1M), syslogd(1M), attributes(5), smf(5)

Name asr-notify – auto service request notification daemon for Fault Management events

Synopsis

```
/usr/lib/fm/notify/asr-notify [- ]
```

Description `asr-notify` is a daemon that subscribes to Fault Management life cycle events and produces HTTPS/XML notifications based on a set of notification preferences that are stored in the SMF service configuration repository.

The messages will be sent to the Oracle Auto Service Request (ASR) service offering once the service has been registered using the `asradm(1M)` command or by setting SMF auto registration properties. If the auto registration properties are set, the service will attempt to register on either startup or refresh. If registration fails, the service will go into maintenance mode. If registration is successful, the SMF auto registration properties will be deleted.

`asr-notify` is managed by the service management facility, `smf(5)`, under the service FMRI:

```
svc:/system/fm/asr-notify:default
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. The service's status can be queried using the `svcs(1)` command.

ASR notification preferences are set using `svccfg(1M)`.

Properties The following service properties can be set:

`config/`*rootdir*
: This is an `astring` property that defaults to `/`. When set, the specified root directory will be used for all pathnames evaluated by `asr-notify`.

`config/`*http_timeout*
: This is an integer property that sets the number of seconds to wait for HTTP connections before generating an error.

`config/`*endpoint*
: This is an `astring` property that sets the endpoint URL used for registration and all message telemetry. The endpoint URL can be set to a local ASR Manager that will aggregate telemetry from many hosts instances. If not set, then all telemetry will be sent directly to the Oracle Auto Service Request service.

`autoreg/`*user*
: This is an `astring` property that is used to set the My Oracle Support user name used for auto registration.

`autoreg/`*password*
: This is an `astring` property that is used to set the My Oracle Support password used for auto registration.

`autoreg/`*proxy-host*
: This is an `astring` property that is used to set the HTTP proxy *hostname*:*port* and needs to be set only if an HTTP proxy is used to connect to the internet.

`autoreg`/*proxy-user*
This is an `astring` property that is used to set the HTTP proxy user name and needs to be set only if the HTTP proxy requires credentials.

`autoreg`/*proxy-password*
This is an `astring` property that is used to set the HTTP proxy password and needs to be set only if the HTTP proxy requires credentials.

Messages The service will send the following types of messages:

- fault messages when an FMA fault is created
- daily heartbeat messages to indicate that the system is still active
- running and weekly audit messages to provide a full inventory of operational components

The contents of each message are an XML document that adheres to the ASR message XML schema. Each message starts with a common message header followed by event specific content.

ASR event header information:

```
XML Element     Description
------------    -----------
site-id         the serial number of the host sending
                the event
host-id         the hostname(1) of the system that the
                message is about
message-uuid    a unique id generated for each message
message-time    local time on the system that generated
                the message
system-id       the serial number of the system that the
                message is about
asset-id        the registered service tag for the system
product-id      the unique product identifier for the system
product-name    the name of the product for the system
event           element containing fault specific event
                information
```

The fault-specific content in the event will be contained within the primary-event-information XML element.

```
XML Element     Description
-----------     -----------
message-id      the knowledge article message ID
event-uuid      the FMA event UUID
event-time      the time that the FMA event was generated
severity        the severity of the FMA event
component       element containing list of suspects
summary         a short summary of the event
```

```
description     event description
payload         the raw FMA event contents
```

For each suspect FRU associated with a fault event, a hardware-component element will be included within the component.

```
XML Element             Description
-----------             -----------
name                    the component path name of the FRU
serial                  the FRU serial number
part                    the FRU part number
revision                the FRU revision level
additional-information  a list of additional properties
                        containing values such as FRU
                        manufacturer and FRU model
```

Some faults might be associated with a software issue instead of a hardware issue. In this case, the fault event will contain a software-module element within the component list.

```
XML Element             Description
-----------             -----------
name                    the name or FMRI of the software
                        module in error
description             a description of the software
                        module error
```

Heartbeat events do not contain any additional data and audit events contain a list of all hardware-component, software-package, and software-module elements available to the system.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/fault-management/asr-notify
Interface Stability	Uncommitted

See Also svcs(1), asradm(1M), svcadm(1M), svccfg(1M), syslogd(1M), attributes(5), smf(5)

Name atohexlabel – convert a human readable label to its internal text equivalent

Synopsis

```
/usr/sbin/atohexlabel [human-readable-sensitivity-label]

/usr/sbin/atohexlabel -c [human-readable-clearance]
```

Description `atohexlabel` converts a human readable label into an internal text representation that is safe for storing in a public object. If no option is supplied, the label is assumed to be a sensitivity label.

Internal conversions can later be parsed to their same value. This internal form is often hexadecimal. The converted label is written to the standard output file. If no human readable label is specified, the label is read from the standard input file. The expected use of this command is emergency repair of labels that are stored in internal databases.

Options

`-c` Identifies the human readable label as a clearance.

Exit Status The following exit values are returned:

`0` On success.

`1` On failure, and writes diagnostics to the standard error file.

Files

`/etc/security/tsol/label_encodings`
The label encodings file contains the classification names, words, constraints, and values for the defined labels of this system.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/trusted
Interface Stability	See below.

The command output is Committed for systems with the same `label_encodings` file. The command invocation is Committed for systems that implement the DIA MAC policy.

See Also `hextoalabel(1M)`, `label_to_str(3TSOL)`, `str_to_label(3TSOL)`, `label_encodings(4)`, `attributes(5)`

Trusted Extensions Configuration and Administration

Notes The functionality described on this manual page is available only if the system is configured with Trusted Extensions.

This file is part of the Defense Intelligence Agency (DIA) Mandatory Access Control (MAC) policy. This file might not be applicable to other MAC policies that might be developed for future releases of Solaris Trusted Extensions software.

Name audit – control the behavior of the audit service

Synopsis

```
audit -n | -s | -t | -v
```

Description The `audit` command is the system administrator's interface to start, terminate, and refresh the audit service, `auditd(1M)`. Refreshing the audit service rereads the service and plugin configuration.

Options

`-n` Notify the audit service `audit_binfile(5)` plugin to close the current audit file and open a new audit file in the current audit directory.

`audit_remote(5)` is notified to close the current open connection which inherently means that the audit remote server will close the related audit file. `audit_remote(5)` attempts to establish a new connection with the same host, thus open a new audit file.

`-s` Start (enable) the audit service if it is not running, or refresh the audit service, if it is currently running.

`-t` Terminate (disable) the audit service. The audit service will close out the active plugins, stop auditing and exit. Use `-s` to restart auditing.

`-v` Verify that at least one plugin is active or audit remote server is enabled. Verify attributes of plugins and audit remote server `ars(5)` configuration.

Diagnostics The `audit` command will exit with `0` upon success and a positive integer upon failure.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os
Interface Stability	Committed

See Also `auditconfig(1M)`, `auditd(1M)`, `ars(5)`, `attributes(5)`, `audit_binfile(5)`

See the section on Auditing in *Oracle Solaris 11.1 Administration: Security Services*.

Notes The `audit` command does not modify a process's preselection mask. Its functions are limited to performing control actions of the auditing subsystem. See `auditconfig(1M)` for configuration.

The `-s` option validates the audit plugin configuration. If it is not valid an error message is displayed and the audit service is not started or refreshed. The `-v` option may be used to validate the audit plugin configuration before using the `-s` option to start or refresh the audit service.

The `-s` option also checks state of the audit service. In case the audit service is found in the maintenance state (thus not able to be enabled or refreshed) the `audit` command returns with an appropriate message and exit code.

The `audit` command is available to administrators who have the Audit Control Rights Profile.

All options are valid in the global zone. In a non-global zone, if perzone policy is disabled and the audit remote server is not enabled, only the `-v` option is valid. See `auditconfig(1M)` for per-zone audit configuration.

Name auditconfig – configure auditing

Synopsis `auditconfig` *subcommand*...

Description `auditconfig` provides a command line interface to get and set kernel audit parameters.

Except for getting or setting the persistent audit service values, this functionality is available only if the Solaris Auditing feature has been enabled.

A zero (`0`) queue value indicates that the system default is in effect.

The setting of the `perzone` policy determines the scope of the audit setting controlled by `auditconfig`. If `perzone` is set, then the values reflect the local zone except as noted. Otherwise, the settings are for the entire system. Any restriction based on the `perzone` setting is noted for each option to which it applies.

A non-global zone administrator can set all audit policy options except `perzone` and `ahlt`. `perzone` and `ahlt` apply only to the global zone; setting these policies requires the privileges of a global zone administrator. `perzone` and `ahlt` are described under the `-setpolicy` option, below.

This command is available to administrators who have been granted the Audit Control Rights Profile.

Options The following option is supported:

`-t`
Display or set the values on the running system in addition to the persistent values of the audit service.

This option is available only for the subcommands that list it below.

Sub-commands `-aconf`
Set the configured non-attributable audit mask, `kmask`, to the configured non-attributable audit mask. For example:

```
# auditconfig -aconf
Configured non-attributable event mask.
```

`-audit` *event sorf retval string*
This command constructs an audit record for audit event *event* using the process's audit characteristics containing a text token *string*. The return token is constructed from the *sorf* (success/failure flag) and the *retval* (return value). The event is type `char*`, the *sorf* is 0/1 for success/failure, *retval* is an errno value, *string* is type `*char`. This command is useful for constructing an audit record with a shell script. An example of this option:

```
# auditconfig -audit AUE_ftpd 0 0 "test string"
#

audit record from audit trail:
```

```
header,76,2,ftp access,,Fri Dec 08 08:44:02 2000, + 669 msec
subject,abc,root,other,root,other,104449,102336,235 197121 elbow
text,test string
return,success,0
```

`-chkaconf`
Checks the configuration of the non-attributable events set in the kernel against the entries configured in the audit service (`-setnaflags`). If the active class mask of a kernel audit event does not match the configured class mask, a mismatch is reported.

`-chkconf`
Check the configuration of kernel audit event to class mappings. If the runtime class mask of a kernel audit event does not match the configured class mask, a mismatch is reported.

`-conf`
Configure kernel audit event to class mappings. Runtime class mappings are changed to match those in the audit event to class database file.

`-getasid`
Prints the audit session ID of the current process. For example:

```
# auditconfig -getasid
audit session id = 102336
```

`-getaudit`
Returns the audit characteristics of the current process.

```
# auditconfig -getaudit
audit id = abc(666)
process preselection mask = lo(0x1000,0x1000)
terminal id (maj,min,host) = 235,197121,elbow(172.146.89.77)
audit session id = 102336
```

`-getauid`
Prints the audit ID of the current process. For example:

```
# auditconfig -getauid
audit id = abc(666)
```

`-getcar`
Prints current active root location (anchored from root [or local zone root] at system boot). For example:

```
# auditconfig -getcar
current active root = /
```

`-getclass` *event*
Display the preselection mask associated with the specified kernel audit event. *event* is the kernel event number or event name.

`-getcond`
Display the kernel audit condition. The condition displayed is the literal string `auditing` meaning auditing is enabled and turned on (the kernel audit module is constructing and queuing audit records, audit daemon is running); `noaudit`, meaning auditing is enabled but turned off (the kernel audit module is not constructing and queuing audit records, audit daemon is not running); `disabled`, meaning that the audit module has not been enabled (the module has been excluded in `system(4)`). See `auditd(1M)` for further information.

`-getestate` *event*
For the specified event (string or event number), print out classes *event* has been assigned. For example:

```
# auditconfig -getestate 20
audit class mask for event AUE_REBOOT(20) = 0x800
# auditconfig -getestate AUE_RENAME
audit class mask for event AUE_RENAME(42) = 0x30
```

`-getflags`
Display the user default audit preselection flags.

`-getkaudit`
Get audit characteristics of the current zone. For example:

```
# auditconfig -getkaudit
audit id = unknown(-2)
process preselection mask = lo,na(0x1400,0x1400)
terminal id (maj,min,host) = 0,0,(0.0.0.0)
audit session id = 0
```

If the audit policy `perzone` is not set, the terminal id is that of the global zone. Otherwise, it is the terminal id of the local zone.

`-getkmask`
Get non-attributable pre-selection mask for the current zone. For example:

```
# auditconfig -getkmask
audit flags for non-attributable events = lo,na(0x1400,0x1400)
```

If the audit policy `perzone` is not set, the kernel mask is that of the global zone. Otherwise, it is that of the local zone.

`-getnaflags`
Display the non-attributable audit flags.

`-getpinfo` *pid*
Display the audit ID, preselection mask, terminal ID, and audit session ID for the specified process.

`-getplugin` [*name*]
Display information about the plugin name. If *name* is not specified, display all plugins.

[-t] -getpolicy
Display the kernel audit policy. The ahlt and perzone policies reflect the settings from the global zone. If perzone is set, all other policies reflect the local zone's settings. If perzone is not set, the policies are machine-wide.

-getremote [*server*|[*group* [*connection_group*]]]
Display the audit remote server-related information. If *server* option argument is used, only the common audit remote server configuration is displayed. If the option argument *group* is used, information about all configured connection groups is displayed. If, in addition to the group argument, the *connection_group* name is specified, information about only the respective connection group is displayed.

If no option arguments are used, information about common audit remote server configuration details and all connection groups are displayed.

-getcwd
Prints current working directory (anchored from zone root at system boot). For example:

```
# cd /usr/tmp
# auditconfig -getcwd
current working directory = /var/tmp
```

[-t] -getqbufsz
Get audit queue write buffer size. For example:

```
# auditconfig -getqbufsz
no configured audit queue size
audit queue buffer size (bytes) = 1024
```

[-t] -getqctrl
Get audit queue write buffer size, audit queue hiwater mark, audit queue lowater mark, audit queue prod interval (ticks).

```
# auditconfig -getqctrl
no configured audit queue lowater mark
no configured audit queue hiwater mark
no configured audit queue size
no configured audit queue delay
audit queue hiwater mark (records) = 100
audit queue lowater mark (records) = 10
audit queue buffer size (bytes) = 1024
audit queue delay (ticks) = 20

# auditconfig -setqbufsz 8192
# auditconfig -t -setqbufsz 12288
# auditconfig -setqdelay 20
# auditconfig -t -setqdelay 25
# auditconfig -getqctrl
no configured audit queue lowater mark
no configured audit queue hiwater mark
```

```
configured audit queue buffer size (bytes) = 8192
configured audit queue delay (ticks) = 20
active audit queue hiwater mark (records) =     100
active audit queue lowater mark (records) =     10
active audit queue buffer size (bytes) = 12288
active audit queue delay (ticks) = 25
```

[-t] -getqdelay
Get interval at which audit queue is prodded to start output. For example:

```
# auditconfig -getqdelay
no configured audit queue delay
audit queue delay (ticks) = 20
```

[-t] -getqhiwater
Get high water point in undelivered audit records when audit generation will block. For example:

```
# ./auditconfig -getqhiwater
no configured audit queue hiwater mark
audit queue hiwater mark (records) = 100
```

[-t] -getqlowater
Get low water point in undelivered audit records where blocked processes will resume. For example:

```
# auditconfig -getqlowater
no configured audit queue lowater mark
audit queue lowater mark (records) = 10
```

-getstat
Print current audit statistics information. For example:

```
# auditconfig -getstat
gen nona kern  aud  ctl  enq wrtn wblk rblk drop  tot  mem
910    1  725  184    0  910  910    0  231    0   88   48
```

See auditstat(1M) for a description of the headings in -getstat output.

-gettid
Print audit terminal ID for current process. For example:

```
# auditconfig -gettid
terminal id (maj,min,host) = 235,197121,elbow(172.146.89.77)
```

-lsevent
Display the currently configured (runtime) kernel and user level audit event information.

-lspolicy
Display the kernel audit policies with a description of each policy.

-setasid *session-ID* [*cmd*]
Execute shell or *cmd* with specified *session-ID*. For example:

```
# ./auditconfig -setasid 2000 /bin/ksh
#
# ./auditconfig -getpinfo 104485
audit id = abc(666)
process preselection mask = lo(0x1000,0x1000)
terminal id (maj,min,host) = 235,197121,elbow(172.146.89.77)
audit session id = 2000
```

`-setaudit` *audit-ID preselect_flags term-ID session-ID* [*cmd*]
Execute shell or *cmd* with the specified audit characteristics.

`-setauid` *audit-ID* [*cmd*]
Execute shell or *cmd* with the specified *audit–ID*.

`-setclass` *event audit_flag*[,*audit_flag* . . .]
Map the kernel event *event* to the classes specified by `audit_flag` list. *event* is an event number or name. An *audit_flag* is a character string representing an audit class. See `audit_flags(5)` for further information. If `perzone` is not set, this option is valid only in the global zone.

`-setflags` *audit_flags*
Set the default user audit preselection flags; see `audit_flags(5)`. The default preselection flags are combined with the user's specific audit flags to form the user's audit preselection mask.

`-setkaudit` *IP-address_type IP_address*
Set IP address of machine to specified values. *IP-address_type* is `ipv6` or `ipv4`.

If `perzone` is not set, this option is valid only in the global zone.

`-setkmask` *audit_flags*
Set non-attributable preselection flags of machine.

If `perzone` is not set, this option is valid only in the global zone.

`-setnaflags` *audit_flags*
Set the non-attributable audit flags; see `audit_flags(5)`. Non-attributable audit flags define which classes of events are to be audited when the action cannot be attributed to an authenticated user. Failed login is an example of an event that is non-attributable.

`-setpmask` *pid flags*
Set the preselection mask of the specified process. *flags* is the ASCII representation of the flags similar to that in `audit_flags(5)`.

If `perzone` is not set, this option is valid only in the global zone.

`-setplugin` *plugin_name* `active`|`inactive` [*attributes* [*qsize*]]
`-setplugin` *plugin_name* [`active`|`inactive`] *attributes* [*qsize*]
Configure the plugin *plugin_name* to be `active` or `inactive`. Optionally configure the attributes and number of unprocessed audit records to queue for the plugin. See the relevant audit plugin man pages and `auditd(1M)`.

[`-t`] `-setpolicy` [+|-]*policy_flag*[,*policy_flag* ...]
Set the kernel audit policy. A policy *policy_flag* is literal strings that denotes an audit policy. A prefix of + adds the policies specified to the current audit policies. A prefix of - removes the policies specified from the current audit policies. No policies can be set from a local zone unless the `perzone` policy is first set from the global zone. The following are the valid policy flag strings (`auditconfig -lspolicy` also lists the current valid audit policy flag strings):

`all`	Include all policies that apply to the current zone.
`ahlt`	Panic is called and the system dumps core if an asynchronous audit event occurs that cannot be delivered because the audit queue has reached the high-water mark or because there are insufficient resources to construct an audit record. By default, records are dropped and a count is kept of the number of dropped records.
`arge`	Include the `execv(2)` system call environment arguments to the audit record. This information is not included by default.
`argv`	Include the `execv(2)` system call parameter arguments to the audit record. This information is not included by default.
`cnt`	Do not suspend processes when audit resources are exhausted. Instead, drop audit records and keep a count of the number of records dropped. By default, process are suspended until audit resources become available.
`group`	Include the supplementary group token in audit records. By default, the group token is not included.
`none`	Include no policies. If used in other than the global zone, the `ahlt` and `perzone` policies are not changed.
`path`	Add secondary path tokens to audit record. These are typically the pathnames of dynamically linked shared libraries or command interpreters for shell scripts. By default, they are not included.
`perzone`	Maintain separate configuration, queues, and logs for each zone and execute a separate version of `auditd(1M)` for each zone.
`public`	Audit public files. By default, read-type operations are not audited for certain files which meet *public* characteristics: owned by root, readable by all, and not writable by all.
`trail`	Include the trailer token in every audit record. By default, the trailer token is not included.
`seq`	Include the sequence token as part of every audit record. By default, the sequence token is not included. The sequence token attaches a sequence number to every audit record.

`windata_down` Include in an audit record any downgraded data moved between windows. This policy is available only if the system is configured with Trusted Extensions. By default, this information is not included.

`windata_up` Include in an audit record any upgraded data moved between windows. This policy is available only if the system is configured with Trusted Extensions. By default, this information is not included.

`zonename` Include the `zonename` token as part of every audit record. By default, the `zonename` token is not included. The `zonename` token gives the name of the zone from which the audit record was generated.

`-setremote` *server* `active`|`inactive` [*attributes*]
`-setremote` *server* [`active`|`inactive`] *attributes*
Configure the main audit remote server switch to be active or inactive. If it is set to inactive, all configured connection groups are deemed inactive. Optionally configure the common audit remote server attributes. For more information, see `ars(5)`.
`-setremote group active`|`inactive` *group_name* [*attributes*]
`-setremote group` [`active`|`inactive`] *group_name attributes*
Configure the audit remote server connection group group_name to be active or inactive. Optionally configure the respective connection group attributes. For more information, see `ars(5)`.
`-setremote group create`|`destroy` *group_name*
Create or destroy the audit remote server connection group *group_name*. For more information, see `ars(5)`.

[`-t`] `-setqbufsz` *buffer_size*
Set the audit queue write buffer size (bytes). Zero (`0`), indicates reset to no configured value.

[`-t`] `-setqctrl` *hiwater lowater bufsz interval*
Set the audit queue write buffer size (bytes), hiwater audit record count, lowater audit record count, and wakeup interval (ticks). Valid within a local zone only if `perzone` is set. Zero (`0`), indicates reset to no configured value.

[`-t`] `-setqdelay` *interval*
Set the audit queue wakeup interval (ticks). This determines the interval at which the kernel pokes the audit queue, to write audit records to the audit trail. Valid within a local zone only if `perzone` is set. Zero (`0`), indicates reset to no configured value.

[`-t`] `-setqhiwater` *hiwater*
Set the number of undelivered audit records in the audit queue at which audit record generation blocks. Valid within a local zone only if `perzone` is set. Zero (`0`), indicates reset to no configured value.

[`-t`] `-setqlowater` *lowater*
Set the number of undelivered audit records in the audit queue at which blocked auditing processes unblock. Valid within a local zone only if `perzone` is set. Zero (`0`), indicates reset to no configured value.

`-setsmask` *asid flags*
Set the preselection mask of all processes with the specified audit session ID. Valid within a local zone only if `perzone` is set.

`-setstat`
Reset audit statistics counters. Valid within a local zone only if `perzone` is set.

`-setumask` *username*|*auid flags*
Set the preselection mask of all processes with the specified username or audit ID. Valid within a local zone only if `perzone` is set.

Examples

EXAMPLE 1 Using `auditconfig`

The following are examples of `auditconfig` commands.

```
#
# Map kernel audit event number 10 to the "fr" audit class.

# auditconfig -setclass 10 fr

#
# Turn on inclusion of exec arguments in exec audit records.

# auditconfig -setpolicy +argv
```

EXAMPLE 2 Setting Only the Number of Unprocessed Audit Records

The following sequence of commands sets only the number of unprocessed audit records to queue for the `audit_binfile` plugin.

```
# See if audit_binfile is active.
% auditconfig -getplugin audit_binfile

# Set to queue 20 unprocessed audit records.
#
% auditconfig -setplugin audit_binfile "" 20
```

EXAMPLE 3 Resetting Queue Control Parameters

The following commands reset active and configured queue control parameters.

```
# Get the audit remote server configuration
auditconfig -getremote

# Change an audit remote server attribute
auditconfig -setremote server \
"listen_address=10.0.0.1,max_startups=10:30:60"

# Create an audit remote server (wild card) connection group
auditconfig -setremote group create egg_farm
```

EXAMPLE 3 Resetting Queue Control Parameters *(Continued)*

```
# Get a specific audit remote server connection group information
auditconfig -getremote group egg_farm

# Set a connection group attribute, activate the connection group
auditconfig -setremote group active egg_farm \
"hosts=tipo.cz.oracle.com,binfile_dir=/var/audit/ARS"
```

EXAMPLE 4 Configuring an Audit Remote Server

The following command configure an audit remote server.

```
# Get the audit remote server configuration
auditconfig -getremote

# Change an audit remote server attribute
auditconfig -setremote server \
"listen_address=10.0.0.1,max_startups=10:30:60"

# Create an audit remote server (wild card) connection group
auditconfig -setremote group create egg_farm

# Get a specific audit remote server connection group information
auditconfig -getremote group egg_farm

# Set a connection group attribute, activate the connection group
auditconfig -setremote group active egg_farm \
"hosts=tipo.cz.oracle.com,binfile_dir=/var/audit/ARS"
```

Exit Status

`0` Successful completion.

`1` An error occurred.

Files

`/etc/security/audit_event` Stores event definitions used in the audit system.

`/etc/security/audit_class` Stores class definitions used in the audit system.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os
Interface Stability	Committed

See Also

`audit(1M)`, `auditd(1M)`, `auditstat(1M)`, `praudit(1M)`, `execv(2)`, `audit_class(4)`, `audit_event(4)`, `system(4)`, `ars(5)`, `attributes(5)`, `audit_binfile(5)`, `audit_flags(5)`, `audit_remote(5)`, `audit_syslog(5)`

See the section on Auditing in *Oracle Solaris 11.1 Administration: Security Services*.

Notes If plugin output is selected using the `-setplugin` option, the behavior of the system with respect to the `-setpolicy +cnt` and the `-setqhiwater` options is modified slightly. If `-setpolicy +cnt` is set, data will continue to be sent to the selected plugin, even though output of the `audit_binary` plugin is stopped, pending the freeing of disk space. If `-setpolicy –cnt` is used, the blocking behavior is as described under `SUBCOMMANDS`, above. The queue high water mark value is used within `auditd` as the upper bound for its queue limits unless overridden by means of the `qsize` attribute, as described in the explanation of the `-setplugin` option, above.

The `auditconfig` options that modify or display process-based information are not affected by the `perzone` policy. Those that modify system audit data such as the terminal id and audit queue parameters are valid only in the global zone, unless the `perzone` policy is set. The display of a system audit reflects the local zone if `perzone` is set. Otherwise, it reflects the settings of the global zone.

The change to plugins (`-setplugin`) and audit remote server (`-setremote`) settings do not take effect (such as becoming active or inactive, or changing the respective attributes) until the audit service is refreshed. Use `audit(1M)` to refresh the audit service.

Name auditd – audit service daemon

Synopsis `/usr/sbin/auditd`

Description The audit service daemon, `auditd`, manages audit data generated either locally (see `audit_binfile(5)`, `audit_syslog(5)` and `audit_remote(5)`) or remotely (see "Audit Remote Server" below). When auditing is enabled, `auditd` reads its configuration to do the following:

- Configure audit policy.
- Configure the audit queue control parameters.
- Configure the event-to-class mappings.
- Set the default audit masks.
- If local auditing is enabled (see "Local Auditing" below), load one or more plugins.

  Solaris provides three plugins. `audit_binfile(5)` writes binary audit data to a file. `audit_remote(5)` sends binary audit data to an authenticated server with privacy and integrity protection. `audit_syslog(5)` sends text summaries of audit records to the `syslog` daemon.
- Read audit data from the kernel and pass that data to each of the active plugins.
- Execute the `audit_warn(1M)` script to warn of various conditions.
- If remote auditing (`ars(5)`) is enabled, process requests and store the remotely generated audit data.

`audit(1M)` is used to control the audit service. It can cause `auditd` to:

- Close a connection to a remote audit server thus causing it to close its respective audit file.
- Start and refresh the service based on the current properties.
- Close the audit trail and disable local auditing and remote audit service.

`auditconfig(1M)` is used to configure the audit service. It can configure the active and permanent:

- audit policy
- audit queue control parameters
- default audit masks
- plugins to be loaded
- plugin attributes
- audit remote server state, attributes, and connection groups

Local Auditing The collecting of audit records that are generated on the local system. The records can be generated in the global zone or in non-global zones, or both.

Remote Auditing The Audit Remote Server, ARS, that receives and stores audit records from a system that is being audited and is configured with an active `audit_remote` plugin. To distinguish an audited system from an ARS, the audited system can be termed the locally audited system.

Auditing Conditions

The audit service daemon enables local auditing in case at least one audit daemon plugin is configured as active.

The Audit Remote Server functionality is enabled, if the server is not configured as inactive (see the `-setremote` server option in `auditconfig(1M)`) and at least one connection group is active. See Audit Remote Server section for more information.

Local auditing and the Audit Remote Server can be configured independently.

Audit Remote Server

The Audit Remote Server, ARS, is an integral part of `auditd`. It makes a counterpart to the `audit_remote(5)` plugin. Data sent by the plugin can be captured, processed, and stored by the server according to its configuration.

ARS is delivered as a disabled Solaris audit component. It is necessary to configure it before it can be used to process a remote audit trail. ARS configuration is twofold: first, the underlying security mechanisms used for secure audit data transport has to be configured (see `audit_remote(5)`); second, the audit subsystem has to be properly configured.

To observe and configure the ARS, use the `auditconfig(1M)` `-setremote` and `-getremote` options. The configuration is divided to the configuration of `server` and `group`. The `server` configuration allows for changing common ARS parameters, while the `group` keyword allows configuration of connection groups, the sets of hosts sharing the same local storage parameters.

Server Configuration Attributes

`listen_address`
: Address the server listens on. Empty `listen_address` attribute defaults to listen on all local addresses.

`listen_port`
: The local listening port; 0 defaults to 16162. Port associated with the `solaris-audit` Internet service name. See `services(4)`.

`login_grace_time`
: The server disconnects after login grace time (in seconds) if the connection has not been successfully established. `0` defaults to no limit.

`max_startups`
: Number of concurrent unauthenticated connections to the server at which the server starts refusing new connections. Note that the value might be specified in *begin*:*rate*:*full* format to allow random early drop mode, for example `10:30:60`. That means that ARS would refuse connection attempts with a probability of *rate*/100 (30% in our example) if there are currently 10 (from the `start` field) unauthenticated connections. The probability increases linearly and all connection attempts are refused if the number of unauthenticated connections reaches `full` (60 in our example).

Group Configuration Attributes

`binfile_dir`, `binfile_fsize`, `binfile_minfree`
Attributes follow the respective `p_*` attributes defined in `audit_binfile(5)`, in short:

`binfile_dir`
Directory for storing per host audit data.

`binfile_fsize`
The maximum size of each of the stored audit trail files; `0` defaults to no limit.

`binfile_minfree`
The minimum free space on file system with binfile_dir before the `audit_binfile(5)` lets administrator know by means of `audit_warn(1M)`; `0` defaults to no limit.

`hosts`
Defines the hosts in the given connection group allowed to send audit data to server. Note that a comma is a delimiter in case of multiple host entries. If `hosts` is empty, such connection group is called a wild card connection group. If a new connection cannot be classified to any other (non-wild card) connection group and there is an active wild card connection group configured, the new connection is classified to that connection group. Only one active wild card connection group can be configured.

For a configuration example, see "Examples".

For comprehensive configuration description and examples, see the appropriate chapter in the *Oracle Solaris 11.1 Administration: Security Services*.

Audit Record Queue

The maximum number of records to queue for audit data sent to the plugin is specified by the `qsize` parameter specified for the plugin. If omitted, the current `hiwater` mark is used. See the `-getqctrl` option in `auditconfig(1M)`. When this maximum is reached, `auditd` will either block processes or discard data, depending on the `cnt` audit policy as described in `auditconfig(1M)`.

Auditing System Warnings

The audit service daemon and audit plugins invoke the script `audit_warn(1M)` under certain conditions. See `audit_warn(1M)` for more information.

Examples

EXAMPLE 1 Audit Remote Server Configuration

The following example describes steps to configure audit remote server to listen on a specific address. One wild card and one non-wild card connection group will be created. The non-wild card connection group configuration will address remote audit data from `tic.cz.example.com` and `tac.us.example.com`. The trail will be stored in `/var/audit/remote`.

```
# Print the current audit remote server configuration.
# Both server and connection groups (if any) is displayed.

# auditconfig -getremote

# Set address the audit remote server will listen on.
```

EXAMPLE 1 Audit Remote Server Configuration *(Continued)*

```
# auditconfig -setremote server "listen_address=192.168.0.1"

# Create two connection groups. Note that by default the
# connection group is created with no hosts specified
# (wild card connection group).

# auditconfig -setremote group create clockhouse
# auditconfig -setremote group create sink

# Add hosts to the connection group (convert the wild card
# connection group no non-wild card one). Set the storage
# directory and activate the connection group.

# auditconfig -setremote group active clockhouse \
# "hosts=tic.cz.example.com,tac.us.example.com,\
# binfile_dir=/var/audit/remote"

# Activate the wild card connection group.

# auditconfig -setremote group active sink

# Verify the audit remote server configuration.

# auditconfig -getremote

# Start or refresh the audit service.

# audit -s
```

Files

- etc/security/audit/audit_class
- etc/security/audit/audit_event

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os
Interface Stability	Committed

See Also audit(1M), audit_warn(1M), auditconfig(1M), praudit(1M), audit_class(4), audit_class(4), audit_event(4), services(4), ars(5), attributes(5), audit_binfile(5), audit_flags(5), audit_remote(5), audit_syslog(5), smf(5)

See the section on Auditing in *Oracle Solaris 11.1 Administration: Security Services*.

Notes `auditd` is loaded in the global zone at boot time if auditing is enabled.

If the audit policy `perzone` is set, `auditd` runs in each zone, starting automatically when the local zone boots. If a zone is running when the `perzone` policy is set, auditing must be started manually in local zones. It is not necessary to reboot the system or the local zone to start auditing in a local zone. `auditd` can be started with `audit -s` and will start automatically with future boots of the zone.

When `auditd` runs in a local zone, the configuration is taken from the local zone's `smf(5)` repository and the `/etc/security` directory's files: `audit_class`, `user_attr`, and `audit_event`.

Configuration changes do not affect audit sessions that are currently running, as the changes do not modify a process's preselection mask. To change the preselection mask on a running process, use the `–setpmask` option of the `auditconfig` command (see `auditconfig(1M)`). If the user logs out and logs back in, the new configuration changes will be reflected in the next audit session.

The audit service FMRI is `svc:/system/auditd:default`.

Name auditrecord – display Solaris audit record formats

Synopsis

```
/usr/sbin/auditrecord [-d] [ [-a] | [-e string] | [-c class] |
      [-i id] | [-p programname] | [-s systemcall] | [-h]]
```

Description The `auditrecord` utility displays the event ID, audit class and selection mask, and record format for audit record event types defined in `audit_event(4)`. You can use `auditrecord` to generate a list of all audit record formats, or to select audit record formats based on event class, event name, generating program name, system call name, or event ID.

There are two output formats. The default format is intended for display in a terminal window; the optional HTML format is intended for viewing with a web browser.

Tokens contained in square brackets (`[ ]`) are optional and might not be present in every record.

Options The following options are supported:

`-a`
List all audit records.

`-c` *class*
List all audit records selected by *class*. *class* is one of the two-character class codes from the file `/etc/security/audit_class`.

`-d`
Debug mode. Display number of audit records that are defined in `audit_event`, the number of classes defined in `audit_class`, any mismatches between the two files, and report which defined events do not have format information available to `auditrecord`.

`-e` *string*
List all audit records for which the event ID label contains the string *string*. The match is case insensitive.

`-h`
Generate the output in HTML format.

`-i` *id*
List the audit records having the numeric event ID *id*.

`-p` *programname*
List all audit records generated by the program *programname*, for example, audit records generated by a user-space program.

`-s` *systemcall*
List all audit records generated by the system call *systemcall*, for example, audit records generated by a system call.

The `-p` and `-s` options are different names for the same thing and are mutually exclusive. The `-a` option is ignored if any of `-c`, `-e`, `-i`, `-p`, or `-s` are given. Combinations of `-c`, `-e`, `-i`, and either `-p` or `-s` are ANDed together.

Examples

EXAMPLE 1 Displaying an Audit Record with a Specified Event ID

The following example shows how to display the contents of a specified audit record.

```
% auditrecord -i 6152
  terminal login
  program     /usr/sbin/login      see login(1)
              /usr/dt/bin/dtlogin  See dtlogin
  event ID    6152                 AUE_login
  class       lo                   (0x00001000)
      header
      subject
      [text]                       error message
      return
```

EXAMPLE 2 Displaying an Audit Record with an Event ID Label that Contains a Specified String

The following example shows how to display the contents of a audit record with an event ID label that contains the string `login`.

```
# auditrecord -e login
terminal login
  program     /usr/sbin/login      see login(1)
              /usr/dt/bin/dtlogin  See dtlogin
  event ID    6152                 AUE_login
  class       lo                   (0x00001000)
      header
      subject
      [text]                       error message
      return

rlogin
  program     /usr/sbin/login      see login(1) - rlogin
  event ID    6155                 AUE_rlogin
  class       lo                   (0x00001000)
      header
      subject
      [text]                       error message
      return
```

Exit Status

`0`
: Successful operation

`non-zero`
: Error

Files

`/etc/security/audit_class`
: Provides the list of valid classes and the associated audit mask.

`/etc/security/audit_event`
Provides the numeric event ID, the literal event name, and the name of the associated system call or program.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os
CSI	Enabled
Interface Stability	Uncommitted

See Also `auditconfig(1M)`, `praudit(1M)`, `audit.log(4)`, `audit_class(4)`, `audit_event(4)`, `attributes(5)`

See the section on Auditing in *Oracle Solaris 11.1 Administration: Security Services*.

Diagnostics If unable to read either of its input files or to write its output file, `auditrecord` shows the name of the file on which it failed and exits with a non-zero return.

If no options are provided, if an invalid option is provided, or if both `-s` and `-p` are provided, an error message is displayed and `auditrecord` displays a usage message then exits with a non-zero return.

Notes This command was formerly known as `bsmrecord`.

If `/etc/security/audit_event` has been modified to add user-defined audit events, `auditrecord` displays the record format as `undefined`.

The audit records displayed by `auditrecord` are the core of the record that can be produced. Various audit policies and optional tokens, such as those shown below, might also be present.

The following is a list of `praudit(1M)` token names with their descriptions.

`group`
Present if the `group` audit policy is set.

`sensitivity label`
Present when Trusted Extensions is enabled and represents the label of the subject or object with which it is associated. The `mandatory_label` token is noted in the basic audit record where a label is explicitly part of the record.

`sequence`
Present when the `seq` audit policy is set.

`trailer`
Present when the `trail` audit policy is set.

`zone`

The name of the zone generating the record when the `zonename` audit policy is set. The `zonename` token is noted in the basic audit record where a zone name is explicitly part of the record.

Name auditreduce – merge and select audit records from audit trail files

Synopsis `auditreduce [`*options*`] [`*audit-trail-file*`]...`

Description `auditreduce` allows you to select or merge records from audit trail files. Audit files can be from one or more machines.

The merge function merges together audit records from one or more input audit trail files into a single output file. The records in an audit trail file are assumed to be sorted in chronological order (oldest first) and this order is maintained by `auditreduce` in the output file.

Unless instructed otherwise, `auditreduce` will merge the entire audit trail, which consists of all the audit trail files in the directory structure *audit_root_dir*`/*` (see `audit.log(4)` for details of the structure of the audit root). Unless specified with the `-R` or `-S` option, *audit_root_dir* defaults to `/var/audit`. By using the file selection options it is possible to select some subset of these files, or files from another directory, or files named explicitly on the command line.

The select function allows audit records to be selected on the basis of numerous criteria relating to the record's content (see `audit.log(4)` for details of record content). A record must meet all of the *record-selection-option* criteria to be selected.

Audit Trail Filename Format

Any audit trail file not named on the command line must conform to the audit trail filename format. Files produced by the audit system already have this format. Output file names produced by `auditreduce` are in this format. It is:

start-time`.` *end-time*`.` *suffix*

where *start-time* is the 14-character timestamp of when the file was opened, *end-time* is the 14-character timestamp of when the file was closed, and *suffix* is the name of the machine which generated the audit trail file, or some other meaningful suffix (for example, `all`, if the file contains a combined group of records from many machines). The *end-time* can be the literal string `not_terminated`, to indicate that the file is still being written to by the audit system. Timestamps are of the form *yyyymmddhhmmss* (year, month, day, hour, minute, second). The timestamps are in Coordinated Universal Time (UTC).

Options

File Selection Options

The file selection options indicate which files are to be processed and certain types of special treatment.

`-A`

All of the records from the input files will be selected regardless of their timestamp. This option effectively disables the `-a`, `-b`, and `-d` options. This is useful in preventing the loss of records if the `-D` option is used to delete the input files after they are processed. Note, however, that if a record is *not* selected due to another option, then `-A` will not override that.

`-C`
Only process complete files. Files whose filename *end-time* timestamp is `not_terminated` are not processed (such a file is currently being written to by the audit system). This is useful in preventing the loss of records if `-D` is used to delete the input files after they are processed. It does not apply to files specified on the command line.

`-D` *suffix*
Delete input files after they are read if the entire run is successful. If `auditreduce` detects an error while reading a file, then that file is not deleted. If `-D` is specified, `-A`, `-C` and `-O` are also implied. *suffix* is given to the `-O` option. This helps prevent the loss of audit records by ensuring that all of the records are written, only complete files are processed, and the records are written to a file before being deleted. Note that if both `-D` and `-O` are specified in the command line, the order of specification is significant. The *suffix* associated with the latter specification is in effect.

`-M` *machine*
Allows selection of records from files with *machine* as the filename suffix. If `-M` is not specified, all files are processed regardless of suffix. `-M` can also be used to allow selection of records from files that contain combined records from many machines and have a common suffix (such as `all`).

`-N`
Select objects in *new mode*.This flag is off by default, thus retaining backward compatibility. In the existing, *old mode*, specifying the `-e`, `-f`, `-g`, `-r`, or `-u` flags would select not only actions taken with those IDs, but also certain objects owned by those IDs. When running in *new mode*, only actions are selected. In order to select objects, the `-o` option must be used.

`-O` *suffix*
Direct output stream to a file in the current `audit_root_dir` with the indicated suffix. *suffix* can alternatively contain a full pathname, in which case the last component is taken as the suffix, ahead of which the timestamps will be placed, ahead of which the remainder of the pathname will be placed. If the `-O` option is not specified, the output is sent to the standard output. When `auditreduce` places timestamps in the filename, it uses the times of the first and last records in the merge as the *start-time* and *end-time*.

`-Q`
Quiet. Suppress notification about errors with input files.

`-R` *pathname*
Specify the pathname of an alternate audit root directory *audit_root_dir* to be *pathname*. Therefore, rather than using `/var/audit` by default, *pathname*`/*` will be examined instead.

Note – The root file system of any non-global zones must not be referenced with the `-R` option. Doing so might damage the global zone's file system, might compromise the security of the global zone, and might damage the non-global zone's file system. See `zones(5)`.

`-S` *specific_directory*
This option causes `auditreduce` to read audit trail files from a specific location (*specific_directory*). *specific_directory* is normally interpreted as the name of a subdirectory of the audit root, therefore `auditreduce` will look in *audit_root_dir*/*specific_directory* for the audit trail files. But if *specific_directory* contains any backslash characters (`/`), it is the name of a directory not necessarily contained in the audit root. In this case, *specific_directory* will be consulted. This option allows archived files to be manipulated easily, without requiring that they be physically located in a directory structure like that of `/var/audit`.

`-V`
Verbose. Display the name of each file as it is opened, and how many records total were written to the output stream.

Record Selection Options

The record selection options listed below are used to indicate which records are written to the output file produced by `auditreduce`.

Multiple arguments of the same type are not permitted.

`-a` *date-time*
Select records that occurred at or after *date-time*. The *date-time* argument is described under `Option Arguments`, below. *date-time* is in local time. The `-a` and `-b` options can be used together to form a range.

`-b` *date-time*
Select records that occurred before *date-time*.

`-c` *audit-classes*
Select records by audit class. Records with events that are mapped to the audit classes specified by *audit-classes* are selected. Audit class names are defined in `audit_class(4)`. The *audit-classes* can be a comma separated list of `audit` *flags* like those described in `audit_flags(5)`. Using the `audit` *flags,* one can select records based upon success and failure criteria.

`-d` *date-time*
Select records that occurred on a specific day (a 24-hour period beginning at 00:00:00 of the day specified and ending at 23:59:59). The day specified is in local time. The time portion of the argument, if supplied, is ignored. Any records with timestamps during that day are selected. If any hours, minutes, or seconds are given in *time,* they are ignored. `-d` can not be used with `-a` or `-b`.

`-e` *effective-user*
Select records with the specified *effective-user*.

`-f` *effective-group*
Select records with the specified *effective-group*.

`-g` *real-group*
Select records with the specified *real-group*.

`-j` *subject-ID*
Select records with the specified *subject-ID* where *subject-ID* is a process ID.

`-l` *label*
Select records with the specified label (or label range), as explained under "Option Arguments," below. This option is available only if the system is configured with Trusted Extensions.

`-m` *event*
Select records with the indicated *event*. The *event* is the literal string or the *event* number.

`-o` *object_type=objectID_value*
Select records by object type. A match occurs when the record contains the information describing the specified *object_type* and the object ID equals the value specified by *objectID_value*. The allowable object types and values are as follows:

`auth=`*authorization*
Select records containing information about used authorization. A period at the end of the authorization name means the authorization is a wild card; records with more specific used authorizations objects are selected.

`file=`*pathname*
Select records containing file system objects with the specified pathname, where pathname is a comma separated list of regular expressions. If a regular expression is preceded by a tilde (~), files matching the expression are excluded from the output. For example, the option `file=~/usr/openwin,/usr,/etc` would select all files in `/usr` or `/etc` except those in `/usr/openwin`. The order of the regular expressions is important because auditreduce processes them from left to right, and stops when a file is known to be either selected or excluded. Thus the option `file=` `/usr`, `/etc`, `~/usr/openwin` would select all files in `/usr` and all files in `/etc`. Files in `/usr/openwin` are not excluded because the regular expression `/usr` is matched first. Care should be given in surrounding the *pathname* with quotes so as to prevent the shell from expanding any tildes.

`filegroup=`*group*
Select records containing file system objects with *group* as the owning group.

`fileowner=`*user*
Select records containing file system objects with *user* as the owning user.

`fmri=`*service_instance*
Select records containing fault management resource identifier (FMRI) objects with the specified *service instance*. See `smf(5)`.

`msgqid=`*ID*
Select records containing message queue objects with the specified *ID* where *ID* is a message queue ID.

`msgqgroup=`*group*
Select records containing message queue objects with *group* as the owning or creating group.

`msgqowner=`*user*
Select records containing message queue objects with *user* as the owning or creating user.

`pid=`*ID*
Select records containing process objects with the specified *ID* where *ID* is a process ID. Process are objects when they are receivers of signals.

`procgroup=`*group*
Select records containing process objects with *group* as the real or effective group.

`procowner=`*user*
Select records containing process objects with *user* as the real or effective user.

`semid=`*ID*
Select records containing semaphore objects with the specified *ID* where *ID* is a semaphore ID.

`semgroup=`*group*
Select records containing semaphore objects with *group* as the owning or creating group.

`semowner=`*user*
Select records containing semaphore objects with *user* as the owning or creating user.

`shmid=`*ID*
Select records containing shared memory objects with the specified *ID* where *ID* is a shared memory ID.

`shmgroup=`*group*
Select records containing shared memory objects with *group* as the owning or creating group.

`shmowner=`*user*
Select records containing shared memory objects with *user* as the owning or creating user.

`sock=`*port_number*|*machine*
Select records containing socket objects with the specified *port_number* or the specified *machine* where *machine* is a machine name as defined in `hosts(4)`.

`user=`*user_name*
Select records containing the user object whose name is specified. User objects are generally specified for administrative actions on a user.

`-r` *real-user*
Select records with the specified *real-user*.

`-s` *session-id*
Select audit records with the specified *session-id*.

`-u` *audit-user*
Select records with the specified *audit-user*.

`-z` *zone-name*
Select records from the specified zone name. The zone name selection is case-sensitive.

When one or more *filename* arguments appear on the command line, only the named files are processed. Files specified in this way need not conform to the audit trail filename format. However, `-M`, `-S`, and `-R` must not be used when processing named files. If the *filename* is "–" then the input is taken from the standard input.

Option Arguments

audit-trail-file
An audit trail file as defined in `audit.log(4)`. An audit trail file not named on the command line must conform to the audit trail file name format. Audit trail files produced as output of `auditreduce` are in this format as well. The format is:

start-time . *end-time* . *suffix*

start-time is the 14 character time stamp denoting when the file was opened. *end-time* is the 14 character time stamp denoting when the file was closed. *end-time* can also be the literal string `not_terminated`, indicating the file is still be written to by the audit daemon or the file was not closed properly (a system crash or abrupt halt occurred). *suffix* is the name of the machine that generated the audit trail file (or some other meaningful suffix; for example, `all` would be a good suffix if the audit trail file contains a combined group of records from many machines).

date-time
The *date-time* argument to `-a`, `-b`, and `-d` can be of two forms: An absolute *date-time* takes the form:

yyyymmdd [*hh* [*mm* [*ss*]]]

where *yyyy* specifies a year (with 1970 as the earliest value), *mm* is the month (01-12), `dd` is the day (01-31), *hh* is the hour (00-23), *mm* is the minute (00-59), and *ss* is the second (00-59). The default is 00 for *hh*, *mm* and *ss*.

An offset can be specified as: +*n* `d`|`h`|`m`| `s` where *n* is a number of units, and the tags `d`, `h`, `m`, and `s` stand for days, hours, minutes and seconds, respectively. An offset is relative to the starting time. Thus, this form can only be used with the `-b` option.

event
The literal string or ordinal event number as found in `audit_event(4)`. If *event* is not found in the `audit_event` file it is considered invalid.

group
The literal string or ordinal group ID number as found in `group(4)`. If *group* is not found in the `group` file it is considered invalid. *group* can be negative.

label
The literal string representation of a MAC label or a range of two valid MAC labels. To specify a range, use `x;y` where `x` and `y` are valid MAC labels. Only those records that are fully bounded by `x` and `y` will be selected. If `x` or `y` is omitted, the default uses `ADMIN_LOW` or `ADMIN_HIGH` respectively. Notice that quotes must be used when specifying a range.

pathname
A regular expression describing a pathname.

user
The literal username or ordinal user ID number as found in `passwd(4)`. If the username is not found in the `passwd` file it is considered invalid. *user* can be negative.

Examples

EXAMPLE 1 Using `auditreduce`

`praudit(1M)` is available to display audit records in a human-readable form.

This will display the entire audit trail in a human-readable form:

```
% auditreduce | praudit
```

If all the audit trail files are being combined into one large file, then deleting the original files could be desirable to prevent the records from appearing twice:

```
% auditreduce -V -D /var/audit/combined/all
```

This displays what user `milner` did on April 13, 1988. The output is displayed in a human-readable form to the standard output:

```
% auditreduce -d 19880413 -u milner | praudit
```

The above example might produce a large volume of data if `milner` has been busy. Perhaps looking at only login and logout times would be simpler. The `-c` option will select records from a specified class:

```
% auditreduce -d 19880413 -u milner -c lo | praudit
```

To see `milner`'s login/logout activity for April 13, 14, and 15, the following is used. The results are saved to a file in the current working directory. Notice that the name of the output file will have `milnerlo` as the *suffix*, with the appropriate timestamp prefixes. Notice also that the long form of the name is used for the `-c` option:

```
% auditreduce -a 19880413 -b +3d -u milner -c login_logout -O milnerlo
```

To follow `milner`'s movement about the file system on April 13, 14, and 15 the `chdir` record types could be viewed. Notice that in order to get the same time range as the above example we needed to specify the `-b` time as the day *after* our range. This is because `19880416` defaults to midnight of that day, and records before that fall on `0415`, the end-day of the range.

```
% auditreduce -a 19880413 -b 19880416 -u milner -m AUE_CHDIR | praudit
```

EXAMPLE 1 Using `auditreduce` *(Continued)*

In this example, the audit records are being collected in summary form (the login/logout records only). The records are being written to a summary file in a different directory than the normal audit root to prevent the selected records from existing twice in the audit root.

```
% auditreduce -d 19880330 -c lo -O /var/audit/audit_summary/logins
```

If activity for user ID 9944 has been observed, but that user is not known to the system administrator, then the command in the following example searches the entire audit trail for any records generated by that user. `auditreduce` queries the system about the current validity of ID 9944 and displays a warning message if it is not currently active:

```
% auditreduce -O /var/audit/audit_suspect/user9944 -u 9944
```

To get an audit log of only the global zone:

```
% auditreduce -z global
```

Files `/var/audit/*` default location of audit trails, when stored

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os
Interface Stability	See below.

The command invocation is Stable. The binary file format is Stable. The binary file contents is Unstable.

See Also `praudit(1M)`, `audit.log(4)`, `audit_class(4)`, `group(4)`, `hosts(4)`, `passwd(4)`, `attributes(5)`, `audit_flags(5)`, `smf(5)`

See the section on Auditing in *Oracle Solaris 11.1 Administration: Security Services*.

Diagnostics `auditreduce` displays error messages if there are command line errors and then exits. If there are fatal errors during the run, `auditreduce` displays an explanatory message and exits. In this case, the output file might be in an inconsistent state (no trailer or partially written record) and `auditreduce` displays a warning message before exiting. Successful invocation returns `0` and unsuccessful invocation returns `1`.

Since `auditreduce` might be processing a large number of input files, it is possible that the machine-wide limit on open files will be exceeded. If this happens, `auditreduce` displays a message to that effect, give information on how many file there are, and exit.

If `auditreduce` displays a record's timestamp in a diagnostic message, that time is in local time. However, when filenames are displayed, their timestamps are in UTC.

Bugs Conjunction, disjunction, negation, and grouping of record selection options should be allowed.

Notes The -z option should be used only if the audit policy `zonename` is set. If there is no zonename token, then no records will be selected.

Name auditstat – display kernel audit statistics

Synopsis

```
auditstat [-c count] [-h numlines] [-i interval] [-n]
     [-T u | d ] [-v]
```

Description `auditstat` displays kernel audit statistics. The fields displayed are as follows:

`aud` The total number of audit records processed by the userland audit.

`ctl` This field is obsolete.

`drop` The total number of audit records that have been dropped. Records are dropped according to the kernel audit policy. See `auditconfig(1M)`, `AUDIT_CNT` policy for details.

`enq` The total number of audit records put on the kernel audit queue.

`gen` The total number of audit records that have been constructed (not the number written).

`kern` The total number of audit records produced by user processes (as a result of system calls).

`mem` The total number of Kbytes of memory currently in use by the kernel audit module.

`nona` The total number of non-attributable audit records that have been constructed. These are audit records that are not attributable to any particular user.

`rblk` The total number of times that the audit queue has blocked waiting to process audit data.

`tot` The total number of Kbytes of audit data written to the audit trail.

`wblk` The total number of times that user processes blocked on the audit queue at the high water mark.

`wrtn` The total number of audit records written. The difference between `enq` and `wrtn` is the number of outstanding audit records on the audit queue that have not been written.

Options

`-c` *count* Display the statistics a total of *count* times. If *count* is equal to zero, statistics are displayed indefinitely. A time interval must be specified.

`-h` *numlines* Display a header for every *numlines* of statistics printed. The default is to display the header every 20 lines. If *numlines* is equal to zero, the header is never displayed.

`-i` *interval* Display the statistics every *interval* where *interval* is the number of seconds to sleep between each collection.

`-n` Display the number of kernel audit events currently configured.

`-T u | d` Display a time stamp.

Specify `u` for a printed representation of the internal representation of time. See `time(2)`. Specify `d` for standard date format. See `date(1)`.

`-v` Display the version number of the kernel audit module software.

Exit Status `auditstat` returns `0` upon success and `1` upon failure.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also `auditconfig(1M)`, `praudit(1M)`, `attributes(5)`

See the section on Auditing in *Oracle Solaris 11.1 Administration: Security Services*.

Name audit_warn – audit service warning script

Synopsis `/etc/security/audit_warn` *option* [*arguments*]

Description The `audit_warn` script processes warning and error messages from the audit service. When a problem is encountered, the audit service calls `audit_warn` with the appropriate arguments. The option argument specifies the type of problem.

The system administrator can specify a list of mail recipients to be notified when an `audit_warn` situation arises by defining a mail alias called `audit_warn` in `aliases(4)`. The users that make up the `audit_warn` alias are typically the `audit` and `root` users.

The default action is to send mail to the `audit_warn` alias and send the mail message to syslog with a `daemon.alert` priority.

The system administrator can customize the `audit_warn` script for the site's specific needs. Care should be taken when updating to a new release to resolve any changes in the release.

Options The following options are supported:

`allhard` *count*	Indicates that the hard limit for all `audit_binfile(5)` directory filesystems has been exceeded *count* times. To avoid filling the mail spool directory, mail is sent only if the count is `1`.
`allsoft`	Indicates that the soft limit for all `audit_binfile(5)` directory filesystems has been exceeded.
`ars message`	Indicates that the Audit Remote Server experienced an error.
`auditoff`	Indicates that the kernel audit subsystem has failed while the audit service is running. The audit service exits in this case.
`config message`	Indicates the audit service detected a configuration error.
`hard` *directory*	Indicates that the hard limit for the `audit_binfile(5)` directory filesystem has been exceeded.
`hostname`	Indicates that the audit service could not find an IP address to associate with the local hostname. It has fallen back to using the “loopback” address. Audit trail translation tools might not translate the hostname properly. See `/var/audit/debug` for more information. The audit service can be refreshed (`audit -s`) to retry to find an IP address.
`nostart`	Indicates that auditing could not be started because the audit subsystem system calls are reporting failure.

`plugin` *name error count text*
: Indicates that an error occurred during execution of the audit service plugin *name*. To avoid filling the mail spool directory, mail is sent only if the count is `1`. A separate count is kept for each error type. The *text* field provides the detailed error message passed from the plug-in. The *error* field is one of the following strings:

  `load_error`
  : Unable to load the plugin *name*.

  `sys_error`
  : The plugin *name* is not executing due to a system error such as a lack of resources.

  `config_error`
  : No plug-ins loaded (including the binary file plug-in, `audit_binfile(5)`) due to configuration errors (see the `-setplugin` option of the `auditconfig(1M)` command). The name string is `--`, to indicate that no plug-in name applies.

  `retry`
  : The plugin *name* reports it has encountered a temporary failure. For example, the `audit_binfree.so` plugin uses `retry` to indicate that all directories are full.

  `no_memory`
  : The plugin *name* reports a failure due to lack of memory.

  `invalid`
  : The plugin *name* reports it received an invalid input.

  `failure`
  : The plugin *name* has reported an error as described in *text*.

`soft` *directory*
: Indicates that the soft limit for the `audit_binfile(5)` directory filesystem has been exceeded.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os
Interface Stability	See below

The command is Committed. The script content is Uncommitted. The presence and contents of `/var/audit/debug` is Not-an-Interface. The syslog and mail output is Not-an-Interface.

See Also `logger(1)`, `mailx(1)`, `audit(1M)`, `auditconfig(1M)`, `auditd(1M)`, `aliases(4)`, `audit.log(4)`, `syslog.conf(4)`, `attributes(5)`, `audit_binfile(5)`

See the section on Auditing in *Oracle Solaris 11.1 Administration: Security Services*.

Notes This functionality is available only when the audit service is enabled.

Hard and soft limits deal with the list of `audit_binfile(5)` and Audit Remote Server directories and the configured free space. When the currently active directory is filled beyond the configured free space, a "soft" limit is reached and the next directory in the list is tried. When the currently active directory space is exhausted a "hard" limit is reached and the next directory in the list is tried.

See the pkg(5) man page (not a SunOS page) for guidance on resolving changes across release updates.

If the `perzone` audit policy is set or `perzone` is not set and the Audit Remote Server is enabled, the `/etc/security/audit_warn` script for the local zone is used for notifications from the local zone's instance of the audit service. If the `perzone` policy is not set and Audit Remote Server is not enabled in the local zone, all audit service errors are generated by the global zone's copy of `/etc/security/audit_warn`.

Name automount – install automatic mount points

Synopsis

```
/usr/sbin/automount [-t duration] [-v]
```

Description The `automount` utility installs `autofs` mount points and associates an automount map with each mount point. It starts the `automountd(1M)` daemon if it finds any non-trivial entries in either local or distributed automount maps and if the daemon is not already running. The `autofs` file system monitors attempts to access directories within it and notifies the `automountd(1M)` daemon. The daemon uses the map to locate a file system, which it then mounts at the point of reference within the `autofs` file system. A map can be assigned to an `autofs` mount using an entry in the `/etc/auto_master` map or a direct map.

If the file system is not accessed within an appropriate interval (`10` minutes by default), the `automountd` daemon unmounts the file system.

The file `/etc/auto_master` determines the locations of all `autofs` mount points. By default, this file contains three entries:

```
# Master map for automounter
#
+auto_master
/net          -hosts    -nosuid
/home         auto_home
```

The `+auto_master` entry is a reference to an external NIS master map. If one exists, then its entries are read as if they occurred in place of the `+auto_master` entry. The remaining entries in the master file specify a directory on which an `autofs` mount will be made followed by the automounter map to be associated with it. Optional mount options may be supplied as an optional third field in the each entry. These options are used for any entries in the map that do not specify mount options explicitly. The `automount` command is usually run without arguments. It compares the entries `/etc/auto_master` with the current list of `autofs` mounts in `/etc/mnttab` and adds, removes or updates `autofs` mounts to bring the `/etc/mnttab` up to date with the `/etc/auto_master`. At boot time it installs all `autofs` mounts from the master map. Subsequently, it may be run to install `autofs` mounts for new entries in the master map or the direct map, or to perform unmounts for entries that have been removed from these maps.

SMF Management The `automount` service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/system/filesystem/autofs:default
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. The service's status can be queried using the `svcs(1)` command.

Startup autofs SMF parameters for `automount` can be manipulated using `sharectl(1M)`. Refer to `sharectl(1M)` for instructions for viewing and updating these parameters. Supported parameters are:

`timeout=`*num*
Specifies a duration, in seconds, that a file system is to remain mounted when not in use. The default value is `600` (10 minutes). Equivalent to the `-t` option in `automount`.

`automount_verbose=TRUE | FALSE`
Verbose mode. Causes you to be notified of non-critical events, suchs as `autofs` mounts and unmounts. The default value is `FALSE`. Equivalent to the `-v` option in `automount`.

Automount with Oracle Solaris Trusted Extensions

If a system is configured with Oracle Solaris Trusted Extensions, users have home directories at each label within their clearance. Therefore a home directory needs to be available in each corresponding labeled zone. Instead of using the `auto_home` map, a special map is automatically generated, using the zone's name as a suffix. By default the map contains the single entry:

```
-fstype=lofs    :/export/home/&
```

When a home directory is referenced and the name does not match any other keys in the zone's `auto_home_`*zonename* map, it will match this loopback mount specification. If this loopback match occurs and the name corresponds to a valid user whose home directory does not exist in the zone, the directory is automatically created on behalf of the user.

It is also possible to share home directories in a zone, in read-only mode, with higher-level zones, using NFS. In this case, the higher-level zone needs to have an automap entry for each lower-level zone that is to be imported. A typical map entry for the public zone, to be interpreted in the internal zone, would be called `auto_home_public`, and would look like this:

```
+auto_home_public

public-zone-IP-address:/export/home/&
```

This automap entry would then be included in `/etc/auto_master`, as follows:

```
/zone/public/home       auto_home_public       -nobrowse
```

Users in higher-level zones can use the `updatehome(1)` utility to synchronize specific startup files using their minimum labeled zone as the source.

Options The following options are supported:

`-t` *duration* Specifies a *duration*, in seconds, that a file system is to remain mounted when not in use. The default is `10` minutes.

`-v` Verbose mode. Notifies of `autofs` mounts, unmounts, or other non-essential information.

Usage

Map Entry Format

A simple map entry (mapping) takes the form:

`key [ -`*mount-options* `] `*location* `. . .`

where `key` is the full pathname of the directory to mount when used in a direct map, or the simple name of a subdirectory in an indirect map. *mount-options* is a comma-separated list of `mount` options, and *location* specifies a file system from which the directory may be mounted. In the case of a simple NFS mount, the options that can be used are as specified in `mount_nfs(1M)`, and *location* takes the form:

host`:` *pathname*

host is the name of the host from which to mount the file system, and *pathname* is the absolute pathname of the directory to mount.

Options to other file systems are documented on the other `mount_*` reference manual pages, for example, `mount_nfs(1M)`.

Replicated File Systems

Multiple *location* fields can be specified for replicated NFS file systems, in which case `automount` and the kernel will each try to use that information to increase availability. If the read-only flag is set in the map entry, `automountd` mounts a list of locations that the kernel may use, sorted by several criteria. Only locations available at mount time will be mounted, and thus be available to the kernel. When a server does not respond, the kernel will switch to an alternate server. The sort ordering of `automount` is used to determine how the next server is chosen. If the read-only flag is not set, `automount` will mount the best single location, chosen by the same sort ordering, and new servers will only be chosen when an unmount has been possible, and a remount is done. Servers on the same local subnet are given the strongest preference, and servers on the local net are given the second strongest preference. Among servers equally far away, response times will determine the order if no weighting factors (see below) are used.

If the list includes server locations using both the NFS Version 2 Protocol and the NFS Version 3 Protocol, `automount` will choose only a subset of the server locations on the list, so that all entries will be the same protocol. It will choose servers with the NFS Version 3 Protocol so long as an NFS Version 2 Protocol server on a local subnet will not be ignored. See the *System Administration Guide: IP Services* for additional details.

If each *location* in the list shares the same *pathname* then a single *location* may be used with a comma-separated list of hostnames:

`hostname,hostname . . . :` *pathname*

Requests for a server may be weighted, with the weighting factor appended to the server name as an integer in parentheses. Servers without a weighting are assumed to have a value of zero (most likely to be selected). Progressively higher values decrease the chance of being selected. In the example,

```
man -ro alpha,bravo,charlie(1),delta(4) : /usr/man
```

hosts `alpha` and `bravo` have the highest priority; host `delta` has the lowest.

Server proximity takes priority in the selection process. In the example above, if the server `delta` is on the same network segment as the client, but the others are on different network segments, then `delta` will be selected; the weighting value is ignored. The weighting has effect only when selecting between servers with the same network proximity. The automounter always selects the localhost over other servers on the same network segment, regardless of weighting.

In cases where each server has a different export point, the weighting can still be applied. For example:

```
man -ro alpha:/usr/man  bravo,charlie(1):/usr/share/man
     delta(3):/export/man
```

A mapping can be continued across input lines by escaping the NEWLINE with a backslash (\) Comments begin with a number sign (#) and end at the subsequent NEWLINE.

Map Key Substitution The ampersand (&) character is expanded to the value of the `key` field for the entry in which it occurs. In this case:

```
jane sparcserver : /home/&
```

the & expands to `jane`.

Wildcard Key The asterisk (*) character, when supplied as the `key` field, is recognized as the catch-all entry. Such an entry will match any key not previously matched. For instance, if the following entry appeared in the indirect map for `/config`:

```
*         & : /export/config/&
```

this would allow automatic mounts in `/config` of any remote file system whose location could be specified as:

```
hostname : /export/config/hostname
```

Note that the wildcard key does not work in conjunction with the `-browse` option.

Variable Substitution Client specific variables can be used within an `automount` map. For instance, if `$HOST` appeared within a map, `automount` would expand it to its current value for the client's host name. Supported variables are:

`ARCH`	The output of `arch`	The architecture name. For example, `sun4` on a `sun4u` machine.
`CPU`	The output of `uname -p`	The processor type.

		For example, “sparc”
HOST	The output of `uname -n`	The host name. For example, `myhost`.
KARCH	The output of `arch -k` or `uname –m`	The kernel architecture name or machine hardware name. For example, `sun4u`.
OSNAME	The output of `uname -s`	The OS name. For example, “SunOS”
OSREL	The output of `uname -r`	The OS release name. For example “5.3”
OSVERS	The output of `uname -v`	The OS version. For example, “beta1.0”
NATISA	The output of `isainfo -n`	The native instruction set architecture for the system. For example, “sparcv9”
PLATFORM	The output of `uname -i`	The platform name. For example, `SUNW,Sun-Fire-V240`.

If a reference needs to be protected from affixed characters, you can surround the variable name with curly braces (`{ }`).

Multiple Mounts

A multiple mount entry takes the form:

`key [`*-mount-options*`] [ [`*mountpoint*`] [`*-mount-options*`] `*location*`. . . ] . . .`

The initial `/`[*mountpoint*] is optional for the first mount and mandatory for all subsequent mounts. The optional *mountpoint* is taken as a pathname relative to the directory named by `key`. If *mountpoint* is omitted in the first occurrence, a *mountpoint* of `/` (root) is implied.

Given an entry in the indirect map for `/src`

```
beta     -ro\
  /           svr1,svr2:/export/src/beta  \
  /1.0        svr1,svr2:/export/src/beta/1.0  \
  /1.0/man    svr1,svr2:/export/src/beta/1.0/man
```

All offsets must exist on the server under `beta`. `automount` will automatically mount `/src/beta`, `/src/beta/1.0`, and `/src/beta/1.0/man`, as needed, from either `svr1` or `svr2`, whichever host is nearest and responds first.

Other File System Types

The automounter assumes NFS mounts as a default file system type. Other file system types can be described using the `fstype` mount option. Other mount options specific to this file system type can be combined with the `fstype` option. The location field must contain information specific to the file system type. If the location field begins with a slash, a colon character must be prepended, for instance, to mount a CD file system:

```
cdrom -fstype=hsfs,ro   : /dev/sr0
```

or to perform an `autofs` mount:

```
src -fstype=autofs    auto_src
```

Use this procedure only if you are not using Volume Manager.

See the `NOTES` section for information on option inheritance.

Indirect Maps

An indirect map allows you to specify mappings for the subdirectories you wish to mount under the `directory` indicated on the command line. In an indirect map, each `key` consists of a simple name that refers to one or more file systems that are to be mounted as needed.

Direct Maps

Entries in a direct map are associated directly with `autofs` mount points. Each *key* is the full pathname of an `autofs` mount point. The direct map as a whole is not associated with any single directory.

Direct maps are distinguished from indirect maps by the `/-` key. For example:

```
# Master map for automounter
#
+auto_master
/net            -hosts          -nosuid,nobrowse
/home           auto_home       -nobrowse
/-              auto_direct
```

Included Maps

The contents of another map can be included within a map with an entry of the form

+*mapname*

If *mapname* begins with a slash, it is assumed to be the pathname of a local file. Otherwise, the location of the map is determined by the policy of the name service switch according to the entry for the automounter in `/etc/nsswitch.conf`, such as

```
automount: files nis
```

If the name service is `files`, then the name is assumed to be that of a local file in `/etc`. If the key being searched for is not found in the included map, the search continues with the next entry.

Special Maps There are three special maps available: `-hosts`, `-fedfs` and `-null`. The `-hosts` map is used with the `/net` directory and assumes that the map key is the hostname of an NFS server. The `automountd` daemon dynamically constructs a map entry from the server's list of exported file systems. References to a directory under `/net/hermes` will refer to the corresponding directory relative to `hermes` root.

The `-fedfs` map is used with the `/nfs4` directory and assumes that the map key is the DNS domain for which the domain root filesystem is needed. The `automountd` daemon looks up the domain root servers with a query equivalent to:

```
% nslookup -q=srv _nfs4._domainroot._tcp.domain
```

...and mounts *server-list*`:/.domainroot-`*domain* at `/nfs4/`*domain*.

This supports the pending IETF standard documented in:

```
http://datatracker.ietf.org/doc/\
draft-ietf-nfsv4-federated-fs-dns-srv-namespace/
```

The `-null` map cancels a previous map for the directory indicated. This is most useful in the `/etc/auto_master` for cancelling entries that would otherwise be inherited from the `+auto_master` include entry. To be effective, the `-null` entries must be inserted before the included map entry.

Executable Maps Local maps that have the execute bit set in their file permissions will be executed by the automounter and provided with a key to be looked up as an argument. The executable map is expected to return the content of an automounter map entry on its stdout or no output if the entry cannot be determined. A direct map cannot be made executable.

Configuration and the auto_master Map When initiated without arguments, `automount` consults the master map for a list of `autofs` mount points and their maps. It mounts any `autofs` mounts that are not already mounted, and unmounts `autofs` mounts that have been removed from the master map or direct map.

The master map is assumed to be called `auto_master` and its location is determined by the name service switch policy. Normally the master map is located initially as a local file `/etc/auto_master`.

Browsing The `automount` daemon supports browsability of indirect maps. This allows all of the potential mount points to be visible, whether or not they are mounted. The `-nobrowse` option can be added to any indirect `autofs` map to disable browsing. For example:

```
/net     -hosts      -nosuid,nobrowse
/home    auto_home
```

In this case, any *hostnames* would only be visible in `/net` after they are mounted, but all potential mount points would be visible under `/home`. The `-browse` option enables browsability of `autofs` file systems. This is the default for all indirect maps.

The `-browse` option does not work in conjunction with the wildcard key.

Restricting Mount Maps

Options specified for a map are used as the default options for all the entries in that map. They are ignored when map entries specify their own mount options.

In some cases, however, it is desirable to force `nosuid`, `nodevices`, `nosetuid`, or `noexec` for a complete mount map and its submounts. This can be done by specifying the additional mount option, `-restrict`.

```
/home     auto_home       -restrict,nosuid,hard
```

The `-restrict` option forces the inheritance of all the restrictive options `nosuid`, `nodevices`, `nosetuid`, and `noexec` as well as the restrict option itself. In this particular example, the `nosuid` and `restrict` option are inherited but the `hard` option is not. The `restrict` option also prevents the execution of "executable maps" and is enforced for auto mounts established by programs with fewer than all privileges available in their zone.

Exit Status

The following exit values are returned:

`0` Successful completion.

`1` An error occurred.

Files

`/etc/auto_master`	Master automount map.
`/etc/auto_home`	Map to support automounted home directories.
`/etc/nsswitch.conf`	Name service switch configuration file. See `nsswitch.conf(4)`.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also

`isainfo(1)`, `ls(1)`, `svcs(1)`, `uname(1)`, `updatehome(1)`, `automountd(1M)`, `mount(1M)`, `mount_nfs(1M)`, `sharectl(1M)`, `svcadm(1M)`, `attributes(5)`, `nfssec(5)`, `smf(5)`

Oracle Solaris Administration: Network Services

Notes

`autofs` mount points must not be hierarchically related. `automount` does not allow an `autofs` mount point to be created within another `autofs` mount.

Since each direct map entry results in a new `autofs` mount such maps should be kept short.

Entries in both direct and indirect maps can be modified at any time. The new information is used when `automountd` next uses the map entry to do a mount.

New entries added to a master map or direct map will not be useful until the automount command is run to install them as new `autofs` mount points. New entries added to an indirect map may be used immediately.

As of the Solaris 2.6 release, a listing (see `ls(1)`) of the `autofs` directory associated with an indirect map shows all potential mountable entries. The attributes associated with the potential mountable entries are temporary. The real file system attributes will only be shown once the file system has been mounted.

Default mount options can be assigned to an entire map when specified as an optional third field in the master map. These options apply only to map entries that have no mount options. Note that map entities with options override the default options, as at this time, the options do not concatenate. The concatenation feature is planned for a future release.

When operating on a map that invokes an NFS mount, the default number of retries for the automounter is 0, that is, a single mount attempt, with no retries. Note that this is significantly different from the default (10000) for the `mount_nfs(1M)` utility.

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same.

Name automountd – autofs mount/unmount daemon

Synopsis `automountd [-Tvn] [-D` *name=value*`]`

Description `automountd` is an RPC server that answers file system mount and unmount requests from the `autofs` file system. It uses local files or name service maps to locate file systems to be mounted. These maps are described with the `automount(1M)` command.

If `automount` finds any non-trivial entries in either the local or distributed automount maps and if the daemon is not running already, the `automountd` daemon is automatically invoked by `automount(1M)`. `automountd` enables the `svc:/network/nfs/nlockmgr` service (`lockd(1M)`), and the `svc:/network/nfs/status` service (`statd(1M)`), if NFS mounts need to be done.

SMF Management The `automountd` service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/system/filesystem/autofs
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. The service's status can be queried using `svcs(1)`. If it is disabled, it is enabled by `automount(1M)` unless the `application/auto_enable` property is set to `false`.

Values for parameters for `automountd` are stored in SMF and manipulated using `sharectl(1M)`. The following are the supported parameters.

`automountd_verbose=TRUE | FALSE`
Verbose mode. Causes status messages to be logged to:

```
/var/svc/log/system-filesystem-autofs:default.log
```

See `smf(5)`. The default value is `FALSE`. Equivalent to the `-v` option.

`nobrowse=ON | OFF`
Turn on or off browsing for all `autofs` mount points. The default value is `OFF`. Equivalent to the `-n` option.

`trace=`*num*
Expands each RPC call and logs it to:

```
/var/svc/log/system-filesystem-autofs:default.log
```

See `smf(5)`. The default value, `0`, turns off such tracing. Starting with `1`, with each higher value, the verbosity of trace output increases. This property is equivalent to the `-T` option.

`environment=`*name=value*
Environment variables. You can specify multiple *name=value* pairs with a comma separator. If there is more than one value for a name, use the double forward slash to indicate that. See `EXAMPLES`. There are no environment variable settings supplied. This property is equivalent to the `-D` option.

All the above parameters can be changed using the `sharectl(1M)` command.

Options The following options are supported:

`-D` *name=value*	Assign *value* to the indicated `automount` map substitution variable. These assignments cannot be used to substitute variables in the master map `auto_master`.
`-n`	Turn off browsing for all `autofs` mount points. This option overrides the `-browse` `autofs` map option on the local host.
`-T`	Trace. Expand each RPC call and display it on the standard output.
`-v`	Verbose. Log status messages to the console.

Usage See `largefile(5)` for the description of the behavior of `automountd` when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

Examples **EXAMPLE 1** Setting Verbose Mode

The following command turns on verbose mode, causing status messages to be logged to the location specified in "SMF Management," above. See that section for a description of the properties specified in this and following examples.

```
# sharectl set -p automountd_verbose=true autofs
```

EXAMPLE 2 Turning on Browsing for Mount Points

The following command turns on browsing for all `autofs` mount points.

```
# sharectl set -p nobrowse=off autofs
```

EXAMPLE 3 Specifying and Displaying Environment Variables

The following commands set and display the value for an environment variable named `DAY`.

```
# sharectl set -p environment=DAY=TUES autofs
% sharectl get -p environment autofs
environment=DAY=TUES
```

The following command sets multiple parameters for an environment variable. The subsequent command displays the result.

```
# sharectl set -p environment=DAY=TUES,TIME=NOON autofs
% sharectl get -p environment autofs
environment=DAY=TUES,TIME=NOON
```

The following command set multiple values for name that is assigned to an environment variable. The subsequent command displays the result.

```
# sharectl set -p environment=DAY=MON\\,TUE,TIME=NOON autofs
% sharectl get -p environment autofs
environment=DAY=MON\,TUE,TIME=NOON
```

Files `/etc/auto_master` Master map for automounter.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also `svcs(1)`, `automount(1M)`, `sharectl(1M)`, `svcadm(1M)`, `attributes(5)`, `largefile(5)`, `smf(5)`

Name autopush – configures lists of automatically pushed STREAMS modules

Synopsis

```
autopush -f filename

autopush -g -M major -m minor

autopush -r -M major -m minor
```

Description The `autopush` command configures the list of modules to be automatically pushed onto the stream when a device is opened. It can also be used to remove a previous setting or get information on a setting.

Options The following options are supported:

`-f` *filename*
Sets up the `autopush` configuration for each driver according to the information stored in *filename*. An `autopush` file consists of lines of four or more fields, separated by spaces as shown below:

major minor last-minor module1 module2 . . . module8

The first field is a string that specifies the *major* device name, as listed in the `/kernel/drv` directory. The next two fields are integers that specify the *minor* device number and *last-minor* device number. The fields following represent the names of modules. If *minor* is `–1`, then all minor devices of a major driver specified by *major* are configured, and the value for *last-minor* is ignored. If *last-minor* is `0`, then only a single minor device is configured. To configure a range of minor devices for a particular major, *minor* must be less than *last-minor*.

The remaining fields list the names of modules to be automatically pushed onto the stream when opened, along with the position of an optional anchor. The maximum number of modules that can be pushed is eight. The modules are pushed in the order they are specified. The optional special character sequence `[anchor]` indicates that a STREAMS anchor should be placed on the stream at the module previously specified in the list; it is an error to specify more than one anchor or to have an anchor first in the list.

A nonzero exit status indicates that one or more of the lines in the specified file failed to complete successfully.

`-g`
Gets the current configuration setting of a particular *major* and *minor* device number specified with the `-M` and `-m` options respectively and displays the autopush modules associated with it. It will also return the starting minor device number if the request corresponds to a setting of a range (as described with the `-f` option).

`-m` *minor*
Specifies the minor device number.

`-M` *major*
Specifies the major device number.

`-r`
Removes the previous configuration setting of the particular *major* and *minor* device number specified with the `-M` and `-m` options respectively. If the values of *major* and *minor* correspond to a previously established setting of a range of minor devices, where *minor* matches the first minor device number in the range, the configuration would be removed for the entire range.

Exit Status The following exit values are returned:

0
Successful completion.

non-zero
An error occurred.

Examples **EXAMPLE 1** Using the `autopush` command.

The following example gets the current configuration settings for the *major* and *minor* device numbers as indicated and displays the `autopush` modules associated with them for the character-special device `/dev/term/a`:

```
example# autopush -g -M 29 -m 0
Major     Minor     Lastminor     Modules
  29        0         1           ldterm  ttcompat
```

Files `/etc/iu.ap`

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also `dladm(1M)`, `ttymon(1M)`, `attributes(5)`, `ldterm(7M)`, `sad(7D)`, `streamio(7I)`, `ttcompat(7M)`

STREAMS Programming Guide

Notes The use of the `autopush` command is obsolete for networking data-links. The preferred method of configuring a list of automatically pushed STREAMS modules on a given networking data-link interface is the `dladm(1M)` `autopush` link property.

Because network devices are self-cloning, the `autopush` command is inadequate for networking devices. The granularity of the `autopush` command's configuration is driver-wide, and not per-interface as one might expect. Another reason not to use `autopush` is that administrators are more familiar with the names of their network interfaces than with device major and minor numbers. The `dladm(1M)` command allows the configuration using data-link interface names.

Name bart – file integrity scanner and reporter

Synopsis

```
/usr/bin/bart create [ -n] [-R root_directory]
     [-r rules_file | -] [-a md5|sha1|sha256|sha384|sha512]

/usr/bin/bart create [-n] [-R root_directory] -I
     [-a md5|sha1|sha256|sha384|sha512] [file_name]...

/usr/bin/bart compare [-i attribute ] [-p]
     [-r rules_file | -] control-manifest test-manifest
```

Description `bart`(1M) is a rule-based file integrity scanning and reporting tool that uses cryptographic-strength checksums and file system metadata to report changes.

The `bart` utility performs two basic functions:

`bart create` The manifest generator tool takes a file-level *snapshot* of a system. The output is a catalog of file attributes referred to as a *manifest*. See `bart_manifest`(4).

You can specify that the list of files be cataloged in three ways. Use `bart create` with no options, specify the files by name on the command line, or create a rules file with directives that specify which the files to monitor. See `bart_rules`(4).

By default, the manifest generator catalogs all attributes of all files in the root (`/`) file system. File systems mounted on the root file system are cataloged only if they are of the same type as the root file system.

For example, `/`, `/usr`, and `/opt` are separate UFS file systems. `/usr` and `/opt` are mounted on `/`. Therefore, all three file systems are cataloged. However, `/tmp`, also mounted on `/`, is not cataloged because it is a TMPFS file system. Mounted CD-ROMs are not cataloged since they are HSFS file systems.

`bart compare` The report tool compares two manifests. The output is a list of per-file attribute discrepancies. These discrepancies are the differences between two manifests: a control manifest and a test manifest.

A discrepancy is a change to any attribute for a given file cataloged by both manifests. A new file or a deleted file in a manifest is reported as a discrepancy.

The reporting mechanism provides two types of output: verbose and programmatic. Verbose output is localized and presented on multiple lines, while programmatic output is more easily parsable by other programs. See `OUTPUT`.

By default, the report tool generates verbose output where all discrepancies are reported except for modified directory timestamps (`dirmtime` attribute).

To ensure consistent and accurate comparison results, *control-manifest* and *test-manifest* must be built with the same rules file.

Use the rules file to ignore specified files or subtrees when you generate a manifest or compare two manifests. Users can compare manifests from different perspectives by re-running the `bart compare` command with different rules files. See `bart_rules(4)` and `bart_manifest(4)`.

You can also specify the files to track and the types of discrepancies to flag by means of a rules file, `bart_rules`.

Options The following options are supported:

`-i` *attribute ...* Specify the file attributes to be ignored globally. Specify attributes as a comma separated list.

This option produces the same behavior as supplying the file attributes to a global `IGNORE` keyword in the rules file. See `bart_rules`(4).

`-I` [*file_name...*] Specify the input list of files. The file list can be specified at the command line or read from standard input.

`-n` Prevent computation of content signatures for all regular files in the file list.

`-p` Display manifest comparison output in "programmatic mode," which is suitable for programmatic parsing. The output is not localized.

`-r` *rules_file* Use *rules_file* to specify which files and directories to catalog, and to define which file attribute discrepancies to flag. If *rules_file* is `-`, then the rules are read from standard input. See `bart_rules`(4) for the definition of the syntax.

`-R` *root_directory* Specify the root directory for the manifest. All paths specified by the rules, and all paths reported in the manifest, are relative to *root_directory*.

Note – The root file system of any non-global zones must not be referenced with the `-R` option. Doing so might damage the global zone's file system, might compromise the security of the global zone, and might damage the non-global zone's file system. See `zones(5)`.

`-a` [*hash*] Specify the cryptographic digest algorithm to use for the hash of the file contents: `md5`, `sha1`, `sha256`, `sha512` are the currently supported values. If `-a` is not specified, `sha256` is used and a Version 1.1 manifest is created that indicates which hash algorithm is used. If `md5` is specified then a

Version 1.0 manifest is created.

Operands bart allows quoting of operands. This is particularly important for white-space appearing in subtree and subtree modifier specifications.

The following operands are supported:

control-manifest Specify the manifest created by `bart create` on the control system.

test-manifest Specify the manifest created by `bart create` on the test system.

Output The `bart create` and `bart compare` commands write output to standard output, and write error messages to standard error.

The `bart create` command generates a system manifest. See `bart_manifest`(4).

When the `bart compare` command compares two system manifests, it generates a list of file differences. By default, the comparison output is localized. However, if the `-p` option is specified, the output is generated in a form that is suitable for programmatic manipulation.

Default Format

filename
attribute `control:`*xxxx* `test:`*yyyy*

filename Name of the file that differs between *control-manifest* and *test-manifest*. For file names that contain embedded whitespace or newline characters, see `bart_manifest`(4).

attribute The name of the file attribute that differs between the manifests that are compared. *xxxx* is the attribute value from *control-manifest*, and *yyyy* is the attribute value from *test-manifest*. When discrepancies for multiple attributes occur for the same file, each difference is noted on a separate line.

The following attributes are supported:

acl ACL attributes for the file. For a file with ACL attributes, this field contains the output from `acltotext()`.

all All attributes.

contents Checksum value of the file. This attribute is only specified for regular files. If you turn off context checking or if checksums cannot be computed, the value of this field is `-`.

dest Destination of a symbolic link.

devnode Value of the device node. This attribute is for character device files and block device files only.

dirmtime Modification time in seconds since 00:00:00 UTC, January 1, 1970 for directories.

gid	Numerical group ID of the owner of this entry.
lnmtime	Creation time for links.
mode	Octal number that represents the permissions of the file.
mtime	Modification time in seconds since 00:00:00 UTC, January 1, 1970 for files.
size	File size in bytes.
type	Type of file.
uid	Numerical user ID of the owner of this entry.

The following default output shows the attribute differences for the `/etc/passwd` file. The output indicates that the `size`, `mtime`, and `contents` attributes have changed.

```
/etc/passwd:
  size  control:74  test:81
  mtime  control:3c165879  test:3c165979
  contents  control:daca28ae0de97afd7a6b91fde8d57afa
test:84b2b32c4165887355317207b48a6ec7
```

Programmatic Format — *filename attribute control-val test-val* [*attribute control-val test-val*]*

filename	Same as *filename* in the default format.
attribute control-val test-val	A description of the file attributes that differ between the control and test manifests for each file. Each entry includes the attribute value from each manifest. See `bart_manifest`(4) for the definition of the attributes.

Each line of the programmatic output describes all attribute differences for a single file.

The following programmatic output shows the attribute differences for the `/etc/passwd` file. The output indicates that the `size`, `mtime`, and `contents` attributes have changed.

```
/etc/passwd size 74 81 mtime 3c165879 3c165979
contents daca28ae0de97afd7a6b91fde8d57afa 84b2b32c4165887355317207b48a6ec7
```

Exit Status

Manifest Generator — The manifest generator returns the following exit values:

`0`	Success
`1`	Non-fatal error when processing files; for example, permission problems
`>1`	Fatal error; for example, invalid command-line options

Report Tool The report tool returns the following exit values:

`0` No discrepancies reported

`1` Discrepancies found

`>1` Fatal error executing comparison

Examples

EXAMPLE 1 Creating a Default Manifest Without Computing Checksums

The following command line creates a default manifest, which consists of all files in the `/` file system. The `-n` option prevents computation of checksums, which causes the manifest to be generated more quickly.

```
bart create -n
```

EXAMPLE 2 Creating a Manifest for a Specified Subtree

The following command line creates a manifest that contains all files in the `/home/nickiso` subtree.

```
bart create -R /home/nickiso
```

EXAMPLE 3 Creating a Manifest by Using Standard Input

The following command line uses output from the `find`(1) command to generate the list of files to be cataloged. The `find` output is used as input to the `bart create` command that specifies the `-I` option.

```
find /home/nickiso -print | bart create -I
```

EXAMPLE 4 Creating a Manifest by Using a Rules File

The following command line uses a rules file, `rules`, to specify the files to be cataloged.

```
bart create -r rules
```

EXAMPLE 5 Comparing Two Manifests and Generating Programmatic Output

The following command line compares two manifests and produces output suitable for parsing by a program.

```
bart compare -p manifest1 manifest2
```

EXAMPLE 6 Comparing Two Manifests and Specifying Attributes to Ignore

The following command line compares two manifests. The `dirmtime`, `lnmtime`, and `mtime` attributes are not compared.

```
bart compare -i dirmtime,lnmtime,mtime manifest1 manifest2
```

EXAMPLE 7 Comparing Two Manifests by Using a Rules File

The following command line uses a rules file, `rules`, to compare two manifests.

```
bart compare -r rules manifest1 manifest2
```

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	security/bart
Interface Stability	Committed

See Also

`cksum(1)`, `digest(1)`, `find(1)`, `bart_manifest(4)`, `bart_rules(4)`, `attributes(5)`

Notes

The file attributes of certain system libraries can be temporarily altered by the system as it boots. To avoid triggering false warnings, you should compare manifests only if they were both created with the system in the same state; that is, if both were created in single-user or both in multi-user.

Name beadm – manage ZFS boot environments

Synopsis

```
beadm create [-a] [-d description]
     [-e non-activeBeName | beName@snapshot]
     [-o property=value] ... [-p zpool] beName

beadm create beName@snapshot

beadm destroy [-fF] beName | beName@snapshot

beadm list [-a | -ds] [-H] [beName]

beadm mount beName mountpoint

beadm unmount [-f] beName

beadm rename beName newBeName

beadm activate beName
```

Description The `beadm` command is the user interface for managing ZFS Boot Environments (BEs). This utility is intended to be used by system administrators who want to manage multiple Oracle Solaris instances on a single system.

Using `beadm`, you can do the following:

- Create a new BE, based on the active BE.
- Create a new BE, based on an inactive BE.
- Create a snapshot of an existing BE.
- Create a new BE, based on an existing snapshot.
- Create a new BE, and copy it to a different zpool.
- Activate an existing, inactive BE.
- Mount a BE.
- Unmount a BE.
- Destroy a BE.
- Destroy a snapshot of a BE.
- Rename an existing, inactive BE.
- Display information about your snapshots and datasets.

Sub-commands The `beadm` command has the subcommands and options listed below. Usage of many of these subcommands and options is illustrated in `EXAMPLES`, below.

`beadm` *(no arguments)*
Displays command usage.

`beadm create` [`-a`] [`-d` *description*] [`-e` *non-activeBeName* | *beName@snapshot*] [`-o` *property=value*] ... [`-p` *zpool*] *beName*
Creates a new boot environment named *beName*. If the `-e` option is not provided, the new boot environment will be created as a clone of the currently running boot environment. If the `-d` option is provided, then the description is also used as the title for the BE's entry in the GRUB menu for x86 systems or in the boot menu for SPARC systems. If the `-d` option is

not provided, *beName* will be used as the title. Nested BEs do not support the use of the `-p` option. Also, non-bootable, nested BEs and snapshots of non-bootable, nested BEs cannot be used with the `-e` option.

`-a`
Activate the newly created BE upon creation. The default is to not activate the newly created BE.

`-d` *description*
Create a new BE with a description associated with it.

`-e` *non-activeBeName*
Create a new BE from an existing inactive BE. In a nested BE, only bootable BEs can be used with this option.

`-e` *beName@snapshot*
Create a new BE from an existing snapshot of the BE named *beName*. In a nested BE, only snapshots of bootable BEs can be used with this option.

`-o` *property=value*
Create the datasets for a new BE with specific ZFS properties. Multiple `-o` options can be specified. See `zfs(1M)` for more information on the `-o` option.

`-p` *zpool*
Create the new BE in the specified *zpool*. If this is not provided, the default behavior is to create the new BE in the same pool as the origin BE. This option is not supported inside of a nested BE.

`beadm create` *beName@snapshot*
Creates a snapshot of the existing BE named *beName*. Inside a nested BE, only bootable BEs can be snapshotted. When inside of a nested BE, only BEs that are bootable or BEs that are not bootable but are not marked as active on reboot can be destroyed.

`beadm destroy` [`-fF`] *beName* | *beName@snapshot*
Destroys the boot environment named *beName* or destroys an existing snapshot of the boot environment named *beName@snapshot*. Destroying a boot environment will also destroy all snapshots of that boot environment. Use this command with caution.

`-f`
Forcefully unmount the boot environment if it is currently mounted.

`-F`
Force the action without prompting to verify the destruction of the boot environment.

`beadm list` [`-a` | `-ds`] [`-H`] [*beName*]
Lists information about the existing boot environment named *beName*, or lists information for all boot environments if *beName* is not provided. The `Active` field indicates whether the boot environment is active now, represented by `N`; active on reboot, represented by `R`; or both, represented by `NR`. Unbootable BEs inside of a nested BE are represented by an exclamation point (`!`)

Each line in the machine-parsable output has the boot environment name as the first field. The `Space` field is displayed in bytes and the `Created` field is displayed in UTC format. The `-H` option used with no other options gives the boot environment's UUID in the second field. This field will be blank if the boot environment does not have a UUID. See the `EXAMPLES` section. Inside of a nested BE, the UUID field actually represents the parent id with which the nested BE is associated.

`-a`
Lists all available information about the boot environment. This includes subordinate file systems and snapshots.

`-d`
Lists information about all subordinate file systems belonging to the boot environment.

`-s`
Lists information about the snapshots of the boot environment.

`-H`
Do not list header information. Each field in the list information is separated by a semicolon.

`beadm mount` *beName mountpoint*
Mounts a boot environment named *beName* at *mountpoint*. *mountpoint* must be an already existing empty directory.

`beadm unmount` [`-f`] *beName*
Unmounts a boot environment named *beName*.

`-f`
Forcefully unmount the boot environment even if it is currently busy.

`beadm rename` *beName newBeName*
Renames the boot environment named *beName* to *newBeName*. In a nested BE, only bootable BEs can be renamed.

`beadm activate` *beName*
Makes *beName* the active BE on next reboot. In a nested BE, only bootable BEs can be activated.

Nested BE Support

`beadm` supports the concept of a nested BE, specifically, as it pertains to BEs for non-global zones. Currently, `beadm` can only manage nested BEs from inside of a non-global zone.

`beadm` functions inside of a non-global zone much the same as it does from the global zone, with a few exceptions. First, the `-p` (alternate pool) option to `beadm create` is not supported within a non-global zone. Second, there is a distinction made for any given nested BE (or snapshot of a BE) to determine if it is bootable or not bootable. A nested BE is bootable if it is associated (that is, shares the same parent id as the active global zone BE's UUID) with the currently active global zone BE. It is unbootable—and marked with an '`!`' in the active column in `beadm list`)—otherwise. Note that, while the non-global zone administrator could mark

such a BE as active by means of `beadm activate`, rebooting the non-global zone would not result in the BE being loaded, because the BE is associated with a non-active global zone BE. Based on these conditions, `beadm` restricts some actions on unbootable BEs thusly:

- You cannot destroy a nested BE that is both unbootable and marked as active on reboot.
- You cannot activate an unbootable BE.
- You cannot snapshot an unbootable BE.
- You cannot use an unbootable BE or BE snapshot with the `-e` option to `beadm create`.
- You cannot rename an unbootable BE.

Examples

EXAMPLE 1 Creating a New BE Using Active BE

The following command creates a new BE, `BE1`, by cloning the current BE.

```
# beadm create BE1
```

EXAMPLE 2 Creating a New BE Using Inactive BE

The following command creates a new BE, `BE2`, by cloning the existing inactive BE named `BE1`.

```
# beadm create -e BE1 BE2
```

EXAMPLE 3 Creating a Snapshot of Existing BE

The following command creates a snapshot named now of the existing BE named `BE1`.

```
# beadm create BE1@now
```

EXAMPLE 4 Cloning a Snapshot to Create a New BE

The following command creates a new BE named `BE3`, by cloning an existing snapshot of `BE1`.

```
# beadm create -e BE1@now BE3
```

EXAMPLE 5 Creating a New BE in Specified zpool

The following command creates a new BE named `BE4`, based on the currently running BE. The command creates the new BE in the zpool `rpool2`.

```
# beadm create -p rpool2 BE4
```

EXAMPLE 6 Creating a New BE in Specified zpool with Compression Enabled

The following command creates a new BE named `BE5`, based on the currently running BE. The command creates the new BE in the zpool `rpool2` and creates its datasets with compression turned on.

```
# beadm create -p rpool2 -o compression=on BE5
```

EXAMPLE 7 Creating a New BE and Providing a Description

The following command creates a new BE named BE6, based on the currently running BE, and provides a description for it.

```
# beadm create -d "BE6 used as test environment" BE6
```

EXAMPLE 8 Activating a BE

The following command activates an existing, inactive BE named BE3.

```
# beadm activate BE3
```

EXAMPLE 9 Mounting a BE

The following command mounts the BE named BE3 at /mnt.

```
# beadm mount BE3 /mnt
```

EXAMPLE 10 Unmounting a BE

The following command unmounts the BE named BE3.

```
# beadm unmount BE3
```

EXAMPLE 11 Destroying a BE

The following command destroys the BE named BE3 without asking for confirmation.

```
# beadm destroy -F BE3
```

EXAMPLE 12 Destroying a Snapshot

The following command destroys the snapshot named now of BE1.

```
# beadm destroy BE1@now
```

EXAMPLE 13 Renaming a BE

The following command renames the existing, inactive BE named BE1 to BE3.

```
# beadm rename BE1 BE3
```

EXAMPLE 14 Listing All BEs

The following command lists all existing BEs.

```
# beadm list
BE  Active Mountpoint Space  Policy Created
--  ------ ---------- -----  ------ -------
BE2 -      -          72.0K  static 2008-05-21 12:26
BE3 -      -          332.0K static 2008-08-26 10:28
BE4 -      -          15.78M static 2008-09-05 18:20
BE5 NR     /          7.25G  static 2008-09-09 16:53
```

EXAMPLE 15 Listing All BEs with Dataset and Snapshot Info

The following command lists all existing BEs and list all dataset and snapshot information about those boot environments.

```
# beadm list -d -s
BE/Dataset/Snapshot     Active Mountpoint Space   Policy Created
-------------------     ------ ---------- -----   ------ -------
BE2
   p/ROOT/BE2           -      -          36.0K   static 2008-05-21 12:26
   p/ROOT/BE2/opt       -      -          18.0K   static 2008-05-21 16:26
   p/ROOT/BE2/opt@now   -      -          0       static 2008-09-08 22:43
   p/ROOT/BE2@now       -      -          0       static 2008-09-08 22:43
BE3
   p/ROOT/BE3           -      -          192.0K  static 2008-08-26 10:28
   p/ROOT/BE3/opt       -      -          86.0K   static 2008-08-26 10:28
   p/ROOT/BE3/opt/local -      -          36.0K   static 2008-08-28 10:58
BE4
   p/ROOT/BE4           -      -          15.78M  static 2008-09-05 18:20
BE5
   p/ROOT/BE5           NR     /          6.10G   static 2008-09-09 16:53
   p/ROOT/BE5/opt       -      /opt       24.55M  static 2008-09-09 16:53
   p/ROOT/BE5/opt@bar   -      -          18.38M  static 2008-09-10 00:59
   p/ROOT/BE5/opt@foo   -      -          18.38M  static 2008-06-10 16:37
   p/ROOT/BE5@bar       -      -          139.44M static 2008-09-10 00:59
   p/ROOT/BE5@foo       -      -          912.85M static 2008-06-10 16:37
```

EXAMPLE 16 Listing Dataset and Snapshot Info for a BE

The following command lists all dataset and snapshot information about BE5.

```
# beadm list -a BE5
BE/Dataset/Snapshot   Active Mountpoint Space   Policy Created
-------------------   ------ ---------- -----   ------ -------
BE5
   p/ROOT/BE5         NR     /          6.10G   static 2008-09-09 16:53
   p/ROOT/BE5/opt     -      /opt       24.55M  static 2008-09-09 16:53
   p/ROOT/BE5/opt@bar -      -          18.38M  static 2008-09-10 00:59
   p/ROOT/BE5/opt@foo -      -          18.38M  static 2008-06-10 16:37
   p/ROOT/BE5@bar     -      -          139.44M static 2008-09-10 00:59
   p/ROOT/BE5@foo     -      -          912.85M static 2008-06-10 16:37
```

EXAMPLE 17 Listing in Machine-Parseable Format

The following command lists information about all BEs in machine-parseable format.

```
# beadm list -H
BE2;;;;55296;static;1211397974
BE3;;;;339968;static;1219771706
BE4;;;;16541696;static;1220664051
BE5;215b8387-4968-627c-d2d0-f4a011414bab;NR;/;7786206208;static;1221004384
```

EXAMPLE 18 Displaying Non-bootable BEs

The following command lists all BEs. When run inside of a non-global zone, it displays both bootable and non-bootable BEs. Non-bootable BEs are designated with an exclamation point (`!`) in the active column.

```
# beadm list
BE    Active Mountpoint Space   Policy Created
--    ------ ---------- -----   ------ -------
zbe-0 -      -          29.22M  static 2011-03-04 09:14
zbe-1 NR     /          815.10M static 2011-03-04 09:28
zbe-2 -      -          35.0K   static 2011-03-04 09:28
zbe-3 -      -          35.0K   static 2011-03-04 09:28
zbe-4 -      -          35.0K   static 2011-03-04 09:28
zbe-5 !      -          35.0K   static 2011-03-04 11:47
zbe-6 !      -          54.0K   static 2011-03-07 14:37
```

Exit Status

`0`

Success.

`>0`

Failure.

Files

`/var/log/beadm/`*beName*`/create.log.`*yyyymmdd_hhmmss*

Log used for capturing `beadm create` output. The time designation portion of the file name is explained as follows.

- *yyyymmdd_hhmmss* — for example, `20071130_140558`.
- *yyyy* — year, `2007`
- *mm* — month, `11`
- *dd* — day, `30`
- *hh* — hour, `14`
- *mm* — minute, `05`
- *ss* — second, `58`

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/boot-environment-utilities
Interface Stability	Committed

See Also

`zfs(1M)`, `attributes(5)`

Name boot – start the system kernel or a standalone program

Synopsis

SPARC
```
boot [OBP names] [file] [-aLV] [-F object] [-D default-file]
     [-Z dataset] [boot-flags] [—] [client-program-args]
```

x86
```
kernel$ root_path/platform/i86pc/amd64/kernel/unix
     /platform/i86pc/amd64/kernel/unix [boot-args]
     [-B prop=val [,val...]]
```

Description Bootstrapping is the process of loading and executing a standalone program. For the purpose of this discussion, bootstrapping means the process of loading and executing the bootable operating system. Typically, the standalone program is the operating system kernel (see `kernel(1M)`), but any standalone program can be booted instead. On a SPARC-based system, the diagnostic monitor for a machine is a good example of a standalone program other than the operating system that can be booted.

If the standalone is identified as a dynamically-linked executable, `boot` will load the interpreter (linker/loader) as indicated by the executable format and then transfer control to the interpreter. If the standalone is statically-linked, it will jump directly to the standalone.

Once the kernel is loaded, it starts the UNIX system, mounts the necessary file systems (see `vfstab(4)`), and runs `/usr/sbin/init` to bring the system to the "initdefault" state specified in `/etc/inittab`. See `inittab(4)`.

SPARC Bootstrap Procedure

On SPARC based systems, the bootstrap procedure on most machines consists of the following basic phases.

After the machine is turned on, the system firmware (in PROM) executes power-on self-test (POST). The form and scope of these tests depends on the version of the firmware in your system.

After the tests have been completed successfully, the firmware attempts to autoboot if the appropriate flag has been set in the non-volatile storage area used by the firmware. The name of the file to load, and the device to load it from can also be manipulated.

These flags and names can be set using the `eeprom(1M)` command from the shell, or by using PROM commands from the `ok` prompt after the system has been halted.

The second level program is either a fileystem-specific boot block (when booting from a disk), or `inetboot` or `wanboot` (when booting across the network).

Network Booting

Network booting occurs in two steps: the client first obtains an IP address and any other parameters necessary to permit it to load the second-stage booter. The second-stage booter in turn loads the boot archive from the boot device.

An IP address can be obtained in one of three ways: RARP, DHCP, or manual configuration, depending on the functions available in and configuration of the PROM. Machines of the `sun4u` and `sun4v` kernel architectures have DHCP-capable PROMs.

The boot command syntax for specifying the two methods of network booting are:

```
boot net:rarp
boot net:dhcp
```

The command:

```
boot net
```

without a `rarp` or `dhcp` specifier, invokes the default method for network booting over the network interface for which `net` is an alias.

The sequence of events for network booting using RARP/`bootparams` is described in the following paragraphs. The sequence for DHCP follows the RARP/`bootparams` description.

When booting over the network using RARP/`bootparams`, the PROM begins by broadcasting a reverse ARP request until it receives a reply. When a reply is received, the PROM then broadcasts a TFTP request to fetch the first block of `inetboot`. Subsequent requests will be sent to the server that initially answered the first block request. After loading, `inetboot` will also use reverse ARP to fetch its IP address, then broadcast `bootparams` RPC calls (see `bootparams(4)`) to locate configuration information and its root file system. `inetboot` then loads the boot archive by means of NFS and transfers control to that archive.

When booting over the network using DHCP, the PROM broadcasts the hardware address and kernel architecture and requests an IP address, boot parameters, and network configuration information. After a DHCP server responds and is selected (from among potentially multiple servers), that server sends to the client an IP address and all other information needed to boot the client. After receipt of this information, the client PROM examines the name of the file to be loaded, and will behave in one of two ways, depending on whether the file's name appears to be an HTTP URL. If it does not, the PROM downloads `inetboot`, loads that file into memory, and executes it. `inetboot` loads the boot archive, which takes over the machine and releases `inetboot`. Startup scripts then initiate the DHCP agent (see `dhcpagent(1M)`), which implements further DHCP activities.

If the file to be loaded is an HTTP URL, the PROM will use HTTP to load the referenced file. If the client has been configured with an HMAC SHA-1 key, it will check the integrity of the loaded file before proceeding to execute it. The file is expected to be the `wanboot` binary. The WAN boot process can be configured to use either DHCP or NVRAM properties to discover the install server and router and the proxies needed to connect to it. When `wanboot` begins executing, it determines whether sufficient information is available to it to allow it to proceed. If any necessary information is missing, it will either exit with an appropriate error or bring up a command interpreter and prompt for further configuration information. Once `wanboot` has obtained the necessary information, it loads the boot loader into memory by means of HTTP.

If an encryption key has been installed on the client, `wanboot` will verify the boot loader's signature and its accompanying hash. Presence of an encryption key but no hashing key is an error.

The `wanboot` boot loader can communicate with the client using either HTTP or secure HTTP. If the former, and if the client has been configured with an HMAC SHA-1 key, the boot loader will perform an integrity check of the root file system. Once the root file system has been loaded into memory (and possibly had an integrity check performed), the boot archive is transferred from the server. If provided with a `boot_logger` URL by means of the `wanboot.conf(4)` file, `wanboot` will periodically log its progress.

Not all PROMs are capable of consuming URLs. You can determine whether a client is so capable using the `list-security-keys` OBP command (see `monitor(1M)`).

WAN booting is not currently available on the x86 platform.

The `wanboot` Command Line

When the client program is `wanboot`, it accepts `client-program-args` of the form:

`boot ... -o` *opt1*`[,`*opt2*`[,...]]`

where each option may be an action:

`dhcp`
Require `wanboot` to obtain configuration parameters by means of DHCP.

`prompt`
Cause `wanboot` to enter its command interpreter.

<*cmd*>
One of the interpreter commands listed below.

...or an assignment, using the interpreter's parameter names listed below.

The `wanboot` Command Interpreter

The `wanboot` command interpreter is invoked by supplying a `client-program-args` of "`-o prompt`" when booting. Input consists of single commands or assignments, or a comma-separated list of commands or assignments. The configuration parameters are:

`host-ip`
IP address of the client (in dotted-decimal notation)

`router-ip`
IP address of the default router (in dotted-decimal notation)

`subnet-mask`
subnet mask (in dotted-decimal notation)

`client-id`
DHCP client identifier (a quoted ASCII string or hex ASCII)

`hostname`
hostname to request in DHCP transactions (ASCII)

`http-proxy`
HTTP proxy server specification (IPADDR[:PORT])

The key names are:

`3des`
the triple DES encryption key (48 hex ASCII characters)

`aes`
the AES encryption key (32 hex ASCII characters)

`sha1`
the HMAC SHA-1 signature key (40 hex ASCII characters)

Finally, the URL or the WAN boot CGI is referred to by means of:

`bootserver`
URL of WAN boot's CGI (the equivalent of OBP's `file` parameter)

The interpreter accepts the following commands:

`help`
Print a brief description of the available commands

var=*val*
Assign *val* to *var*, where *var* is one of the configuration parameter names, the key names, or `bootserver`.

var=
Unset parameter *var*.

`list`
List all parameters and their values (key values retrieved by means of OBP are never shown).

`prompt`
Prompt for values for unset parameters. The name of each parameter and its current value (if any) is printed, and the user can accept this value (press Return) or enter a new value.

`go`
Once the user is satisfied that all values have been entered, leave the interpreter and continue booting.

`exit`
Quit the boot interpreter and return to OBP's `ok` prompt.

Any of these assignments or commands can be passed on the command line as part of the `-o` options, subject to the OBP limit of 128 bytes for boot arguments. For example, `-o list,go` would simply list current (default) values of the parameters and then continue booting.

iSCSI Boot iSCSI boot is supported on both x86 and SPARC.

iSCSI Boot on x86

For iSCSI boot on x86, the host being booted must be equipped with NIC(s) capable of iBFT (iSCSI Boot Firmware Table) or have the mainboard's BIOS be iBFT-capable. iBFT, defined in the Advanced Configuration and Power Interface (ACPI) 3.0b specification, specifies a block of information that contains various parameters that are useful to the iSCSI Boot process.

Firmware implementing iBFT presents an iSCSI disk in the BIOS during startup as a bootable device by establishing the connection to the iSCSI target. The rest of the process of iSCSI booting is the same as booting from a local disk.

To configure the iBFT properly, users need to refer to the documentation from their hardware vendors.

iSCSI Boot on SPARC

iSCSI boot on SPARC is supported with OpenBoot level 4.31 and above, and does not require a specific NIC.

The boot command in OpenBoot takes a series of keywords to identify the destination iSCSI target, following the *keyword=value* format. The complete form of the iSCSI boot command is:

```
boot net:iscsi-target-ip=t-ip,iscsi-target-name=name
     host-ip=h-ip[,router-ip=r-ip]
     [,subnet-mask=m-ip]
     [,iscsi-port=port]
     [,iscsi-lun=lun]
     [,iscsi-partition=partition]
```

The descriptions of the preceding keywords are as follows:

`host-ip`	IP address of booting host.
`router-ip`	IP address of routing gateway.
`subnet-mask`	Subnet mask of `host-ip`.
`iscsi-target-ip`	IP address of iSCSI target storing OS.
`iscsi-target-name`	Name of iSCSI target storing OS.
`iscsi-partition`	Partition containing the bootable root.
`iscsi-port`	IP port of the target.
`iscsi-lun`	LUN to be booted off on target.

The values of `iscsi-target-ip`, `route-ip`, and `subnet-mask` are in standard, IPv4 dotted-decimal format; for example, `255.255.255.0` for `subnet-mask`. IPv6 is not supported in the current OpenBoot implementation.

The value of `iscsi-port`, a decimal number, is in the range of 1 to 65535.

The value of `iscsi-lun` is in the format of a dashed hexadecimal LUN, *ffff-ffff-ffff-ffff*. Please refer to section 5 of RFC 4173 for details. Leading zeroes and trailing dashes can be excluded, thus, `3`, for example, is equivalent to `0003-0000-0000-0000`.

The value of `iscsi-partition` is one ASCII character, used to specify the root partition. Most commonly, it is `a`.

The value of `iscsi-target-name` is in the format of a string, as specified by RFC 3720 and RFC 3722.

Two security keys are added to provide CHAP authentication on the target side. These are:

`chap-user` CHAP name

`chap-password` CHAP secret

Currently these two keys can be set with the command `set-ascii-security-key` at the Open Boot PROM (`ok`) prompt. For example:

```
ok set-ascii-security-key chap-user chap name
ok set-ascii-security-key chap-password chap password
```

Bi-directional authentication is not yet supported. These two variables can be changed only under the Open Boot PROM prompt.

RFC 4173 is supported, to retrieve iSCSI boot information from a DHCP server. The DHCP server must specify the Root Path option for the booting client, after which the client can do an iSCSI boot by means of the simple command:

```
boot net:dhcp
```

Currently the key `boot-device` is used to retrieve the physical boot device path during iSCSI boot. This key is setup during the Solaris installation. A manually modified key value might break iSCSI boot.

Booting from Disk

When booting from disk, the OpenBoot PROM firmware reads the boot blocks from the partition specified as the boot device. This standalone booter usually contains a file reader capable of reading the boot archive.

If the pathname to the standalone is relative (does not begin with a slash), the second level boot will look for the standalone in a platform-dependent search path. This path is guaranteed to contain `/platform/`*platform-name*. Many SPARC platforms next search the platform-specific path entry `/platform/`*hardware-class-name*. See `filesystem(5)`. If the pathname is absolute, `boot` will use the specified path. The `boot` program then loads the standalone at the appropriate address, and then transfers control.

Once the boot archive has been transferred from the boot device, Solaris can initialize and take over control of the machine. This process is further described in the "Boot Archive Phase," below, and is identical on all platforms.

If the filename is not given on the command line or otherwise specified, for example, by the `boot-file` NVRAM variable, `boot` chooses an appropriate default file to load based on what software is installed on the system and the capabilities of the hardware and firmware.

The path to the kernel must not contain any whitespace.

Booting from ZFS

Booting from ZFS differs from booting from UFS in that, with ZFS, a device specifier identifies a storage pool, not a single root file system. A storage pool can contain multiple bootable datasets (that is, root file systems). Therefore, when booting from ZFS, it is not sufficient to specify a boot device. One must also identify a root file system within the pool that was identified by the boot device. By default, the dataset selected for booting is the one identified by the pool's `bootfs` property. This default selection can be overridden by specifying an alternate bootable dataset with the `-Z` option. Use the `-L` option to list the bootable datasets within a ZFS pool.

Boot Archive Phase

The boot archive contains a file system image that is mounted using an in-memory disk. The image is self-describing, specifically containing a file system reader in the boot block. This file system reader mounts and opens the RAM disk image, then reads and executes the kernel contained within it. By default, this kernel is in:

```
/platform/‘uname -i‘/kernel/unix
```

If booting from ZFS, the pathnames of both the archive and the kernel file are resolved in the root file system (that is, dataset) selected for booting as described in the previous section.

The initialization of the kernel continues by loading necessary drivers and modules from the in-memory filesystem until I/O can be turned on and the root filesystem mounted. Once the root filesystem is mounted, the in-memory filesystem is no longer needed and is discarded.

OpenBoot PROM `boot` Command Behavior

The OpenBoot `boot` command takes arguments of the following form:

```
ok boot [device-specifier] [arguments]
```

The default `boot` command has no arguments:

```
ok boot
```

If no *device-specifier* is given on the `boot` command line, OpenBoot typically uses the *boot-device* or *diag-device* `NVRAM` variable. If no optional *arguments* are given on the command line, OpenBoot typically uses the *boot-file* or *diag-file* `NVRAM` variable as default `boot` arguments. (If the system is in diagnostics mode, *diag-device* and *diag-file* are used instead of *boot-device* and *boot-file*).

arguments may include more than one string. All *argument* strings are passed to the secondary booter; they are not interpreted by OpenBoot.

If any *arguments* are specified on the `boot` command line, then neither the *boot-file* nor the *diag-file* `NVRAM` variable is used. The contents of the `NVRAM` variables are not merged with command line arguments. For example, the command:

```
ok boot -s
```

ignores the settings in both *boot-file* and *diag-file*; it interprets the string "`-s`" as *arguments*. `boot` will not use the contents of *boot-file* or *diag-file*.

With older PROMs, the command:

```
ok boot net
```

took no arguments, using instead the settings in *boot-file* or *diag-file* (if set) as the default file name and arguments to pass to boot. In most cases, it is best to allow the `boot` command to choose an appropriate default based upon the system type, system hardware and firmware, and upon what is installed on the root file system. Changing *boot-file* or *diag-file* can generate unexpected results in certain circumstances.

This behavior is found on most OpenBoot 2.x and 3.x based systems. Note that differences may occur on some platforms.

The command:

ok **`boot cdrom`**

...also normally takes no arguments. Accordingly, if *boot-file* is set to the 64-bit kernel filename and you attempt to boot the installation CD or DVD with `boot cdrom`, boot will fail if the installation media contains only a 32-bit kernel.

Because the contents of *boot-file* or *diag-file* can be ignored depending on the form of the `boot` command used, reliance upon *boot-file* should be discouraged for most production systems.

When executing a WAN boot from a local (CD or DVD) copy of `wanboot`, one must use:

ok **`boot cdrom -F wanboot - install`**

Modern PROMs have enhanced the network boot support package to support the following syntax for arguments to be processed by the package:

[*protocol*,] [*key*=*value*,]*

All arguments are optional and can appear in any order. Commas are required unless the argument is at the end of the list. If specified, an argument takes precedence over any default values, or, if booting using DHCP, over configuration information provided by a DHCP server for those parameters.

protocol, above, specifies the address discovery protocol to be used.

Configuration parameters, listed below, are specified as *key=value* attribute pairs.

`tftp-server`
IP address of the TFTP server

`file`
file to download using TFTP or URL for WAN boot

`host-ip`
IP address of the client (in dotted-decimal notation)

`router-ip`
IP address of the default router

`subnet-mask`
subnet mask (in dotted-decimal notation)

`client-id`
DHCP client identifier

`hostname`
hostname to use in DHCP transactions

`http-proxy`
HTTP proxy server specification (IPADDR[:PORT])

`tftp-retries`
maximum number of TFTP retries

`dhcp-retries`
maximum number of DHCP retries

The list of arguments to be processed by the network boot support package is specified in one of two ways:

- As arguments passed to the package's `open` method, or
- arguments listed in the NVRAM variable `network-boot-arguments`.

Arguments specified in `network-boot-arguments` will be processed only if there are no arguments passed to the package's `open` method.

Argument Values

protocol specifies the address discovery protocol to be used. If present, the possible values are `rarp` or `dhcp`.

If other configuration parameters are specified in the new syntax and style specified by this document, absence of the *protocol* parameter implies manual configuration.

If no other configuration parameters are specified, or if those arguments are specified in the positional parameter syntax currently supported, the absence of the *protocol* parameter causes the network boot support package to use the platform-specific default address discovery protocol.

Manual configuration requires that the client be provided its IP address, the name of the boot file, and the address of the server providing the boot file image. Depending on the network configuration, it might be required that `subnet-mask` and `router-ip` also be specified.

If the *protocol* argument is not specified, the network boot support package uses the platform-specific default address discovery protocol.

`tftp-server` is the IP address (in standard IPv4 dotted-decimal notation) of the TFTP server that provides the file to download if using TFTP.

When using DHCP, the value, if specified, overrides the value of the TFTP server specified in the DHCP response.

The TFTP RRQ is unicast to the server if one is specified as an argument or in the DHCP response. Otherwise, the TFTP RRQ is broadcast.

file specifies the file to be loaded by TFTP from the TFTP server, or the URL if using HTTP. The use of HTTP is triggered if the file name is a URL, that is, the file name starts with `http:` (case-insensitive).

When using RARP and TFTP, the default file name is the ASCII hexadecimal representation of the IP address of the client, as documented in a preceding section of this document.

When using DHCP, this argument, if specified, overrides the name of the boot file specified in the DHCP response.

When using DHCP and TFTP, the default file name is constructed from the root node's `name` property, with commas (,) replaced by periods (.).

When specified on the command line, the filename must not contain slashes (`/`).

The format of URLs is described in RFC 2396. The HTTP server must be specified as an IP address (in standard IPv4 dotted-decimal notation). The optional port number is specified in decimal. If a port is not specified, port 80 (decimal) is implied.

The URL presented must be “safe-encoded”, that is, the package does not apply escape encodings to the URL presented. URLs containing commas must be presented as a quoted string. Quoting URLs is optional otherwise.

`host-ip` specifies the IP address (in standard IPv4 dotted-decimal notation) of the client, the system being booted. If using RARP as the address discovery protocol, specifying this argument makes use of RARP unnecessary.

If DHCP is used, specifying the `host-ip` argument causes the client to follow the steps required of a client with an "Externally Configured Network Address", as specified in RFC 2131.

`router-ip` is the IP address (in standard IPv4 dotted-decimal notation) of a router on a directly connected network. The router will be used as the first hop for communications spanning networks. If this argument is supplied, the router specified here takes precedence over the preferred router specified in the DHCP response.

`subnet-mask` (specified in standard IPv4 dotted-decimal notation) is the subnet mask on the client's network. If the subnet mask is not provided (either by means of this argument or in the DHCP response), the default mask appropriate to the network class (Class A, B, or C) of the address assigned to the booting client will be assumed.

`client-id` specifies the unique identifier for the client. The DHCP client identifier is derived from this value. Client identifiers can be specified as:

- The ASCII hexadecimal representation of the identifier, or
- a quoted string

Thus, `client-id="openboot"` and `client-id=6f70656e626f6f74` both represent a DHCP client identifier of 6F70656E626F6F74.

Identifiers specified on the command line must must not include slash (`/`) or spaces.

The maximum length of the DHCP client identifier is 32 bytes, or 64 characters representing 32 bytes if using the ASCII hexadecimal form. If the latter form is used, the number of characters in the identifier must be an even number. Valid characters are 0-9, a-f, and A-F.

For correct identification of clients, the client identifier must be unique among the client identifiers used on the subnet to which the client is attached. System administrators are responsible for choosing identifiers that meet this requirement.

Specifying a client identifier on a command line takes precedence over any other DHCP mechanism of specifying identifiers.

`hostname` (specified as a string) specifies the hostname to be used in DHCP transactions. The name might or might not be qualified with the local domain name. The maximum length of the hostname is 255 characters.

Note – The `hostname` parameter can be used in service environments that require that the client provide the desired hostname to the DHCP server. Clients provide the desired hostname to the DHCP server, which can then register the hostname and IP address assigned to the client with DNS.

`http-proxy` is specified in the following standard notation for a host:

host `[":"` *port*`]`

...where *host* is specified as an IP ddress (in standard IPv4 dotted-decimal notation) and the optional *port* is specified in decimal. If a port is not specified, port 8080 (decimal) is implied.

`tftp-retries` is the maximum number of retries (specified in decimal) attempted before the TFTP process is determined to have failed. Defaults to using infinite retries.

`dhcp-retries` is the maximum number of retries (specified in decimal) attempted before the DHCP process is determined to have failed. Defaults to of using infinite retries.

x86 Bootstrap Procedure

On x86 based systems, the bootstrapping process consists of two conceptually distinct phases, kernel loading and kernel initialization. Kernel loading is implemented in GRUB (GRand Unified Bootloader) using the firmware on the system board and firmware extensions in ROMs on peripheral boards. The system firmware loads GRUB. The loading mechanism differs, depending on the type of system firmware that is shipped on the system board.

For systems with BIOS firmware, the first physical sector of a hard disk (known as the boot sector) is loaded into memory and its code executed. Traditionally, this code has inspected the DOS partition table, has found the partition marked as the active one, and has loaded the first sector from *that* partition into memory, and (finally) has executed that code. Disks that are partitioned with the GPT (GUID Partition Table) must have boot sector code that behaves differently, loading code from another location (because the GPT scheme does not reserve the first sector of each partition for boot sector code storage). In the case of GRUB running on BIOS firmware, that other location is a dedicated partition known as the BIOS Boot Partition. Once GRUB's boot sector code loads the rest of GRUB into memory, the boot process continues in earnest. Booting from a DVD, the firmware's reading special data structures (defined by the El Torito Bootable CD Specification) from the disc and loading sectors from the DVD into memory, as defined by those structures. These sectors comprise the first stage boot program. This boot program then loads the next stage, which, in the case of Solaris, is GRUB itself. Booting from the network is yet a different process on BIOS systems. Bootable network adapters include firmware that complies with the PXE (Preboot eXecution Environment) specification. When activated, the PXE firmware performs a DHCP exchange on the network, and downloads the BootFile that the DHCP server included in the DHCP response from the TFTP server that is also in the DHCP response. For Solaris, this BootFile (`pxegrub2`, or equivalent) is GRUB itself. GRUB then proceeds, ultimately, to download the Unix kernel and the boot archive (see below), loads them into memory, and transfers control to Unix.

For systems with UEFI-based firmware, the boot process is quite different. The UEFI firmware looks for the EFI System Partition (ESP) on disks that it has enumerated, and loads and executes UEFI boot programs according to a UEFI-specification-defined process. The net result is that a UEFI boot application is loaded into memory and executed. For Solaris, that UEFI boot application is GRUB, which has been specifically built to run as a UEFI boot application. The boot process then continues largely as it does on systems with BIOS firmware. Booting from a DVD also involves a search for a UEFI boot application, but the search method is quite different and uses data structures on the DVD defined by the El Torito Bootable CD Specification. The UEFI specification defines how the El Torito specification is

used to locate UEFI boot applications. The boot process for network boot on a UEFI system is very similar to that of a BIOS system, except that UEFI systems make a slightly different DHCP request, which provides the DHCP server enough information to customize the BootFile that is returned for the UEFI system. Recall that UEFI systems require UEFI boot applications, not BIOS-targeted boot programs, which would otherwise be returned as the BootFile from the DHCP server. Once the UEFI boot application (which is GRUB itself) specified in the BootFile (`grub2netx64.efi`, or equivalent) is downloaded to the UEFI client, it (GRUB) is executed. As with the BIOS network boot process, GRUB downloads the Unix kernel and boot archive from the DHCP-specified TFTP server, loads them into memory, then transfers control to Unix.

Once GRUB is running, it executes script commands in its configuration file, `grub.cfg`, and, when directed to do so, loads the SunOS kernel (Unix) kernel and a pre-constructed boot archive that contains kernel modules and essential data required for boot.

If the device identified by GRUB as the boot device contains a ZFS storage pool, the `grub.cfg` file used to create the GRUB menu will be found in the pool's top level dataset. (This is the dataset with the same name as the pool itself.) There is always exactly one such dataset in a pool, so this dataset is well-suited for pool-wide data such as the GRUB configuration files and data. After the system is booted, this dataset is mounted at /*poolname* in the root file system.

There can be multiple bootable datasets (that is, root file systems) within a pool. The default root file system in a pool is identified by the pool's `bootfs` property (see `zpool(1M)`). If a specific `bootfs` (file system consistent with the naming scheme /*root*/*name*) is not specified (by means of the `zfs-bootfs` command in a GRUB `menuentry` block in the `grub.cfg`), the default `bootfs` root file system is used. Each GRUB menu entry may specify the `bootfs` to use, enabling the administrator to select from many bootable Solaris instances in a pool.

Kernel initialization starts when GRUB finishes loading the boot archive and hands control over to the `unix` binary. At this point, GRUB becomes inactive and no more I/O occurs with the boot device. The Unix operating system initializes, links in the necessary modules from the boot archive and mounts the root file system on the real root device. At this point, the kernel regains storage I/O, mounts additional file systems (see `vfstab(4)`), and starts various operating system services (see `smf(5)`).

Enabling Automatic Rebooting (x86)

The Solaris operating system supports an `smf(5)` property that enables a system to automatically reboot from the current boot device, to recover from conditions such as an out-of-date boot archive.

The service `svc:/system/boot-config:default` contains the boolean property `auto-reboot-safe`, which is set to `false` by default. Setting it to `true` communicates that both the system's firmware and default GRUB menu entry are set to boot from the current boot device. The value of this property can be changed using `svccfg(1M)` and `svcadm(1M)`. For example, to set auto-reboot-safe to enable automatic rebooting, enter a command such as:

```
example# svccfg -s svc:/system/boot-config:default \
      setprop config/auto-reboot-safe = true
```

Most systems are configured for automatic reboot from the current boot device. However, in some instances, automatic rebooting to an unknown operating system might produce undesirable results. For these instances, the `auto-reboot-safe` property allows you to specify the behavior you want.

Failsafe Mode

A requirement of booting from a root filesystem image built into a boot archive then remounting root onto the actual root device is that the contents of the boot archive and the root filesystem must be consistent. Otherwise, the proper operation and integrity of the machine cannot be guaranteed.

The term "consistent" means that all files and modules in the root filesystem are also present in the boot archive and have identical contents. Since the boot strategy requires first reading and mounting the boot archive as the first-stage root image, all unloadable kernel modules and initialization derived from the contents of the boot archive are required to match the real root filesystem. Without such consistency, it is possible that the system could be running with a kernel module or parameter setting applied to the root device before reboot, but not yet updated in the root archive. This inconsistency could result in system instability or data loss.

Once the root filesystem is mounted, and before relinquishing the in-memory filesystem, Solaris performs a consistency verification against the two file systems. If an inconsistency is detected, Solaris will automatically try to fix it and reboot into the same boot environment. If this fails (or if the system is an x86 machine that does not support fast reboot and has `auto-reboot-safe` not set to true), then the failsafe mode will be entered. Correcting the inconsistency requires the administrator take one of two steps. The recommended procedure is to reboot to a boot environment known to be consistent and rebuild the boot archive. This ensures that a known kernel is booted and functioning for the archive rebuild process. Alternatively, the administrator can elect to clear the inconsistent boot archive service state and continue system bring-up if the inconsistency is such that correct system operation will not be impaired. See `svcadm(1M)`.

If the boot archive service is cleared and system bring-up is continued (the second alternative above), the system may be running with unloadable kernel drivers or other modules that are out-of-date with respect to the root filesystem. As such, correct system operation may be compromised.

To ensure that the boot archive is consistent, the normal system shutdown process, as initiated by `reboot(1M)` and `shutdown(1M)`, checks for and applies updates to the boot archive at the conclusion of the `umountall(1M)` milestone.

An update to any kernel file, driver, module or driver configuration file that needs to be included in the boot archive after the `umountall` service is complete will result in a failed boot archive consistency check during the next boot. To avoid this, it is recommended to always shut down a machine cleanly.

If an update is required to the kernel after completion of the `umountall` service, the administrator may elect to rebuild the archive by invoking:

```
# bootadm update-archive
```

Options

SPARC The following SPARC options are supported:

`-a`
The boot program interprets this flag to mean `ask me`, and so it prompts for the name of the standalone. The `'-a'` flag is then passed to the standalone program.

`-D` *default-file*
Explicitly specify the *default-file*. On some systems, `boot` chooses a dynamic default file, used when none is otherwise specified. This option allows the *default-file* to be explicitly set and can be useful when booting `kmdb(1)` since, by default, `kmdb` loads the default-file as exported by the `boot` program.

`-F` *object*
Boot using the named object. The object must be either an ELF executable or bootable object containing a boot block. The primary use is to boot the failsafe or `wanboot` boot archive.

`-L`
List the bootable datasets within a ZFS pool. You can select one of the bootable datasets in the list, after which detailed instructions for booting that dataset are displayed. Boot the selected dataset by following the instructions. This option is supported only when the boot device contains a ZFS storage pool.

`-V`
Display verbose debugging information.

boot-flags
The boot program passes all *boot-flags* to `file`. They are not interpreted by `boot`. See the `kernel(1M)` and `kmdb(1)` manual pages for information about the options available with the default standalone program.

client-program-args
The `boot` program passes all *client-program-args* to *file*. They are not interpreted by `boot`.

file
Name of a standalone program to `boot`. If a filename is not explicitly specified, either on the `boot` command line or in the *boot-file* NVRAM variable, `boot` chooses an appropriate default filename.

OBP names
Specify the open boot prom designations. For example, on Desktop SPARC based systems, the designation `/sbus/esp@0,800000/sd@3,0:a` indicates a SCSI disk (sd) at target 3, lun0 on the SCSI bus, with the esp host adapter plugged into slot 0.

`-Z` *dataset*
Boot from the root file system in the specified ZFS dataset.

x86 The following x86 options are supported:

`-B` *prop=val...*
One or more property-value pairs to be passed to the kernel. Multiple property-value pairs must be separated by a comma. Use of this option is the equivalent of the command: `eeprom` *prop=val*. See `eeprom(1M)` for available properties and valid values.

boot-args
The boot program passes all *boot-args* to `file`. They are not interpreted by `boot`. See `kernel(1M)` and `kmdb(1)` for information about the options available with the kernel.

Unless otherwise specified, an x86 system will boot `/platform/i86pc/kernel/amd64/unix`.

x86 Boot Sequence Details

After a PC-compatible machine is turned on, the system firmware executes a power-on self test (POST), runs BIOS extensions in peripheral board ROMs, and locates and installs firmware extensions from peripheral board ROMs, and begins the boot process through a firmware-specific mechanism.

For BIOS systems, software interrupt INT 19h is executed. The INT 19h handler typically performs the standard PC-compatible boot, which consists of trying to read the first physical sector from the first hard disk. The processor then jumps to the first byte of the sector image in memory.

For UEFI firmware, the process is quite different, as previously explained in the "x86 Bootstrap Procedure" section, above.

x86 Primary Bios Boot

The first sector on a disk medium contains the master boot record (which is either GRUB's first stage loader if GRUB is installed in the MBR, or another boot loader). This code is responsible for loading the next stage boot loader. For GRUB, that means loading the rest of GRUB into memory. Once that is done, GRUB is fully functional. It locates the GRUB prefix (the directory that contains the GRUB configuration file and GRUB loadable modules) and reads and executes the GRUB configuration file `/boot/grub/grub.cfg`. A similar sequence occurs for DVD or CD boot, but the master boot record location and contents are dictated by the El Torito specification (as previously described).

The first sector on a hard disk contains the master boot record (MBR), which contains the master boot program and the DOS partition table (also referred to as the `FDISK` table, named for the program that maintained it in DOS). If the disk is partitioned with the GPT scheme, the master boot program must be specialized to load the next stage boot loader into memory from a safe location on the disk. That safe location, in the case of GRUB, is a special GPT partition called the BIOS Boot Partition (BBP). This partition does not contain a file system, just empty space in which the second stage portion of GRUB can reside. It is from the BBP that the master boot program completes GRUB's loading.

If the disk is partitioned with the traditional DOS scheme, the master boot program finds the active partition in the DOS partition table, loads its first sector, and jumps to that which it loaded into memory. This completes the standard PC-compatible hard disk boot sequence. If

GRUB's first stage is installed in the MBR (see the `-m` option of the bootadm(1M) `install-bootloader` subcommand), then the remainder of GRUB is loaded directly from the Solaris DOS partition, regardless of the active partition.

The Solaris DOS partition begins with a one-cylinder boot slice, which contains GRUB's first stage loader in the first sector, the standard Solaris disk label and volume table of contents (VTOC) in the second and third sectors, and the GRUB second stage loader in the fiftieth and subsequent sectors. The area from sector 4 to 49 is unused (because it had been used to store boot blocks for older versions of Solaris). When the DOS partition for the Solaris software is the active partition, the master boot program (`mboot`, the generic master boot program) loads the partition boot program from the Solaris partition's first sector into memory and jumps to it. It, in turn, reads GRUB's second stage loader into memory and jumps to it. Once the GRUB menu is displayed, the user can choose to boot an operating system on a different partition, a different disk, or possibly from the network (provided the proper firmware support is present).

For network booting, the supported method is Intel's Preboot eXecution Environment (PXE) standard. When booting from the network using PXE, the system or network adapter BIOS uses DHCP to locate a network bootstrap program (`pxegrub2`) on a boot server and reads it using Trivial File Transfer Protocol (TFTP). The BIOS executes the `pxegrub2` by jumping to its first byte in memory. The `pxegrub` program downloads a configuration file and presents the entries to user.

x86 Kernel Startup

The kernel startup process is independent of the kernel loading process. During kernel startup, console I/O goes to the device specified by the `console` property.

When booting from UFS, the root device is specified by the `bootpath` property, and the root file system type is specified by the `fstype` property. These properties should be setup by the Solaris Install/Upgrade process in `/boot/solaris/bootenv.rc` and can be overridden with the `-B` option, described above (see the eeprom(1M) man page).

When booting from ZFS, the root device is specified by a set of boot parameters specified on the multiboot command line in the GRUB `menuentry`. These boot parameters are synthesized by the GRUB `zfs-bootfs` command and are stored in the GRUB environment variable whose name is specified as the second argument to `zfs-bootfs`. This variable is then supplied, along with the `-B` kernel argument to pass vital ZFS parameters that identify the root filesystem to the kernel. (The previous version of Solaris GRUB used the substitution macro `$ZFS-BOOTFS` for this purpose. This is no longer supported, because `$ZFS-BOOTFS` is not a valid GRUB variable name.)

If the console is not specified as a kernel argument, the console is derived from the `/boot/solaris/bootenv.rc` on the root file system of the Solaris instance that is being booted. If no console variable is present in that file, the default console device is set to the graphical text console, and system keyboard (USB and PS/2 keyboards are supported).

It is important to note that the Solaris console can be configured differently from the GRUB console. For example, the GRUB console can be configured (see `bootadm(1M)`'s `set-menu` subcommand) to use the screen and keyboard, while Solaris uses the serial port. The console transition will occur when GRUB transfers control to Solaris when the menu entry is booted.

Examples

SPARC **EXAMPLE 1** To Boot the Default Kernel In Single-User Interactive Mode

To boot the default kernel in single-user interactive mode, respond to the `ok` prompt with one of the following:

```
boot -as
```

```
boot disk3 -as
```

EXAMPLE 2 Network Booting with WAN Boot-Capable PROMs

To illustrate some of the subtle repercussions of various boot command line invocations, assume that the `network-boot-arguments` are set and that `net` is devaliased as shown in the commands below.

In the following command, device arguments in the device alias are processed by the device driver. The network boot support package processes arguments in `network-boot-arguments`.

```
boot net
```

The command below results in no device arguments. The network boot support package processes arguments in `network-boot-arguments`.

```
boot net:
```

The command below results in no device arguments. `rarp` is the only network boot support package argument. `network-boot-arguments` is ignored.

```
boot net:rarp
```

In the command below, the specified device arguments are honored. The network boot support package processes arguments in `network-boot-arguments`.

```
boot net:speed=100,duplex=full
```

EXAMPLE 3 Using `wanboot` with Older PROMs

The command below results in the `wanboot` binary being loaded from DVD or CD, at which time `wanboot` will perform DHCP and then drop into its command interpreter to allow the user to enter keys and any other necessary configuration.

```
boot cdrom -F wanboot -o dhcp,prompt
```

x86 **EXAMPLE 4** To Boot the Default Kernel in Single-User Interactive Mode

To boot the default kernel in single-user interactive mode, edit the GRUB `multiboot2` command line to read:

```
multiboot2 root_path/platform/i86pc/kernel/amd64/unix \
/platform/i86pc/kernel/amd64/unix -as
```

Files

`/etc/inittab`
Table in which the `initdefault` state is specified

`/usr/sbin/init`
Program that brings the system to the `initdefault` state

SPARC Only `/platform/`*platform-name*`/kernel/sparcv9/unix`
Default program to boot system.

x86 Only `/boot`
Directory containing boot-related files.

`/rpool/boot/grub/grub.cfg`
Menu of bootable operating systems displayed by GRUB. `/rpool` is a common convention. The pathname is configurable, depending on the capabilities of your installer. This file should never be edited directly, as it is auto- generated without notice. For an administrator-editable file, see `custom.cfg`, listed below.

`/rpool/boot/grub/custom.cfg`
Administrator-customizable supplemental GRUB configuration file. This file is "sourced" by `grub.cfg` after all other system-generated `grub.cfg` content is processed. This file will never be automatically overwritten, and can contain any valid GRUB configuration file syntax.

`/rpool/boot/grub/menu.conf`
Data file used by the Solaris boot administration infrastructure to store details of boot loader configuration that is ultimately used to build the GRUB configuration file (`grub.cfg`).

`/platform/i86pc/kernel/amd64/unix`
Default program to boot system.

See Also `kmdb(1)`, `uname(1)`, `bootadm(1M)`, `eeprom(1M)`, `init(1M)`, `installboot(1M)`, `kernel(1M)`, `monitor(1M)`, `shutdown(1M)`, `svcadm(1M)`, `svccfg(1M)`, `umountall(1M)`, `zpool(1M)`, `uadmin(2)`, `bootparams(4)`, `inittab(4)`, `vfstab(4)`, `wanboot.conf(4)`, `attributes(5)`, `filesystem(5)`, `smf(5)`

RFC 903, *A Reverse Address Resolution Protocol*, `http://www.ietf.org/rfc/rfc903.txt`

RFC 2131, *Dynamic Host Configuration Protocol*, `http://www.ietf.org/rfc/rfc2131.txt`

RFC 2132, *DHCP Options and BOOTP Vendor Extensions*, `http://www.ietf.org/rfc/rfc2132.txt`

RFC 2396, *Uniform Resource Identifiers (URI): Generic Syntax*, `http://www.ietf.org/rfc/rfc2396.txt`

Oracle Solaris Administration: Common Tasks

Sun Hardware Platform Guide

OpenBoot Command Reference Manual

Warnings The `boot` utility is unable to determine which files can be used as bootable programs. If the booting of a file that is not bootable is requested, the `boot` utility loads it and branches to it. What happens after that is unpredictable.

Notes *platform-name* can be found using the `-i` option of `uname(1)`. *hardware-class-name* can be found using the `-m` option of `uname(1)`.

The current release of the Solaris operating system does not support machines running an UltraSPARC-I CPU.

Name bootadm – manage boot configuration

Synopsis

```
/usr/sbin/bootadm update-archive [-vn] [-R altroot [-p platform]]

/usr/sbin/bootadm list-archive [-vn] [-R altroot [-p platform]]

/usr/sbin/bootadm install-bootloader [-Mfv] [-P pool] [-R path]
     [device1 ... deviceN]

x86 only

/usr/sbin/bootadm set-menu [-P pool] [-R altroot [-p platform]]
     {key=value [key=value ...]}

/usr/sbin/bootadm list-menu [-P pool] [-R altroot [-p platform]]
     [[-i entry_number] | entry_title]

/usr/sbin/bootadm generate-menu [-P pool] [-f]

/usr/sbin/bootadm add-entry [-P pool] [-i entry_number] entry_title

/usr/sbin/bootadm change-entry [-P pool] {[entry_title[,entry_title...]
| -i entry_number[,entry_number]...]} {key=value [key=value ...]
| set-default }

/usr/sbin/bootadm remove-entry [-P pool] {[entry_title[,entry_title...]
| -i entry_number[,entry_number]...]}
```

Description The `bootadm` command manages the boot archive and, with x86 boot environments, the GRUB (GRand Unified Bootloader) menu. For x86, both Legacy GRUB and GRUB2 are supported (but not concurrently).

The `update–archive` option provides a way for user to update the boot archive as a preventative measure or as part of a recovery procedure.

The `set-menu` subcommand allows you to switch the `auto-boot` timeout and default boot entry in the GRUB menu.

The `list-menu` subcommand displays the current GRUB menu entries, or, optionally, details about a specific entry identified by an index (if `-i` is used) or a title string (if `-i` is omitted).

The `install-bootloader` subcommand installs the system bootloader. It supersedes the functionality of `installgrub(1M)` on x86 and `installboot(1M)` on SPARC, as well as supporting installation of GRUB2's bootloader on x86.

The `generate-menu` subcommand provides a way to create a new menu configuration file for Solaris entries. If boot loader configuration files already exist, `-f` must be passed to force this subcommand to overwrite those files.

The `add-entry`, `change-entry` and `remove-entry` subcommands provide options to add, change, or remove an entry from the GRUB menu.

Note that OpenBoot PROM (OBP)-based machines, such as SPARC systems, do not use GRUB and have no boot menu manageable by `bootadm`.

The `bootadm` command determines dynamically the options supported by the image to be managed, so that `bootadm` invoked on one platform can be used to manage diskless clients of a different platform type.

Subcommands For the subcommands that support specifying *entry_title*, *entry_title* is a string that can be either double- or single-quoted.

An *entry_number* is a non-negative integer number representing the index of the menu entry in the GRUB menu.

The `bootadm` command has the following subcommands:

`update-archive` [`-vn`] [`-R` *altroot* [`-p` *platform*]]
Updates current boot archive if required. Applies to both SPARC and x86 platforms.

`list-archive` [`-vn`] [`-R` *altroot* [`-p` *platform*]]
Lists the files and directories to be included in the boot archive. Applies to both SPARC and x86 platforms.

`set-menu` [`-vn`] [`-R` *altroot* [`-p` *platform*]] {*key*=*value* [*key*=*value*]...}
Maintain the GRUB menu. A space-separated list of key-value pairs can be specified.

key=*value*
Possible values are:

`default`=*entry_number*
The entry number (for example, `0`, `1`, or `2`) in the GRUB menu designating the operating system to boot when the timer expires.

`timeout`=*seconds*
The number of seconds before the operating system designated by the default item number is booted. If the value is `-1`, auto boot is disabled.

`console`=*GRUB_console_type*
Sets the type of console used for GRUB.

Possible values are:

`'text'`
Selects a high resolution console.

`'graphics'`
Selects a high resolution console which additionally leads to graphical boot. If BIOS console redirection is enabled, `graphics` must not be used to set console.

`'serial'`
Serial console for GRUB bootloader. Please see `serial_params` below for specific settings of serial parameters.

If BIOS console redirection is enabled, `'serial'` must not be used to set `console`.

When a system is installed by booting with a serial console, that serial console will become the Solaris's kernel default console device. However, GRUB's console will *not* be changed to `serial` (it will be `'text'`).

`serial_params='`*port*[,*speed*[,*databits*[,*parity*[,*stopbits*[,*flowcontrol*]]]]]`'`
Specifies the serial parameters for the serial console.

port is a number specifying the serial port number.

speed is a number specifying the data rate for the connection in bits/second.

databits is the number of data bits in each character.

parity specifies the method for detecting transmission errors. Possible values are:

- `N` for no parity
- `O` for odd parity
- `E` for even parity

Values for *parity* are not case-sensitive.

stopbits specifies the stop bit sent for the character transmission. Possible values are `0` or `1`.

flowcontrol specifies the flow control. Possible values are:

- `H` for hardware flow control
- `S` for software flow control
- `N` for no flow control

If *serial_params* is not set, the default is:

```
0,9600,8,N,1,N
```

...which makes the first serial port (`COM1`), using 9600 bits/sec baud rate, no parity checking, with databits of 8 bits per character, stop bit of 1, and no flow control to be the default.

`quiet`
Specifies whether printing informative messages to the console should be suppressed. By default its value is `false`.

Possible values are `true` or `false`.

`splashimage`
Specifies the path to the file used as an image to appear during boot.

`foreground`
Sets the foreground color. It is a string of hex values with a format of *RRGGBB*, where *RR* is for Red, *GG* for Green and *BB* for Blue.

`background`
Sets the background color. See `foreground` for possible values.

`list-menu` [`-P` *pool*] [`-R` *altroot* [`-p` *platform*]]
Lists the current GRUB menu entries. This includes the autoboot-timeout, the default entry number, and the title of each entry. Applies to x86 platforms only. If an entry title or entry index is supplied, details about that specific entry are printed.

`generate-menu` [`-P` *pool*]
Create a new menu configuration that contains only the Solaris entries currently installed on the system.

`add-entry` [-P *pool*] [-i *entry_number*] *entry_title*
Create a new entry in the menu with given entry title.

If *entry_number* is specified, the new entry will be inserted at the given position, or added as the last entry if the given *entry_number* is more than current number of entries.

`change-entry` [`-P` *pool*] {[*entry_title*[,*entry_title*...] | `-i` *entry_number*[,*entry_number*]...]} { *key*=*value* [*key*=*value* ...] | `set-default` }
Modify the contents of a given entry or a comma-separated list of entries. An entry is specified either by an entry title or by an entry number. If there are multiple entries with the same title, all will be affected.

The special property, `set-default`, sets the entry to be the default entry to boot from when the timer expires. Only one entry in the subcommand can be specified when specifying this property.

A space separated list of key value pairs can be specified: *key*=*value*

Possible values are:

`title`=*entry_title*	The new title for the entry (or entries).
`kernel`=*path_to_kernel*	Path to the kernel. Example: `/platform/i86pc/kernel/amd64/unix`
`kargs`=*kernel_arguments*	Argument or a list of arguments passed to kernel during boot. Please refer to `kernel(1M)` for possible options. If there are any spaces in the list, value of the key should be enclosed in quotes or double quotes.
`boot_archive`=*path_to_boot_archive*	The path to the boot archive.
`bootfs`=*bootfs*	The `bootfs` property value. Please refer to `zpool(1M)` for further information.

`remove-entry` [`-P` *pool*] [{*entry_title* [,*entry_title*...] | `-i` *entry_number*[,*entry_number*...]}
Remove a given entry or a comma-separated list of entries. If there are multiple entries with the same specified title, all will be removed.

`install-bootloader` [`-Mfv`] [`-P` *pool*] [`-R` *path*] [*device1 ... deviceN*]
Install the system bootloader. If a list of devices is specified, the bootloader will be installed only on the given devices. Otherwise the bootloader will be installed on a list of devices that is automatically extracted from system configuration.

The device is the name of a raw character device of a slice or partition on the disk on which the root file system resides.

Options The `bootadm` command has the following options:

`-f`
In `install-bootloader` installation, forces the installation of the bootloader and bypasses any versioning checks for not downgrading the version of the bootloader on the system.

`-i`
Entry number or a list of comma-separated entry numbers to which to japply the specified operation.

`-M` [x86 systems with BIOS firmware only]
In an `install-bootloader` operation on x86 systems, installs the boot loader into the Master Boot Record (MBR), making it the system boot loader. The default (on systems with BIOS firmware) is to install the boot loader into the Partition Boot Record (PBR).

`-n`
In an `update-archive` operation, archive content is checked but not updated.

`-P` *pool*
The boot configuration associated with the specified pool to be used. When this option is not used, the current pool from which the system was booted is used for boot configuration.

`-p` *platform*
The platform, or machine hardware class, of the client. The platform type can only be specified together with `-R`, and is generally useful only for managing a diskless client where the client is of a different platform class than the server. Platform must be one of `i86pc`, `sun4u`, or `sun4v`.

`-R` *altroot*
Operation is applied to the path or alternate root path.

Note – The root file system of any non-global zones must not be referenced with the `-R` option. Doing so might damage the global zone's file system, might compromise the security of the global zone, and might damage the non-global zone's file system. See `zones(5)`.

`-v`
In an `update-archive` operation, stale files are displayed on stderr. In an `install-bootloader` operation, enables verbose mode to print more information about the process.

Examples

EXAMPLE 1 Updating the Current Boot Archive

The following command updates the current boot archive:

```
# bootadm update-archive
```

EXAMPLE 2 Updating the Boot Archive on an Alternate Root

The following command updates the boot archive on an alternate root:

```
# bootadm update-archive -R /a
```

EXAMPLE 3 Switching Default Boot Entry

The following command refers to the menu displayed in the previous example. The user selects Linux (item 2).

```
# bootadm set-menu default=2
```

or

```
# bootadm change-entry -i 2 set-default
```

EXAMPLE 4 Listing GRUB Menu Entries

The following command lists the GRUB menu entries:

```
# bootadm list-menu
The location for the active GRUB menu is: /stubboot/boot/grub/menu.lst
default 0
timeout 10
0 Solaris10
1 Solaris10 failsafe
2 Linux
```

EXAMPLE 5 Adding and Changing a Menu Entry

The following command adds a menu entry with the title "New Solaris Entry" at position 8 in the GRUB menu.

```
# bootadm  add-entry -i 8  "New Solaris Entry"
```

The following command changes the just-added entry with the kernel argument of `-s` to boot into level `s`.

```
# bootadm  change-entry "New Solaris Entry" kargs="-s"
```

EXAMPLE 6 Installing Bootloader on a Second Root Pool

The following command installs the bootloader on the pool `secondrpool`.

```
# bootadm install-bootloader -P secondrpool
```

EXAMPLE 7 Setting Foreground and Background Color

The following command sets the foreground color to be red and the background color to blue.

```
# bootadm set-menu splashimage=/boot/grub/splash.xpm.gz \
foreground=ff0000 background=0000ff
```

Exit Status

The following exit values are returned:

`0`
The command completed successfully.

`1`
The command exited due to an error.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os
Interface Stability	Committed

See Also

`boot(1M)`, `installboot(1M)`, `installgrub(1M)`, `kernel(1M)`, `zpool(1M)`, `attributes(5)`, `grub(5)`

Consult the GRUB home page, under:

`http://www.gnu.org/`

Name bootconfchk – verify the integrity of a network boot configuration file

Synopsis

```
/usr/sbin/bootconfchk [bootconf-file]
```

Description The `bootconfchk` command checks that the file specified is a valid network boot configuration file as described in `wanboot.conf(4)`.

Any discrepancies are reported on standard error.

Exit Status

`0` Successful completion.

`1` An error occurred.

`2` Usage error.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/boot/wanboot
Interface Stability	Committed

See Also `wanboot.conf(4)`, `attributes(5)`

Name busstat – report bus-related performance statistics

Synopsis

```
busstat -e device-inst | -h | -l

busstat [-a] [-n]
     [-w  device-inst [,pic0=event,picn=event ]]...
     [-r device-inst]... [interval  [count]]
```

Description `busstat` provides access to the bus-related performance counters in the system. These performance counters allow for the measurement of statistics like hardware clock cycles, bus statistics including DMA and cache coherency transactions on a multiprocessor system. Each bus device that supports these counters can be programmed to count a number of events from a specified list. Each device supports one or more Performance Instrumentation Counters (PIC) that are capable of counting events independently of each other.

Separate events can be selected for each PIC on each instance of these devices. `busstat` summarizes the counts over the last interval seconds, repeating forever. If a count is given, the statistics are repeated count times.

Only root users can program these counters. Non-root users have the option of reading the counters that have been programmed by a root user.

The default value for the *interval* argument is 1 second, and the default *count* is unlimited.

The devices that export these counters are highly platform-dependent and the data may be difficult to interpret without an in-depth understanding of the operation of the components that are being measured and of the system they reside in.

Options The following options are supported:

`-a`	Display absolute counter values. The default is `delta` values.
`-e` *device-inst*	Display the list of events that the specified device supports for each pic. Specify *device-inst* as device (`name`) followed by an optional instance number. If an instance number is specified, the events for that instance are displayed. If no instance number is specified, the events for the first instance of the specified device are displayed.
`-h`	Print a usage message.
`-l`	List the devices in the system which support performance counters.
`-n`	Do not display a title in the output. The default is to display titles.

`-r` *device-inst*	Read and display all pic values for the specified device Specify *device-inst* as *device* (`name`) followed by *instance number*, if specifying an instance number of a device whose counters are to be read and displayed. If all instances of this device are to be read, use *device* (`name`) without an instance number. All pic values will be sampled when using the `-r` option.
`-w` *device-inst* [,pic0=*event*] [,pic*n*=*event*]	Program (write) the specified devices to count the specified events. Write access to the counters is restricted to root users only. Non-root users can use `-r` option. Specify *device-inst* as *device* (`name`) followed by an optional *instance number*. If specifying an instance number of a device to program these events on. If all instances of this device are to be programmed the same, then use *device* without an instance number. Specify an event to be counted for a specified pic by providing a comma separated list of `pic`*n*=*event* values. The `-e` option displays all valid event names for each device. Any devices that are programmed will be sampled every interval seconds and repeated count times. It is recommended that the interval specified is small enough to ensure that counter wraparound will be detected. The rate at which counters wraparound varies from device to device. If a user is programming events using the `-w` option and `busstat` detects that another user has changed the events that are being counted, the tool will terminate as the programmed devices are now being controlled by another user. Only one user can be programming a device instance at any one time. Extra devices can be sampled using the `-r` option. Using multiple instances of the `-w` option on the same command line, with the same *device-inst* specifying a different list of events for the pics will give the effect of multiplexing for that device. `busstat` will

switch between the list of events for that device every interval seconds. Event can be a string representing the event name, or even a number representing the bit pattern to be programmed into the Performance Control Register (PCR). This assumes explicit knowledge of the meaning of the control register bits for a device. The number can be specified in hexadecimal, decimal, or octal, using the usual conventions of `strtol(3C)`.

Exit Status The following exit values are returned:

0 Successful completion.

1 An error occurred.

2 Another user is writing to the same devices.

Examples

SPARC Only **EXAMPLE 1** Programming and monitoring the Address Controller counters

In this example, `ac0` refers to the Address Controller instance 0. The counters are programmed to count Memory Bank stalls on an Ultra Enterprise system at 10 second intervals with the values displayed in absolute form instead of deltas.

```
# busstat -a -w ac0,pic0=mem_bank0_stall,pic1=mem_bank1_stall 10
time  dev   event0            pic0   event1             pic1
10    ac0   mem_bank0_stall   1234   mem_bank1_stall    5678
20    ac0   mem_bank0_stall   5678   mem_bank1_stall   12345
30    ac0   mem_bank0_stall  12345   mem_bank1_stall   56789
...
```

For a complete list of the supported events for a device, use the `-e` option.

EXAMPLE 2 Programming and monitoring the counters on all instances of the Address Controller

In this example, `ac` refers to all `ac` instances. This example programs all instances of the Address Controller counters to `count_clock` cycles and `mem_bank0_rds` at 2 second intervals, 100 times, displaying the values as deltas.

```
# busstat -w ac,pic0=clock_cycles,pic1=mem_bank0_rds 2 100
time  dev     event0          pic0            event1          pic1
2     ac0     clock_cycles    167242902       mem_bank0_rds   3144
2     ac1     clock_cycles    167254476       mem_bank0_rds   1392
4     ac0     clock_cycles    168025190       mem_bank0_rds   40302
4     ac1     clock_cycles    168024056       mem_bank0_rds   40580
...
```

EXAMPLE 3 Monitoring the events being counted

This example monitors the events that are being counted on the sbus1 device, 100 times at 1 second intervals. It suggests that a root user has changed the events that `sbus1` was counting to be `dvma_tlb_misses` and interrupts instead of `pio_cycles`.

```
% busstat -r sbus0 1 100

time    dev     event0              pic0       event1          pic1
1       sbus1   pio_cycles          2321       pio_cycles      2321
2       sbus1   pio_cycles          48         pio_cycles      48
3       sbus1   pio_cycles          49         pio_cycles      49
4       sbus1   pio_cycles          2281       pio_cycles      2281
5       sbus1   dvma_tlb_misses     0          interrupts      0
6       sbus1   dvma_tlb_misses     6          interrupts      2
7       sbus1   dvma_tlb_misses     8          interrupts      11
...
```

EXAMPLE 4 Event Multiplexing

This example programs `ac0` to alternate between counting (clock cycles, `mem_bank0_rds`) and (`addr_pkts`, `data_pkts`) at 2 second intervals while also monitoring what `ac1` is counting :

It shows the expected output of the above `busstat` command. Another root user on the machine has changed the events that this user had programmed and `busstat` has detected this and terminates the command with a message.

```
# busstat -w ac0,pic0=clock_cycles,pic1=mem_bank0_rds \
          -w ac0,pic0=addr_pkts,pic1=data_pkts \
          -r ac1 2

time    dev     event0          pic0            event1          pic1
2       ac0     addr_pkts       12866           data_pkts       17015
2       ac1     rio_pkts        385             rio_pkts        385
4       ac0     clock_cycles    168018914       mem_bank0_rds   2865
4       ac1     rio_pkts        506             rio_pkts        506
6       ac0     addr_pkts       144236          data_pkts       149223
6       ac1     rio_pkts        522             rio_pkts        522
8       ac0     clock_cycles    168021245       mem_bank0_rds   2564
8       ac1     rio_pkts        387             rio_pkts        387
10      ac0     addr_pkts       144292          data_pkts       159645
10      ac1     rio_pkts        506             rio_pkts        506
12      ac0     clock_cycles    168020364       mem_bank0_rds   2665
12      ac1     rio_pkts        522             rio_pkts        522
busstat: events changed (possibly by another busstat).
#
```

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also iostat(1M), mpstat(1M), vmstat(1M), strtol(3C), attributes(5)

Name captoinfo – convert a termcap description into a terminfo description

Synopsis

```
captoinfo [-1] [-v]... [-V] [-w width] filename...
```

Description `captoinfo` looks in *filename* for `termcap` descriptions. For each one found, an equivalent `terminfo` description is written to standard output, along with any comments found. A description which is expressed as relative to another description (as specified in the `termcap` `tc = field`) is reduced to the minimum superset before being displayed.

If no *filename* is given, then the environment variable `TERMCAP` is used for the filename or entry. If `TERMCAP` is a full pathname to a file, only the terminal whose name is specified in the environment variable `TERM` is extracted from that file. If the environment variable `TERMCAP` is not set, then the file `/usr/share/lib/termcap` is read.

Options

–1	Display the fields one to a line. Otherwise, the fields are printed several to a line, with a maximum width of 60 characters.
`-v`	Display tracing information on the standard error as the program runs. Specifying additional `-v` options displays more detailed information.
`-V`	Display the version of the program in use on the standard error and then exit.
`-w` *width*	Change the output to *width* characters.

Files

`/usr/share/lib/terminfo/?/*`	compiled terminal description database
`/usr/share/lib/termcap`	

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also `infocmp(1M)`, `curses(3CURSES)`, `terminfo(4)`, `attributes(5)`

Notes `captoinfo` should be used to convert `termcap` entries to `terminfo` entries because the `termcap` database may not be supplied in future releases.

Name catman – create the formatted files for the reference manual

Synopsis

```
/usr/bin/catman [-c] [-n] [-p] [-t] [-w] [-M directory]
     [-T macro-package] [sections]

/usr/bin/catman [-M directory] -w
```

Description The `catman` utility creates the preformatted versions of the on-line manual from the `nroff(1)` or `sgml(5)` input files. This feature allows easy distribution of the preformatted manual pages among a group of associated machines, since it makes the directories of preformatted manual pages self-contained and independent of the unformatted entries.

With the `-w` option, `catman` also creates index files, in the directories specified by the `MANPATH` or the `-M` option. If there is no `MANPATH` or `-M` option specified, unless `-n` is specified, `catman` creates index files at the `/usr/share/man/` and `/usr/gnu/share/man/` directories by default. When any specified or default directory is read-only, `catman` fails and displays an error message to stderr, indicating that writing is not allowed to the directory.

Each manual page is examined and those whose preformatted versions are missing or out of date are recreated. If any changes are made, `catman` recreates the index files.

If a manual page is a *shadow* page, that is, it sources another manual page for its contents, a symbolic link is made in the `cat`*x* or `fmt`*x* directory to the appropriate preformatted manual page.

Shadow files in an unformatted nroff source file are identified by the first line being of the form `.so man`*x*`/yyy.`*x*.

Shadow files in the SGML sources are identified by the string `SHADOW_PAGE`. The file entity declared in the shadow file identifies the file to be sourced.

Options The following options are supported:

`-c`
: Create unformatted nroff source files in the appropriate `man` subdirectories from the SGML sources. This option will overwrite any existing file in the `man` directory of the same name as the SGML file.

`-n`
: Do not create (or recreate) the index files. If the `-n` option is specified, the index files are not created and the `apropos(1)` and `whatis(1)` commands might run more slowly than otherwise.

`-p`
: Dry—run option. That is, display what would be done instead of doing it.

`-t`
: Create `troff`ed entries in the appropriate `fmt` subdirectories instead of `nroff`ing into the `cat` subdirectories.

`-w`
Create the index files that are used by `apropos(1)`, `whatis(1)` and the `man(1)` `-f`, `-k`, and `-K` options, in the directories specified by the `MANPATH` environment variable or the `-M` option. If no `MANPATH` or `-M` option is specified, index files are created in the `/usr/share/man/` and `/usr/gnu/share/man/` directories by default. No manual reformatting is done.

`-M` *directory*
Update manual pages located in the specified *directory*, (`/usr/share/man` by default). If the `-M` option is specified, the directory argument must not contain a ',' (comma), since a comma is used to delineate section numbers. See `man(1)`.

`-T` *macro-package*
Use *macro-package* in place of the standard manual page macros, (`man(5)` by default).

Operands The following operand is supported:

sections
If there is one parameter not starting with a '–', it is taken to be a space separated list of manual sections to be processed by `catman`. If this operand is specified, only the manual sections in the list will be processed. For example,

```
catman 1 2 3
```

only updates manual sections `1`, `2`, and `3`. If specific sections are not listed, all sections in the `man` directory specified by the environment variable `MANPATH` are processed.

Environment Variables

`TROFF`
The name of the formatter to use when the `-t` flag is given. If not set, `troff(1)` is used.

`MANPATH`
A colon-separated list of directories that are processed by `catman` and `man(1)`. Each directory can be followed by a comma-separated list of sections. If set, its value overrides `/usr/share/man` as the default directory search path, and the `man.cf` file as the default section search path. The `-M` and `-s` flags, in turn, override these values.

Examples **EXAMPLE 1** Creating an Index File

The following command creates an index file in the `/usr/local/share/man` directory.

```
# catman -M /usr/local/share/man -w
```

Files

`/usr/share/man`
default manual directory location

`/usr/share/man/man*/*.*`
raw nroff input files

`/usr/share/man/sman*/*.*`
raw SGML input files

`/usr/share/man/cat*/*.*`
preformatted `nroff`ed manual pages

`/usr/share/man/fmt*/*.*`
preformatted `troffed` manual pages

`/usr/share/lib/tmac/an`
default macro package

`/usr/share/man/man_index/*`
`/usr/share/man/man_index/man.idx`
`/usr/share/man/man_index/man.dic`
`/usr/share/man/man_index/man.frq`
`/usr/share/man/man_index/man.pos`
index file for `-K` query

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	text/doctools
CSI	Enabled
Interface Stability	Committed

See Also `apropos(1)`, `man(1)`, `nroff(1)`, `rm(1)`, `troff(1)`, `whatis(1)`, `attributes(5)`, `man(5)`, `sgml(5)`

Diagnostics `man?/xxx.? (.so'ed from man?/yyy.?): No such file or directory`
The file outside the parentheses is missing, and is referred to by the file inside them.

`target of .so in man?/xxx.? must be relative to /usr/man`
`catman` only allows references to filenames that are relative to the directory `/usr/man`.

`opendir:man?: No such file or directory`
A harmless warning message indicating that one of the directories `catman` normally looks for is missing.

`*.*: No such file or directory`
A harmless warning message indicating `catman` came across an empty directory.

Warnings If a user, who has previously run `catman` to install the `cat*` directories, upgrades the operating system, the entire `cat*` directory structure should be removed prior to running `catman`. See `rm(1)`.

Do not re-run `catman` to rebuild the index files unless the complete set of `man*` directories is present. `catman` builds the index files based on the `man*` directories.

Notes The `windex` database has been replaced by index files. Unlike the case with `windex`, index file generation does not pose any specific restrictions or prerequisites on what can be indexed.

Name cfgadm – configuration administration

Synopsis

```
/usr/sbin/cfgadm [-f] [-y | -n] [-v] [-o hardware_options]
     -c function ap_id...

/usr/sbin/cfgadm [-f] [-y | -n] [-v] [-o hardware_options]
     -x hardware_function ap_id...

/usr/sbin/cfgadm [-v] [-a] [-s listing_options]
     [-o hardware_options] [-l [ap_id | ap_type]]

/usr/sbin/cfgadm [-v] [-o hardware_options] -t ap_id...

/usr/sbin/cfgadm [-v] [-o hardware_options] -h
     [ap_id | ap_type]
```

Description The `cfgadm` command provides configuration administration operations on dynamically reconfigurable hardware resources. These operations include displaying status, (`-l`), initiating testing, (`-t`), invoking configuration state changes, (`-c`), invoking hardware specific functions, (`-x`), and obtaining configuration administration help messages (`-h`). Configuration administration is performed at *attachment points*, which are places where system software supports dynamic reconfiguration of hardware resources during continued operation of Solaris.

Configuration administration makes a distinction between hardware resources that are physically present in the machine and hardware resources that are configured and visible to Solaris. The nature of configuration administration functions are hardware specific, and are performed by calling hardware specific libraries.

Configuration administration operates on an *attachment point*. Hardware resources located at attachment points can or can not be physically replaceable during system operation, but are dynamically reconfigurable by way of the configuration administration interfaces.

An attachment point defines two unique elements, which are distinct from the hardware resources that exist beyond the attachment point. The two elements of an attachment point are a *receptacle* and an *occupant*. Physical insertion or removal of hardware resources occurs at attachment points and results in a receptacle gaining or losing an occupant. Configuration administration supports the physical insertion and removal operations as well as other configuration administration functions at an attachment point.

Attachment points have associated state and condition information. The configuration administration interfaces provide control for transitioning attachment point states. A receptacle can exist in one of three states: `empty`, `disconnected` or `connected`, while an occupant can exist in one of two states: `configured` or `unconfigured`.

A receptacle can provide the `empty` state, which is the normal state of a receptacle when the attachment point has no occupants. A receptacle can also provide the `disconnected` state if it has the capability of isolating its occupants from normal system access. Typically this state is used for various hardware specific testing prior to bringing the occupant's resources into full

use by the system, or as a step in preparing an occupant for physical removal or reconfiguration. A receptacle in the disconnected state isolates its occupant from the system as much as its hardware allows, but can provide access for testing and setup. A receptacle must provide the `connected` state, which allows normal access to hardware resources contained on any occupants. The connected state is the normal state of a receptacle that contains an occupant and that is not currently undergoing configuration administration operations.

The hardware resources contained on an occupant in the `unconfigured` state are not represented by normal Solaris data structures and are thus not available for use by Solaris. Operations allowed on an unconfigured occupant are limited to configuration administration operations. The hardware resources of an occupant in the `configured` state are represented by normal Solaris data structures and thus some or all of those hardware resources can be in use by Solaris. All occupants provide both the `configured` and `unconfigured` states,

An attachment point can be in one of five conditions: `unknown`, `ok`, `failing`, `failed`, or `unusable`. An attachment point can enter the system in any condition depending upon results of power-on tests and non-volatile record keeping.

An attachment point with an occupant in the `configured` state is in one of four conditions: `unknown`, `ok`, `failing`, or `failed`. If the condition is not `failing` or `failed` an attachment point can change to `failing` during the course of operation if a hardware dependent recoverable error threshold is exceeded. If the condition is not `failed` an attachment point can change to `failed` during operation as a result of an unrecoverable error.

An attachment point with an occupant in the `unconfigured` state can be in any of the defined conditions. The condition of an attachment point with an `unconfigured` occupant can decay from `ok` to `unknown` after a machine dependent time threshold. Initiating a test function changes the attachment point's condition to `ok`, `failing` or `failed` depending on the outcome of the test. An attachment point that does not provide a test function can leave the attachment point in the `unknown` condition. If a test is interrupted, the attachment point's condition can be set to the previous condition, `unknown` or `failed`. An attachment point in the `unknown`, `ok`, `failing`, or `failed` conditions can be re-tested.

An attachment point can exist in the `unusable` condition for a variety of reasons, such as inadequate power or cooling for the receptacle, an occupant that is unidentifiable, unsupported, incorrectly configured, etc. An attachment point in the `unusable` condition can never be used by the system. It typically remains in this condition until the physical cause is remedied.

An attachment point also maintains busy information that indicates when a state change is in progress or the condition is being reevaluated.

Attachment points are referred to using hardware specific identifiers (*ap_id*s) that are related to the type and location of the attachment points in the system device hierarchy. An *ap_id* can not be ambiguous, it must identify a single attachment point. Two types of *ap_id*

specifications are supported: physical and logical. A physical *ap_id* contains a fully specified pathname, while a logical *ap_id* contains a shorthand notation that identifies an attachment point in a more user-friendly way.

For example, an attachment point representing a system's backplane slot number `7` could have a physical *ap_id* of `/devices/central/fhc/sysctrl:slot7` while the logical *ap_id* could be `system:slot7`. Another example, the third receptacle on the second PCI I/O bus on a system could have a logical *ap_id* of `pci2:plug3`.

Attachment points may also be created dynamically. A dynamic attachment point is named relative to a base attachment point which is present in the system. `ap_id`s for dynamic attachment points consist of a base component followed by two colons (`::`) and a dynamic component. The base component is the base attachment point `ap_id`. The dynamic component is hardware specific and generated by the corresponding hardware specific library.

For example, consider a base attachment point, which represents a SCSI HBA, with the physical `ap_id` `/devices/sbus@1f,0/SUNW,fas@e,8800000:scsi` and logical `ap_id` `c0` . A disk attached to this SCSI HBA could be represented by a dynamic attachment point with logical `ap_id` `c0::dsk/c0t0d0` where `c0` is the base component and `dsk/c0t0d0` is the hardware specific dynamic component. Similarly the physical `ap_id` for this dynamic attachment point would be: `/devices/sbus@1f,0/SUNW,fas@e,8800000:scsi::dsk/c0t0d0`

An *ap_type* is a partial form of a logical `ap_id` that can be ambiguous and not specify a particular attachment point. An *ap_type* is a substring of the portion of the logical `ap_id` up to but not including the colon (`:`) separator. For example, an *ap_type* of `pci` would show all attachment points whose logical *ap_id*s begin with `pci`.

The use of *ap_types* is discouraged. The new select sub-option to the `-s` option provides a more general and flexible mechanism for selecting attachment points. See `OPTIONS`.

The `cfgadm` command interacts primarily with hardware dependent functions contained in hardware specific libraries and thus its behavior is hardware dependent.

For each configuration administration operation a service interruption can be required. Should the completion of the function requested require a noticeable service interruption to interactive users, a prompt is output on the standard error output for confirmation on the standard input before the function is started. Confirmation can be overridden using the `-y` or `-n` options to always answer yes or no respectively. Hardware specific options, such as test level, are supplied as sub-options using the `-o` option.

Operations that change the state of the system configuration are audited by the system log daemon `syslogd(1M)`.

The arguments for this command conform to the `getopt(3C)` and `getsubopt(3C)` syntax convention.

Options The following options are supported:

`-a` Specifies that the `-l` option must also list dynamic attachment points.

`-c`*function* Performs the state change *function* on the attachment point specified by *ap_id.*

Specify *function* as `insert`, `remove`, `disconnect`, `connect`, `configure` or `unconfigure`. These functions cause state transitions at the attachment point by calling hardware specific library routines and are defined in the following list.

`insert` Performs operations that allows the user to manually insert an occupant or to activate a hardware supplied mechanism that performs the physical insertion. `insert` can have hardware specific side effects that temporarily suspend activity in portions of the system. In such cases the hardware specific library generates appropriate warning messages and informs the user of any special considerations or procedures unique to that hardware. Various hardware specific errors can cause this function to fail and set the receptacle condition to `unusable`.

`remove` Performs operations that allow the user to manually remove an occupant or to activate a hardware supplied mechanism to perform the physical removal. `remove` can have hardware specific side effects that temporarily suspend activity in portions of the system. In such cases the hardware specific library generates appropriate warning messages and informs the user of any special considerations or procedures unique to that hardware. Various hardware specific errors can cause this function to fail and set the receptacle condition to unusable.

`disconnect` Performs hardware specific operations to put a receptacle in the disconnected state, which can prevent an occupant from operating in a normal fashion through the receptacle.

`connect` Performs hardware specific operations to put the receptacle in the `connected` state, which allows an occupant to operate in a normal fashion through the receptacle.

`configure` Performs hardware specific operations that allow an occupant's hardware resources to be usable by Solaris. Occupants that are configured are part of the system configuration and are available for manipulation by Solaris device manipulation maintenance commands (eg: `psradm`(1M), `mount`(1M), `ifconfig`(1M)).

`unconfigure` Performs hardware specific operations that logically remove an occupant's hardware resources from the system. The occupant must currently be configured and its hardware resources must not be in use by Solaris.

State transition functions can fail due to the condition of the attachment point or other hardware dependent considerations. All state change *function*s in the direction of adding resources, (`insert`, `connect` and `configure`) are passed onto the hardware specific library when the attachment point is in the `ok` or `unknown` condition. All other conditions require the use of the force option to allow these *function*s to be passed on to the hardware specific library. Attachment point condition does not prevent a hardware specific library being called for related to the removal (`remove`, `disconnect` and `unconfigure`), of hardware resources from the system. Hardware specific libraries can reject state change *function*s if the attachment point is in the `unknown` condition.

The condition of an attachment point is not necessarily changed by the state change functions, however errors during state change operations can change the attachment point condition. An attempt to override a condition and force a state change that would otherwise fail can be made by specifying the force option (`-f`). Hardware specific safety and integrity checks can prevent the force option from having any effect.

`-f` Forces the specified action to occur. Typically, this is a hardware dependent override of a safety feature. Forcing a state change operation can allow use of the hardware resources of occupant that is not in the `ok` or `unknown` conditions, at the discretion of any hardware dependent safety checks.

`-h` [*ap_id* | *ap_type* . . .]

Prints out the help message text. If *ap_id* or *ap_type* is specified, the help routine of the hardware specific library for the attachment point indicated by the argument is called.

`-l` [*ap_id* | *ap_type* . . .]

Lists the state and condition of attachment points specified. Attachment points can be filtered by using the `-s` option and `select` sub-option. Invoking `cfgadm` without one of the action options is equivalent to `-l` without an argument. The format of the list display is controlled by the `-v` and `-s` options. When the `-a` option is specified attachment points are dynamically expanded.

`-n`

Suppress any interactive confirmation and assume that the answer is *no*. If neither `-n` or `-y` is specified, interactive confirmation is obtained through the standard error output and the standard input. If either of these standard channels does not correspond to a terminal (as determined by `isatty(3C)`) then the `-n` option is assumed.

`-o`*hardware_options*

Supplies hardware specific options to the main command option. The format and content of the hardware option string is completely hardware specific. The option string *hardware_options* conforms to the `getsubopt(3C)` syntax convention.

`-s`*listing_options*

Supplies listing options to the list (`-l`) command. *listing_options* conforms to the `getsubopt(3C)` syntax convention. The sub-options are used to specify the attachment point selection criteria (`select=`*select_string*), the type of matching desired (`match=`*match_type*), order of listing (`sort=`*field_spec*), the data that is displayed (`cols=`*field_spec* and `cols2=`*field_spec*), the column delimiter (`delim=`*string*) and whether to suppress column headings (`noheadings`).

When the `select` sub-option is specified, only attachment points which match the specified criteria will be listed. The `select` sub-option has the following syntax:

```
cfgadm -s select=attr1(value1):attr2(value2)...
```

where an *attr* is one of `ap_id`, `class` or `type`. `ap_id` refers to the logical *ap_id* field, `class` refers to attachment point class and `type` refers to the type field. *value1*, *value2*, etc. are the corresponding values to be matched. The type of match can be specified by the `match` sub-option as follows:

```
cfgadm -s match=match_type,select=attr1(value1)...
```

where *match_type* can be either `exact` or `partial`. The default value is `exact`.

Arguments to the `select` sub-option can be quoted to protect them from the shell.

A *field_spec* is one or more *data-field*s concatenated using colon (`:`), as in *data-field*:*data-field*:*data-field*. A *data-field* is one of `ap_id`, `physid`, `r_state`, `o_state`, `condition`, `type`, `busy`, `status_time`, `status_time_p`, `class`, and `info`. The `ap_id` field output is the logical name for the attachment point, while the `physid` field contains the physical name. The `r_state` field can be `empty`, `disconnected` or `connected`. The `o_state` field can be `configured` or `unconfigured`. The `busy` field can be either `y` if the attachment point is busy, or `n` if it is not. The `type` and `info` fields are hardware specific. The `status_time` field provides the time at which either the `r_state`, `o_state`, or condition of the attachment point last changed. The `status_time_p` field is a parsable version of the `status_time` field. If an attachment point has an associated class, the `class` field lists the class name. If an attachment point does not have an associated class, the `class` field lists `none`.

The order of the fields in *field_spec* is significant: For the `sort` sub-option, the first field given is the primary sort key. For the `cols` and `cols2` sub-options, the fields are printed in the order requested. The order of sorting on a *data-field* can be reversed by placing a minus (–) before the *data-field* name within the *field_sec* for the `sort` sub-option. The default value for `sort` is `ap_id`. The defaults values for `cols` and `cols2` depend on whether the `-v` option is given: Without it `cols` is `ap_id:r_state:o_state:condition` and `cols2` is not set. With `-v` `cols` is `ap_id:r_state:o_state:condition:info` and `cols2` is `status_time:type:busy:physid:`. The default value for `delim` is a single space. The value of `delim` can be a string of arbitrary length. The delimiter cannot include comma (`,`) character, see `getsubopt(3C)`. These listing options can be used to create parsable output. See `NOTES`.

`-t`

Performs a test of one or more attachment points. The test function is used to re-evaluate the condition of the attachment point. Without a test level specifier in *hardware_options*, the fastest test that identifies hard faults is used.

More comprehensive tests are hardware specific and are selected using the *hardware_options*.

The results of the test is used to update the condition of the specified occupant to either `ok` if no faults are found, `failing` if recoverable faults are found or `failed` if any unrecoverable faults are found.

If a test is interrupted, the attachment point's condition can be restored to its previous value or set to `unknown` if no errors were found or `failing` if only recoverable errors were found or to `failed` if any unrecoverable errors were found. The attachment point should only be set to `ok` upon normal completion of testing with no errors.

`-v` Executes in verbose mode. For the `-c`, `-t` and `-x` options outputs a message giving the results of each attempted operation. Outputs detailed help information for the `-h` option. Outputs verbose information for each attachment point for the `-l` option.

`-x`*hardware_function* Performs hardware specific functions. Private hardware specific functions can change the state of a receptacle or occupant. Attachment point conditions can change as the result of errors encountered during private hardware specific functions. The format and content of the *hardware_function* string is completely hardware specific. The option string *hardware_function* conforms to the `getsubopt(3C)` syntax convention.

`-y` Suppresses any interactive confirmation and assume that the answer is `yes`.

Usage

The required privileges to use this command are hardware dependent. Typically, a default system configuration restricts all but the list option to the superuser.

Examples

EXAMPLE 1 Listing Attachment Points in the Device Tree

The following example lists all attachment points except dynamic attachment points.

```
example# cfgadm

  Ap_Id         Type        Receptacle      Occupant       Cond
system:slot0    cpu/mem     connected       configured     ok
system:slot1    sbus-upa    connected       configured     ok
system:slot2    cpu/mem     connected       configured     ok
system:slot3    unknown     connected       unconfigured   unknown
system:slot4    dual-sbus   connected       configured     failing
system:slot5    cpu/mem     connected       configured     ok
system:slot6    unknown     disconnected    unconfigured   unusable
```

EXAMPLE 1 Listing Attachment Points in the Device Tree *(Continued)*

```
system:slot7    unknown     empty           unconfigured   ok
c0              scsi-bus    connected       configured     unknown
c1              scsi-bus    connected       configured     unknown
```

EXAMPLE 2 Listing All Configurable Hardware Information

The following example lists all current configurable hardware information, including those represented by dynamic attachment points:

```
example# cfgadm -al

  Ap_Id            Type         Receptacle      Occupant        Cond
system:slot0       cpu/mem      connected       configured      ok
system:slot1       sbus-upa     connected       configured      ok
system:slot2       cpu/mem      connected       configured      ok
system:slot3       unknown      connected       unconfigured    unknown
system:slot4       dual-sbus    connected       configured      failing
system:slot5       cpu/mem      connected       configured      ok
system:slot6       unknown      disconnected    unconfigured    unusable
system:slot7       unknown      empty           unconfigured    ok
c0                 scsi-bus     connected       configured      unknown
c0::dsk/c0t14d0    disk         connected       configured      unknown
c0::dsk/c0t11d0    disk         connected       configured      unknown
c0::dsk/c0t8d0     disk         connected       configured      unknown
c0::rmt/0          tape         connected       configured      unknown
c1                 scsi-bus     connected       configured      unknown
```

EXAMPLE 3 Listing Selectively, Based on Attachment Point Attributes

The following example lists all attachment points whose class begins with `scsi`, `ap_id` begins with `c` and `type` field begins with `scsi`. The argument to the `-s` option is quoted to protect it from the shell.

```
example# cfgadm -s "match=partial,select=class(scsi):ap_id(c):type(scsi)"

Ap_Id          Type          Receptacle      Occupant          Cond
 c0          scsi-bus      connected       configured        unknown
 c1          scsi-bus      connected       configured        unknown
```

EXAMPLE 4 Listing Current Configurable Hardware Information in Verbose Mode

The following example lists current configurable hardware information for *ap-type* `system` in verbose mode:

```
example# cfgadm -v -l system
Ap_Id                     Receptacle Occupant   Condition Information
When         Type      Busy     Phys_Id
```

EXAMPLE 4 Listing Current Configurable Hardware Information in Verbose Mode *(Continued)*

```
system:slot1              connected  configured ok
Apr  4 23:50 sbus-upa  n        /devices/central/fhc/sysctrl:slot1
system:slot3              connected  configured ok        non-detachable
Apr 17 11:20 cpu/mem   n        /devices/central/fhc/sysctrl:slot3
system:slot5              connected  configured ok
Apr  4 23:50 cpu/mem   n        /devices/central/fhc/sysctrl:slot5
system:slot7              connected  configured ok
Apr  4 23:50 dual-sbus n        /devices/central/fhc/sysctrl:slot7
```

The `When` column represents the `status_time` field.

EXAMPLE 5 Testing Two Occupants Using the Hardware Specific Extended Test

The following example tests two occupants using the hardware specific `extended` test:

```
example# cfgadm -v -o extended -t system:slot3 system:slot5
Testing attachment point system:slot3 ...  ok
Testing attachment point system:slot5 ...  ok
```

EXAMPLE 6 Configuring an Occupant Using the Force Option

The following example configures an occupant in the `failing` state to the system using the force option:

```
example# cfgadm -f -c configure system:slot3
```

EXAMPLE 7 Unconfiguring an Occupant From the System

The following example unconfigures an occupant from the system:

```
example# cfgadm -c unconfigure system:slot4
```

EXAMPLE 8 Configuring an Occupant at an Attachment Point

The following example configures an occupant:

```
example# cfgadm -c configure c0::dsk/c0t0d0
```

Environment Variables

See `environ(5)` for descriptions of the following environment variables that affect the execution of `cfgadm`: `LC_TIME`, `LC_MESSAGES`, `NLSPATH` and `TZ`.

`LC_MESSAGES` Determines how `cfgadm` displays column headings and error messages. Listing output data is not affected by the setting of this variable.

`LC_TIME` Determines how `cfgadm` displays human readable status changed time (`status_time`).

`TZ` Specifies the timezone used when converting the status changed time. This applies to both the human readable (`status_time`) and parsable

(`status_time_p`) formats.

Exit Status The following exit values are returned:

`0` Successful completion.

`1` An error occurred.

`2` Configuration administration not supported on specified target.

`3` Usage error.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also `cfgadm_fp(1M)`, `cfgadm_ib(1M)`, `cfgadm_pci(1M)`, `cfgadm_sata(1M)`, `cfgadm_sbd(1M)`, `cfgadm_scsi(1M)`, `cfgadm_usb(1M)`, `ifconfig(1M)`, `mount(1M)`, `prtdiag(1M)`, `psradm(1M)`, `syslogd(1M)`, `config_admin(3CFGADM)`, `getopt(3C)`, `getsubopt(3C)`, `isatty(3C)`, `attributes(5)`, `environ(5)`

Diagnostics Diagnostic messages appear on the standard error output. Other than options and usage errors, the following are diagnostic messages produced by this utility:

`cfgadm: Configuration administration not supported on`*`ap_id`*

`cfgadm: No library found for` *`ap_id`*

`cfgadm:` *`ap_id`*`is ambiguous`

`cfgadm:` *`operation`*`: Insufficient privileges`

`cfgadm: Attachment point is busy, try again`

`cfgadm: No attachment points with specified attributes found`

`cfgadm: System is busy, try again`

`cfgadm:` *`operation`*`: Operation requires a service interruption`

`cfgadm:` *`operation`*`: Data error:` *`error_text`*

`cfgadm:` *`operation`*`: Hardware specific failure:` *`error_text`*

See `config_admin(3CFGADM)` for additional details regarding error messages.

Notes Hardware resources enter the unconfigured pool in a hardware specific manner. This can occur at various times such as: system initialization or as a result of an unconfigure operation. An occupant that is in the `unconfigured` state is not available for use by the system until specific intervention occurs. This intervention can be manifested as an operator initiated command or it can be by way of an automatic configuring mechanism.

The listing option of the `cfgadm` command can be used to provide parsable input for another command, for example within a shell script. For parsable output, the `-s` option must be used to select the fields required. The `-s` option can also be used to suppress the column headings. The following fields always produce parsable output: `ap_id`, `physid`, `r_state`, `o_state`, `condition`, `busy` `status_time_p`, `class`, and `type`. Parsable output never has white-space characters embedded in the field value.

The following shell script fragment finds the first good `unconfigured` occupant of type `CPU`.

```
found=
cfgadm -l -s "noheadings,cols=ap_id:r_state:condition:type" | \
while read ap_id r_state cond type
do
     if [ "$r_state" = unconfigured -a "$cond" = ok -a "$type" = CPU ]
     then
          if [ -z "$found" ]
          then
               found=$ap_id
          fi
     fi
done
if [ -n "$found" ]
then
         echo "Found CPU $found"
fi
```

The format of the parsable time field (`status_time_p`) is *YYYYMMDDhhmmss*, giving the year, month, day, hour, minute and second in a form suitable for string comparison.

Reference should be made to the hardware specific documentation for details of System Configuration Administration support.

Name cfgadm_ac – EXX00 memory system administration

Synopsis

```
/usr/sbin/cfgadm [-c configure] [-f]
     [-o disable-at-boot | enable-at-boot ] ac#:bank# ...

/usr/sbin/cfgadm [-c unconfigure]
     [-o disable-at-bootp | enable-at-boot ] ac#:bank# ...

/usr/sbin/cfgadm [-v]
     [-o quick | normal | extended, [max_errors=#] ] -t ac#:bank#...

/usr/sbin/cfgadm -x relocate-test ac#:bank# ...

/usr/sbin/cfgadm [-l] -o disable-at-boot | enable-at-boot ac#:bank# ...
```

Description The ac hardware specific library `/usr/platform/sun4u/lib/cfgadm/cfgadm_ac.so.1` provides the functionality for configuring and unconfiguring memory banks on `E6X00`, `E5X00`, `E4X00` and `E3X00` systems as part of the Dynamic Reconfiguration of CPU/Memory boards using `cfgadm_sysctrl(1M)`.

Memory banks appear as attachment points in the device tree. For each CPU/Memory board, two attachment points are published, one for each bank on the board: `bank0` and `bank1`. If the bank is unpopulated, the receptacle state is empty. If the bank is populated, the receptacle state is connected. The receptacle state of a memory bank can never be disconnected. The occupant state of a connected memory bank can be configured or unconfigured. If the occupant state is configured, the memory is in use by Solaris, if unconfigured it is not.

Options Refer to `cfgadm`(1M) for complete descriptions of the command options.

The following options are supported:

`-c` configure | unconfigure

Change the occupant state. The `configure` argument ensures that the memory is initialized and adds the memory to the Solaris memory pool. The `unconfigure` argument removes the memory from use by Solaris. When a CPU/Memory board is to be removed from a system, both banks of memory must be unconfigured.

`cfgadm` refuses the `configure` operation if the memory on the board is marked `disabled-at-boot` (see `info` field), unless either the `-f` (force) option or the enable at boot flag, (`-o enable-at-boot`), is given. The configure operation takes a short time proportional to the size of memory that must be initialized.

`cfgadm` refuses the `unconfigure` operation if there is not enough uncommitted memory in the system (`VM viability` error) or if the bank to be unconfigured has memory that can't be removed (`non-relocatable`

pages error). The presence of non-relocatable pages is indicated by the word permanent in the info listing field. Removing memory from use by Solaris may take a significant time due to factors such as system load and how much paging to secondary storage is required. The unconfigure operation can be cancelled at any time and the memory returned to the fully configured state by interrupting the command invocation with a signal. The unconfigure operation self-cancels if no memory can be removed within a timeout period. The default timeout period of 60 seconds can be changed using the -o timeout=# option, with a value of 0 disabling the timeout.

`-f` Force option. Use this option to override the block on configuring a memory bank marked as disabled at boot in the non-volatile disabled-memory-list variable. See *Platform Notes:Sun Enterprise 6x00/5x00/4x00/3x00 Systems*

`-l` List option. This option is supported as described in cfgadm(1M).

The type field is always *memory*.

The *info* field has the following information for empty banks:

```
slot# empty
```

The slot# indicates the system slot into which the CPU/Memory board is inserted. For example, if this were slot11 the attachment point for use with cfgadm to manipulate the associated board would be sysctrl0:slot11. The info field has the following information for connected banks:

```
slot# sizeMb|sizeGb [(sizeMb|sizeGb used)] base 0x###
      [interleaved #-way] [disabled at boot] [permanent]
```

The size of the bank is given in Mb or Gb as appropriate. If the memory is less than completely used, the used size is reported. The physical base address is given in hexadecimal. If the memory bank is interleaved with some other bank, the interleave factor is reported. If the memory on the board is disabled at boot using the non-volatile disabled-memory-list

	variable, this is reported. If the bank has memory that cannot be removed this is reported as permanent.
`-o` disable-at-boot | enable-at-boot	These options allow the state of the non-volatile disabled-memory-list variable to be modified. These options can be used in conjunction with the issuing of a `-c` option or with the explicit or implied listing command, -l, if no command is required. Use of `-o enable-at-boot` with the `configure` command to override the block on configuring memory on a board in the disabled memory list.
`-o` extended | normal | quick	Use with the `-t` option to specify test level. The `normal` test level ensures that each memory cell stores both a `0` and a `1`, and checks that all cells are separately addressable. The `quick` test level only does the `0`s and `1`s test, and typically misses address line problems. The `extended` test uses patterns to test for adjacent cell interference problems. The default test level is `normal`. See `-t` option.
`-o` max_errors=#	Use with the `-t` option to specify the maximum number of allowed errors. If not specified, a default of `32` is assumed.
`-o` timeout=#	Use with the `unconfigure` command to set the self-cancelling timeout. The default value is `60` and the unit is seconds. A value of `0` means no timeout.
`-t`	Test an unconfigured bank of memory. Specify the test level using the `-o quick` | `normal` | `extended` option. `cfgadm` exits with a `0` (success) if the *test* was able to run on the memory bank. The result of the test is available in the condition for the attachment point.
`-v`	Verbose option. Use this option in combination with the `-t` option to display detailed progress and results of tests.
`-x` relocate-test	For all pages of memory in use on the specified memory bank, a relocation operation as used in the unconfigure command is attempted. The success of this operation does not guarantee that the bank can be unconfigured. Failure indicates that it probably cannot be unconfigured. This option is for test purposes only.

Operands The following operand is supported:

ac#:bank# The attachment points for memory banks are published by instances of the address controller (ac) driver (*ac#*). One instance of the ac driver is created for each system board, but only those instances associated with CPU/Memory boards publish the two bank attachment points, bank0 and bank1.

This form conforms to the logical `ap_id` specification given in `cfgadm`(1M). The corresponding physical `ap_id`s are listed in the `FILES` section.

The ac driver instance numbering has no relation to the slot number for the corresponding board. The full physical attachment point identifier has the slot number incorporated into it as twice the slot number in hexadecimal directly following the `fhc@` part.

Files

/devices/fhc@*,f8800000/ac@0,1000000:bank?	attachment points
/usr/platform/sun4u/lib/cfgadm/cfgadm_ac.so.1	hardware specific library file

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/library/platform

See Also `cfgadm(1M)`, `cfgadm_sysctrl(1M)`, `config_admin(3CFGADM)`, `attributes(5)`

Sun Enterprise 6x00, 5x00, 4x00 and 3x00 Systems Dynamic Reconfiguration User's Guide

Platform Notes:Sun Enterprise 6x00/5x00/4x00/3x00 Systems

Notes Refer to the *Sun Enterprise 6x00, 5x00, 4x00 and 3x00 Systems Dynamic Reconfiguration User's Guide* for additional details regarding dynamic reconfiguration of EXX00 system CPU/Memory boards.

Name cfgadm_cardbus – cardbus hardware specific commands for cfgadm

Synopsis

```
/usr/sbin/cfgadm [-f ] [-y  | -n ] [-v]
     [-o hardware_options] -c function ap_id [ap_id]

/usr/sbin/cfgadm [-f ] [-y  | -n ] [-v]
     [-o hardware_options] -x hardware_function ap_id
     [ap_id]

/usr/sbin/cfgadm [-v] [-s listing_options]
     [-o hardware_options] [-l [ap_id | ap_type]]

/usr/sbin/cfgadm [-v] [-o harware_options] -t ap_id [ap_id]

/usr/sbin/cfgadm [-v] [-o hardware_function] -h
     [ap_id| ap_type]
```

Description The CardBus slots in Solaris are hot plug capable. This capability is supported by the PCI hardware specific library `/usr/lib/cfgadm/pci.so.1` through the `cfgadm` command (see `cfgadm(1M)`).

The hot plug administrative models between CardBus, PCI, CompactPCI, and PCI Express operate the same fashion. Please refer to `cfgadm_pci(1M)` for the usage information.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/library

See Also `cfgadm(1M)`, `config_admin(3CFGADM)`, `libcfgadm(3LIB)`, `attributes(5)`

Oracle Solaris Administration: Common Tasks

Name cfgadm_fp – driver specific commands for cfgadm

Synopsis

```
/usr/sbin/cfgadm [-f] [-n | -y] [-v] [-o hardware_options]
     -c function ap_id [ap_id]

/usr/sbin/cfgadm [-v] [-a] [-s listing_options]
     [-o hardware_options] [-l [ap_id]]

/usr/sbin/cfgadm [-v] [-o hardware_options] -h [ap_id]
```

Description The `fp` port driver plug-in `/usr/lib/cfgadm/fp.so.1` provides the functionality for Fibre Channel Fabric device node management through `cfgadm(1M)`. `cfgadm` operates on attachment points. Attachment points are locations in the system where hardware resources can be dynamically reconfigured. Refer to `cfgadm(1M)` for additional details on attachment points.

For Fibre Channel Fabric device node management, each `fp` port node is represented by an attachment point in the device tree. In addition, each Fibre Channel device is represented by a dynamic attachment point. Attachment points are named through `ap_ids`. Two types of `ap_ids` are defined: logical and physical. The physical `ap_id` is based on the physical pathname. The logical `ap_id` is a shorter, more user-friendly name. For `fp` port nodes, the logical `ap_id` is the corresponding disk controller number. For example, `c0` is a typical logical `ap_id`.

Fibre Channel devices are named with a port World Wide Name (WWN). If a disk device is connected to controller `c0`, its `ap_id` can be:

```
c0::50020f2300006077
```

where `50020f2300006077` identifies the port WWN of a specific Fibre Channel device.

Each device on the Fibre Channel private loop port, Fabric port or public loop port is probed and made available to Solaris by default. Devices connected to the Fibre Channel Fabric port or public loop port can be made unavailable to Solaris by initiating an application or an end user operation. The operation is similar to the hot unplugging of devices by way of management user interfaces. Applications or users can use the `/usr/lib/cfgadm/fp.so.1` library to enable `libcfgadm` to provide interfaces to accomplish this task.

The list of currently connected Fabric devices is generated in the form of the attachment point.

A simple listing of attachment points in the system includes attachment points at `fp` port nodes but not Fibre Channel devices. The following example uses the `-a` flag to the list option (`-l`) to list Fibre Channel devices:

```
# cfgadm -l
     Ap_Id                Type         Receptacle   Occupant     Condition
     c0                   fc-fabric    connected    configured   unknown
     c1                   fc-private   connected    configured   unknown
     c2                   fc-pt_to_pt  connected    configured   unknown
     c3                   fc           connected    unconfigured unknown
```

```
     sysctrl0:slot0       cpu/mem      connected    configured   ok
     sysctrl0:slot1       sbus-upa     connected    configured   ok
```

The following example lists Fibre Channel devices connected to `fp` ports.

```
# cfgadm -al
     Ap_Id                Type         Receptacle   Occupant     Condition
     c0                   fc-fabric    connected    configured   unknown
     c0::50020f2300006077 disk         connected    configured   unknown
     c0::50020f23000063a9 disk         connected    configured   unknown
     c0::50020f2300005f24 disk         connected    configured   unknown
     c0::50020f2300006107 disk         connected    configured   unknown
     c1                   fc-private   connected    configured   unknown
     c1::220000203708b69c disk         connected    configured   unknown
     c1::220000203708ba7d disk         connected    configured   unknown
     c1::220000203708b8d4 disk         connected    configured   unknown
     c1::220000203708b9b2 disk         connected    configured   unknown
     c2                   fc-pt_to_pt  connected    configured   unknown
     c2::500104f000937528 tape         connected    configured   unknown
     c3                   fc           connected    unconfigured unknown
     sysctrl0:slot0       cpu/mem      connected    configured   ok
     sysctrl0:slot1       sbus-upa     connected    configured   ok
```

In this example, the `fc-fabric` type of `ap_id` `c0` indicates that the `fp` port is connected to Fabric. For an `fp` port with a Fabric-related type such as `fc-fabric` and `fc-public`, device node creation happens by default at the boot time and can be managed by the `cfgadm` configure and unconfigure operations. The `fc-private` type of `ap_id` `c1` indicates that `fp` port is connected to private-loop and device node creation happens by default as well. The `fc-pt_to_pt` type of `ap_id` `c2` indicates that the `fp` port is directly connected to another `N_port` and device node creation also happens by default. The `fc` type of `ap_id` `c3` indicates that nothing is attached to `fp` port `c2`. The Type field of a Fibre Channel device `ap_id` shows the SCSI device type of LUN 0 in the device.

A Fibre Channel device with multiple FCP SCSI LUNs is configured into Solaris and each `FCP SCSI LUN` is available as a Solaris device. Suppose that ap_ids `c0::50020f2300006077` and `c0::50020f23000063a9` represent Fibre Channel devices with multiple `FCP SCSI LUN`s.

The following example shows how to list ap_ids with `FCP SCSI LUN` information:

```
# cfgadm -al -o show_SCSI_LUN
     Ap_Id                  Type         Receptacle   Occupant     Condition
     c0                     fc-fabric    connected    configured   unknown
     c0::50020f2300006077,0 disk         connected    configured   unknown
     c0::50020f2300006077,1 disk         connected    configured   unknown
     c0::50020f2300006077,2 disk         connected    configured   unknown
     c0::50020f2300006077,3 disk         connected    configured   unknown
     c0::50020f23000063a9,0 disk         connected    configured   unknown
     c0::50020f23000063a9,1 disk         connected    configured   unknown
     c0::50020f23000063a9,2 disk         connected    configured   unknown
```

```
c0::50020f23000063a9,3 disk         connected    configured   unknown
c0::50020f2300005f24,0 disk         connected    unconfigured unknown
c0::50020f2300005f24,1 disk         connected    unconfigured unknown
c0::50020f2300006107,0 disk         connected    unconfigured unknown
c0::50020f2300006107,1 disk         connected    unconfigured unknown
c1                     fc-private   connected    configured   unknown
c1::220000203708b69c,0 disk         connected    configured   unknown
c1::220000203708ba7d,0 disk         connected    configured   unknown
c1::220000203708b8d4,0 disk         connected    configured   unknown
c1::220000203708b9b2,0 disk         connected    configured   unknown
c2                     fc-pt_to_pt  connected    configured   unknown
c2::500104f000937528,0 tape         connected    configured   unknown
c3                     fc           connected    unconfigured unknown
```

In this example, the `ap_id` `c0::50020f2300006077,0` identifies the `FCP SCSI LUN` `0` of the Fibre Channel device which is represented by port `WWN` `50020f2300006077`. The Fibre Channel device is reported to have 4 `FCP SCSI LUN`s and they are all configured. `4 FCP SCSI LUN` level ap_ids associated with port `WWN` `50020f2300006077` are listed. The listing also displays `FCP SCSI LUN`s for unconfigured Fibre Channel devices. The Fibre Channel device represented by `c0::50020f2300005f24` is reported to have two `FCP SCSI LUN`s. The configure operation on `c0::50020f2300005f24` creates two Solaris devices. The `Type` field of `FCP SCSI LUN` level ap_ids show the `SCSI` device type of each `LUN`. When a Fibre Channel device has different device type `LUN`s, the `Type` field reflects that.

The receptacle and occupant state for attachment points at the `fp` port have the following meanings:

configured
: One or more devices configured on the `fp` port

connected
: `fp` port active

disconnected
: `fp` port quiesced (IO activity is suspended)

empty
: Not applicable

unconfigured
: No devices configured on the `fp` port

The state for individual Fibre Channel devices on an `fp` port:

configured
: Device is configured into Solaris and is available for use

connected
: `fp` port to which the device is connected to is active

disconnected
`fp` port to which the device is attached is quiesced

unconfigured
Device is available to be configured

The `condition` field for attachment points at the `fp` port has the following meanings:

failed
An error condition has prevented the `fp` port from being able to detect the presence or type of a Fibre Channel connection.

The `condition` field for individual Fibre Channel devices on an `fp` port has the following meanings:

failed
An error is encountered while probing a device on Fabric.

failing
A device was configured on a host and its state as seen by Solaris appears to be normal (i.e., online) but it is either not currently present or visible in the fabric or its presence could not be verified due to an error condition on the local port through which the device was configured.

unusable
A device has been configured on the host, but is currently offline or failed.

The unknown `condition` indicates that probing a device on Fabric completed without an error and the device state within Solaris host is normal if the device was configured previously. The internal condition of the device cannot be guaranteed.

Options `cfgadm` defines several types of operations in addition to listing (`-l`). These operations include invoking configuration state changes and obtaining configuration administration help messages (`-h`).

The following options are supported:

`-c` *function*
The following generic commands are defined for the `fp`-transport-specific library:

For Fibre Channel device attachment points on the `fc-fabric` type `fp` port attachment point, the following configuration state change operations are supported:

`configure`
Configure a connected Fibre Channel Fabric device to a host. When a Fibre Channel device is listed as an unknown type in the output of the list operation the device might not be configurable. No attempt is made to configure devices with unknown types. The force option (`-f`) can be used to force the `fp` port driver plug-in to make an attempt to configure any devices. Any errors in the process are reported. By default, each FCP SCSI

LUN that is discovered on a Fibre channel Fabric device is configured. However, FCP SCSI LUNs that are specified in the "pwwn-lun-blacklist" property in the `fp.conf` file will remain unconfigured. The FCP SCSI LUN level listing reflects the state of such FCP SCSI LUNs. They stay in the "unconfigured" state after reboot or Solaris Dynamic Reconfiguration on the controller that they are connected through. Refer to `fp(7d)` for additional details on the "pwwn-lun-blacklist" property.

`unconfigure`
Unconfigure a Fibre Channel Fabric device from a host. This device stays unconfigured until the next reboot or Solaris Dynamic Reconfiguration on the controller that the device is connected, at which time all fabric devices are automatically enumerated. The default behavior may be changed through the use of the "manual_configuration_only" property in the `fp.conf` file. If the property is set, the device remains unconfigured after reboot. Refer to `fp(7d)` for additional details on the "manual_configuration_only" property.

For Fibre Channel private loop devices and `N_Port` point-to-point devices, the `configure` command returns success without doing any operation. The `unconfigure` command is not supported on the private loop devices and `N_Port` point-to-point devices. The private loop devices and `N_Port` point-to-point devices are configured by Solaris Fibre Channel drivers by default and are not managed through end user- or application-initiated operations. The `pwwn-lun-blacklist` property in the `fp.conf` file is applied to the private loop device and `N_Port` point-to-point device in the same way it is applied to a Fabric device.

`-f`
Force the `configure` change state operation to occur irrespective of the `condition` or `type`. Refer to the above description of the `configure` change state operation.

`-h` *ap_id*
Obtain `fp`—transport-specific help. Specify any `fp` attachment point.

`-o` *hardware_options*
The following hardware options are supported.

show_SCSI_LUN
Lists ap_ids associated with each FCP SCSI LUN for discovered Fibre Channel devices when specified with the `list` option `-al`. Refer to the previously mentioned description and example of FCP SCSI LUN level listing. Device node creation is not supported on the FCP SCSI LUN level. See `NOTES`.

All Fibre Channel devices are available to Solaris by default. Enabling only a subset of Fabric devices available to Solaris by default can be accomplished by setting the property "manual_configuration_only" in `/kernel/drv/fp.conf` file. When "manual_configuration_only" in `fp.conf` is set, all Fabric devices are not available to Solaris unless an application or an end user had previously requested the device be configured into Solaris. The `configure` state-change command makes the device available

to Solaris. After a successful `configure` operation on a Fabric device, the associated links are added to the `/dev` namespace. The `unconfigure` state-change command makes a device unavailable to Solaris.

When a Fibre Channel Fabric device is configured successfully to a host using the `-c configure` operation, its physical `ap_id` is stored in a repository. When a Fibre Channel Fabric device is unconfigured using the `-c unconfigure` operation, its physical `ap_id` is deleted from the same repository. All fabric devices are automatically enumerated by default and the repository is used only if the `fp.conf` "manual_configuration_only" property is set. Refer to `fp(7d)` for additional details on the "manual_configuration_only" property.

You can specify the following commands with the `-c` option to control the update behavior of the repository:

`force_update`
: For `configure`, the attachment point is unconditionally added to the repository; for `unconfigure`, the attachment point is unconditionally deleted.

`no_update`
: No update is made to the repository regardless of the operation.

These options should not be used for normal `configure` and `unconfigure` operations. See `WARNINGS`.

When a Fibre Channel device has multiple `FCP SCSI LUN`s configured and any Solaris device associated with its `FCP SCSI LUN` is in the unusable condition, the whole Fibre Channel device is reported as unusable. The following option with the `-c` unconfigure command removes only Solaris devices with the unusable condition for a Fibre Channel device.

`unusable_SCSI_LUN`
: For `unconfigure` operation, any offlined device nodes for a target device is removed.

`-s` *listing_options*
: Refer to `cfgadm(1M)` for usage information.

`-t` *ap_id*
: No test commands are available at present.

`-x` *hardware_function*
: No hardware specific functions are available at present.

All other options have the same meaning as defined in the `cfgadm(1M)` man page.

Examples

EXAMPLE 1 Unconfiguring a Disk

The following command unconfigures a disk:

```
# cfgadm -c unconfigure c0::210000203708b606
```

EXAMPLE 2 Unconfigure all the Configured Disks under Single Attachment Point

The following command unconfigures all configured disks under the attachment point `c0`.

```
# cfgadm -c unconfigure c0
```

EXAMPLE 3 Configuring a Disk

The following command configures a disk:

```
# cfgadm -c configure c0::210000203708b606
```

EXAMPLE 4 Configure all the Unconfigured Disks under Single Attachment Point

The following command configures all unconfigured disks under the attachment point `c0`.

```
# cfgadm -c configure c0
```

EXAMPLE 5 Removing the Fibre Channel Fabric Device Attachment Point from Repository

The following command unconditionally removes the fibre channel fabric device attachment point from the Fabric device repository.

```
# cfgadm -c unconfigure -o force_update c0::210000203708b606
```

EXAMPLE 6 Removing Offlined Solaris Device Nodes for a Target Device

The following command removes offlined Solaris device nodes for a target device:

```
# cfgadm -c unconfigure -o unusable_SCSI_LUN c0::210000203708b606
```

Files

`/usr/lib/cfgadm/fp.so.1`
: Hardware-specific library for Fibre Channel Fabric device node management.

`/etc/cfg/fp/fabric_WWN_map`
: Repository of physical `ap_id`s of Fabric devices currently configured. It is used only to reconfigure those Fabric devices at boot time. This repository is only used when the "manual_configuration_only" `/kernel/drv/fp.conf` file is set.

`/etc/rcS.d/fdevattach`
: Reconfigures Fabric device(s) of which physical `ap_id` is listed in `/etc/cfg/fp/fabric_WWN_map` on boot time.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	service/storage/fibre-channel/fc-fabric, service/storage/fibre-channel/fc-fabricx

See Also svcs(1), cfgadm(1M), svcadm(1M), config_admin(3CFGADM), libcfgadm(3LIB), attributes(5), smf(5), fp(7d)

Warnings Do not use hardware-specific options for the repository update under normal configure/unconfigure operations. The hardware-specific options are expected to be used when the node creation of a Fabric device fails at boot time and the error condition is considered to be permanent. The unconfigure command with force_update hardware-specific option unconditionally removes the attachment point of a failing Fabric device from the repository.

Notes For devices with unknown or no SCSI device type (for example, a Fibre Channel Host Bus Adapter), the configure operation might not be applicable.

The configure and unconfigure commands operate on the Fibre Channel device level which is represented by port WWN ap_id. If a Fibre Channel device has multiple FCP SCSI LUNs configured, the configure command on the associated port WWN ap_id results in creating a Solaris device for each FCP SCSI LUN unless it is specified in the "pwwn-lun-blacklist" property in the fp.conf file. The unconfigure command removes all Solaris devices associated with the port WWN ap_id. The FCP SCSI LUN level ap_id is not valid for the configure and unconfigure commands.

The deprecated show_FCP_dev option has been replaced by the new show_SCSI_LUN option, and the deprecated unusable_FCP_dev option has been replaced by the new unusable_SCSI_LUN option.

The cfgadm_fp service is managed by the service management facility, smf(5), under the service identifier:

```
svc:/system/device/fc-fabric:default
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using svcadm(1M). The service's status can be queried using the svcs(1) command.

No administrative actions on this service are required for Fabric device configuration once this service is started on boot time.

Name cfgadm_ib – InfiniBand hardware specific commands for cfgadm

Synopsis

```
/usr/sbin/cfgadm -f [-y | -n] [-v] -c function ap_id...

/usr/sbin/cfgadm [-f] [-y | -n] [-v] [-o hardware_options]
     -x hardware_function ap_id...

/usr/sbin/cfgadm -v [-a] [-s listing_option] [-] [ap_id | ap_type...]

/usr/sbin/cfgadm -v -h [ap_id]...
```

Description The InfiniBand hardware specific library `/usr/lib/cfgadm/ib.so.1` provides the functionality for administering its fabric through the `cfgadm`(1M) utility. `cfgadm` operates on attachment points. See `cfgadm`(1M).

An InfiniBand (IB) device is enumerated by the IB nexus driver, `ib`(7D), based on the services from the IB Device Manager (IBDM).

The IB nexus driver creates and initializes five types of child device nodes:

- IB Port devices
- IB HCA service (HCA_SVC) devices
- IB Virtual Physical Point of Attachment (VPPA) devices
- I/O Controller (IOC)
- IB Pseudo devices

See `ib`(7D) for details on enumeration of IB Port, IB VPPA, and IB HCA_SVC devices. For additional information on IBDM, see `ibdm`(7D). See `ib`(4) for details on IB Pseudo devices.

For IB administration, two types of static attachment point are created for the fabric administration as seen by the given host. There is one static attachment point `ib` and all IB devices (either an IOC, Port, VPPA, HCA_SVC, or a Pseudo device) in the fabric are represented as dynamic attachment points based off of it. There is another static attachment point for each Host Channel Adapter (HCA) in the host based on its node Globally Unique Identifier (GUID) value.

Attachment points are named through ap_ids. There are two types of ap_ids: logical and physical. The physical ap_id is based on the physical path name. For the IB fabric it is `/devices/ib:fabric`. The logical ap_id is a shorter, and has a more user friendly name.

The static ap_id for the IB fabric is `ib`. The IB devices are dynamic attachment points and have no physical ap_id. The logical ap_id of an IOC contains its GUID, `ib::`*IOC-GUID*. An example of an IOC ap_id is `ib::80020123456789a`. The logical ap_id of a Pseudo device, see ib(4) for details, is of the format `ib::`*driver_name*`,`*unit-address*. An example of a pseudo ap_id would be `ib::sdp,0` where "`sdp`" is the driver name and "`0`" is its *unit-address* property. The logical ap_id of Port, VPPA and HCA_SVC device contains its Partition Key (*P_Key*), *Port GUID* / *Node GUID* and a communication service-name. The format of ap_id is as below:

Port device
`ib::`*PORT_GUID*`,0,`*service-name*

VPPA device
`ib::`*PORT_GUID*`,`*P_Key*`,`*service-name*

HCA_SVC device
`ib::`*HCA_GUID*`,0,`*servicename*

The Partition Key (*P_Key*) is `0` for Port and HCA_SVC devices. The *P_Key* helps determine the partition to which this port belongs for a VPPA device node. A port might have more than one *P_Key*. An example of a VPPA device logical ap_id point is `ib::80245678,ffff,ipib`. The *port-GUID* is `80245678`, the *P_Key* is `0xffff`, and the service name is `ipib`. The service-name information is obtained from the file `/kernel/drv/ib.conf` which contains service-name strings. The HCA's logical ap_id contains its node GUID value, hca:HCA-GUID. An example is `hca:21346543210a987`.

A listing of the IB attachment points includes information on all IB devices (IOC, VPPA, HCA_SVC, Pseudo, and Port devices seen by the IBDM and the IB nexus driver) in the fabric even if they are not seen by the host and configured for use.

The following shows a listing of five IB devices (two IOC, one VPPA, one Port, one HCA_SVC) and one HCA:

```
example# cfgadm -al
Ap_Id                     Type       Receptacle   Occupant      Condition
hca:21346543210a987       IB-HCA     connected    configured    ok
ib                        IB-FABRIC  connected    configured    ok
ib::80020123456789a       IB-IOC     connected    configured    ok
ib::802abc9876543         IB-IOC     connected    unconfigured  unknown
ib::80245678,ffff,ipib    IB-VPPA    connected    configured    ok
ib::12245678,0,nfs        IB-PORT    connected    configured    ok
ib::21346543,0,hnfs       IB-HCA_SVC connected    configured    ok
ib::sdp,0                 IB-PSEUDO  connected    configured    ok
```

The ap_id `ib::802abc9876543` shows an IOC device that is not yet configured by the host for use or had been previously offlined by an explicit

```
cfgadm -c unconfigure
```

operation. The distinction was made by the information displayed under the `Condition` column. The IB device with a zero P_Key and HCA GUID is a HCA_SVC device. Refer to `cfgadm(1M)` for more information regarding listing attachment points.

The receptacle state for attachment points have the following meanings:

connected
For an IOC/VPPA/Port/Pseudo/HCA_SVC device, `connected` implies that it has been seen by the host. The device might not have been configured for use by Solaris.

For a HCA attachment point, `connected` implies that it has been configured and is in use.

All IB ap_ids are always shown as connected.

The occupant state for attachment points have the following meanings:

configured
The IB device, and the HCA ap_id, are configured and usable by Solaris.

unconfigured
The IB device at the ap_id was explicitly offlined using `cfgadm -c unconfigure`, was not successfully configured. This could be because it wasn not successfully configuref for use with Solaris (no driver, or a device problem), or because it was never configured for use by the IB nexus driver.

The `unconfigured` operation is not supported for the HCA attachment point. The IB static apid, `ib`, is shown unconfigured if the system has no IB hardware.

The attachment point conditions are:

failed
Not used.

failing
Not used.

ok
Normal state. Ready for use.

unknown
This state is only valid for IB device that have been probed by IBDM but not yet configured for use by Solaris. It is also shown for devices that have been explicitly offlined by a `cfgadm -c unconfigure` operation. This condition does not apply to a HCA attachment point.

unusable
Not used.

Options The following options are supported:

`-c` *function*
The IB hardware specific library supports two generic commands (*function*s). These commands are not supported on the static attachment points (that is, the HCA ap_ids and the IB static `ib` ap_id).

The following generic commands are supported:

configure
Configure the IB device to be used by Solaris.

unconfigure
Unconfigure the IB device. If successful, `cfgadm` reports the `condition` of this `ap_id` as `unknown`.

`-f`
Not supported.

`-h` *ap_id*
Obtain IB specific help for an IB attachment point.

`-l`
List the state and condition of IB attachment points. The `-l` option works as described in `cfgadm(1M)`.

When paired with the `-a` option, displays the dynamic attachment points as well (IOC, VPPA, Port, Pseudo, and HCA_SVC devices).

When paired with `-v` option, displays verbose data about the ap_ids. For an IOC, the `Info` field in the

```
cfgadm -avl
```

output displays the following information: VendorID, IOCDeviceID, DeviceVersion, SubsystemVendorID, SubsystemID, Class, Subclass, Protocol, ProtocolVersion and IDString from the IOCControllerProfile. If the ID string isn't provided then nothing is displayed in its place. These fields are defined in the InfiniBand Specification Volume 1 (http://www.infinibandta.org).

For a VPPA, Port, or HCA_SVC device the `Info` field in the `cfgadm -lav` display shows the service name information to which this device is bound. If no such information exists, nothing is displayed.

For a Pseudo device `cfgadm -alv` displays the driver name and its `unit-address` information. For a HCA the verbose listing displays the VendorID, ProductID of the HCA, number of ports it has, and the PortGUID value of its ports. See `EXAMPLES`.

`-o` *hardware_option*
This option is not currently defined.

`-s` *listing_option*
Attachment points of class ib can be listed by using the select sub-option. Refer to the `cfgadm(1M)` man page for more information.

`-x` *hardware_function*
Perform a hardware specific function. Note that the *name* can not be more than 4 characters long.

The following hardware specific functions are supported:

`add_service -ocomm=[port|vppa|hca_svc],service=`*name*
This hardware specific function is supported on the static `IB` attachment point. It can be used to add a new service to `/kernel/drv/ib.conf` file and to update the `ib(7D)` driver.

You must use the `service=`*name* option to indicate the new service to be added. You must use the option `comm=[port|vppa|hca_svc]` option to add the `name` service to either `port-svc-list` or to the `hca-svc-list` in the `/kernel/drv/ib.conf` file. See `EXAMPLES`.

`delete_service -ocomm=[port|vppa|hca_svc],service=`*name*
This hardware specific function is supported on the static `IB` attachment point only. It can be used to delete an existing service from the `/kernel/drv/ib.conf` file and also from the ib(7D) driver's data base. You must use the `service=name` option to indicate which service to delete. You must use the `comm=[port|vppa|hca_svc]` option to delete this service from the `port-svc-list`, `vppa-svc-list`, or `vppa-svc-list` of the `/kernel/drv/ib.conf` file. See `EXAMPLES`.

`list_clients`
Supported on `HCA` attachment points. Displays all the kernel IB clients using this `HCA`. It also displays the respective ap_ids of these kernel IB clients and if they have opened an alternate `HCA` device. See `EXAMPLES`.

.

If a given kernel IB client does not have a valid ap_id then a `-` is displayed in that column.

`list_services`
This hardware specific function is supported on the static `IB` attachment point only. It lists all the Port and VPPA services as read from the `/kernel/drv/ib.conf` file. See `EXAMPLES`.

`unconfig_clients`
This hardware specific function is supported on the static HCA attachment point only. It can be used to unconfigure all IB kernel clients of this given HCA. Only IB kernel clients that do not have an alternate HCA are unconfigured. See `EXAMPLES`.

`update_ioc_config`
This hardware specific function is supported on static ib attachment point and the IOC attachment points. For the `ib` APID, this function updates properties of all the IOC device nodes. For the IOC APID, this function updates the properties of specified IOC device node. This command updates the `port-list`, `port-entries`, `service-id`, and `service-name` IOC node properties .

See ib(7D).

`update_pkey_tbls`
Supported on the static `ib` attachment point. Updates the `PKEY` information inside `IBTL`. `IBTL` re-reads the `P_Key` tables for all the ports on each `HCA` present on the host.

See ibtl(7D).

Examples

EXAMPLE 1 Listing the State and Condition of IB Devices

The following command lists the state and condition of IB devices on the system. It only shows the static attachment points.

EXAMPLE 1 Listing the State and Condition of IB Devices *(Continued)*

```
example# cfgadm
hca:21346543210a987        IB-HCA      connected    configured   ok
ib                         IB-FABRIC   connected    configured   ok
```

The `-a` option lists all attachment points. The following example uses the `-a` option and lists all attachment points:

```
example# cfgadm -a
hca:21346543210a987          IB-HCA      connected    configured   ok
ib                           IB-FABRIC   connected    configured   ok
ib::80020123456789a          IB-IOC      connected    unconfigured ok
ib::80245678,ffff,ipib       IB-VPPA     connected    configured   ok
ib::21346543,0,hnfs          IB-HCA_SVC  connected    configured   ok
ib::12245678,0,nfs           IB-PORT     connected    configured   ok
ib::sdp,0                    IB-PSEUDO   connected    configured   ok
```

EXAMPLE 2 Listing the Verbose Status of a IB VPPA Device

The following command lists the verbose status of a IB VPPA device:

```
example# cfgadm -alv ib::80245678,ffff,ipib
Ap_Id                  Receptacle Occupant   Condition Information
When         Type     Busy  Phys_Id
ib::80245678,ffff,ipib    connected    configured   ok        ipib
unavailable  IB-VPPA  n     /devices/ib:fabric::80245678,ffff,ipib
```

A verbose listing of an IOC shows additional information. The following command shows a verbose listing:

```
example# cfgadm -alv ib::80020123456789a
Ap_Id      Receptacle   Occupant     Condition Information
When       Type     Busy  Phys_Id
ib::80020123456789a     connected    configured   ok         VID: 0xeaea
DEVID: 0xeaea VER: 0x5 SUBSYS_VID: 0x0 SUBSYS_ID: 0x0 CLASS: 0xffff
SUBCLASS: 0xff PROTO: 0xff PROTOVER: 0x1 ID_STRING: Sample Host Adapter
unavailable IB-IOC   n     /devices/ib:fabric::80020123456789a
```

A verbose listing of a Pseudo device shows:

```
example# cfgadm -alv ib::sdp,0
Ap_Id                 Receptacle  Occupant   Condition Information
When       Type   Busy  Phys_Id
ib::sdp,0         connected   configured   ok       Driver = "sd
p" Unit-address = "0"
unavailable  IB-PSEUDO   n  /devices/ib:fabric::sdp,0
```

A verbose listing of a HCA shows:

EXAMPLE 2 Listing the Verbose Status of a IB VPPA Device *(Continued)*

```
example# cfgadm -alv hca:21346543210a987
Ap_Id               Receptacle   Occupant     Condition Information
When       Type    Busy  Phys_Id
hca:21346543210a987  connected    configured   ok         VID: 0x15b3,
PID: 0x5a44, #ports: 0x2, port1 GUID: 0x80245678, port2 GUID: 0x80245679
unavailable  IB-HCA     n  /devices/ib:21346543210a987
```

You can obtain more user-friendly output if you specify these following `cfgadm` class and field selection options: `-s "select=class(ib),cols=ap_id:info"`

The following command displays only IB ap_ids. The output only includes the `ap_id` and `Information` fields.

```
# cfgadm -al -s "cols=ap_id:info"  ib::80245678,ffff,ipib
Ap_Id                               Information
ib::80245678,ffff,ipib              ipib
```

EXAMPLE 3 Unconfiguring an Existing IB IOC

The following command unconfigures the IB IOC attached to `ib::80020123456789a`, then displays the status of the `ap_id`:

```
# cfgadm -c unconfigure ib::80020123456789a
Unconfigure the device: /devices/ib:fabric::80020123456789a
This operation will suspend activity on the IB device
Continue (yes/no)?
```

Enter: `y`

```
IB device unconfigured successfully.
# cfgadm -al ib::80020123456789a
Ap_Id                  Type      Receptacle  Occupant     Condition
ib::80020123456789     IB-IOC    connected   unconfigured unknown
#
```

The condition unknown implies that the device node doesn't exist anymore and this IB device's existence is known only to the IB Device Manager.

EXAMPLE 4 Configuring an IB IOC

The following series of commands configures an IB device attached to `ib::80020123456789a`:

```
# cfgadm -yc configure ib::80020123456789a
# cfgadm -al ib::80020123456789a
Ap_Id                  Type        Receptacle   Occupant     Condition
ib::80020123456789a    IB-IOC      connected    configured   ok
```

EXAMPLE 5 Listing All Kernel IB Clients of a HCA

The following command lists all kernel IB clients of an HCA attached to `hca:21346543210a987`:

```
# cfgadm -x list_clients hca:21346543210a987
Attachment Point       Clients                Alternate HCA
ib::80020123456789a    ioc1                   Yes
ib::80245678,ffff,ipib ipib                   No
ib::21346543,0,hnfs    hnfs                   No
-                      ibdm                   No
-                      ibmf                   No
```

EXAMPLE 6 Adding a Port Service

The following command adds a new Port service called `srp`:

```
 # cfgadm -o comm=port,service=srp -x add_service ib
```

EXAMPLE 7 Deleting a VPPA Service

The following command deletes the `ibd` VPPA service `ibd`:

```
# cfgadm -o comm=vppa,service=ipib -x delete_service ib
```

EXAMPLE 8 Listing Port, VPPA, HCA_SVC Services

The following command lists all Port, VPPA, and HCA_SVC services:

```
# cfgadm -x list_services ib
Port communication services:
         srp

VPPA communication services:
         ipib
         nfs

HCA_SVC communication services:
         hnfs
```

EXAMPLE 9 Reprobing IOC Devices

The following command reprobes all IOC device nodes.

```
# cfgadm -x update_ioc_config ib
This operation can update properties of IOC devices.
Continue (yes/no)?

Enter: y

#
```

EXAMPLE 10 Unconfiguring All Kernel Clients of a HCA

The following command unconfigures all kernel clients of a HCA

```
# cfgadm -x unconfig_clients hca:21346543
 This operation will unconfigure clients of this HCA.
 Continue (yes/no)?

 Enter: y
```

Files

`/usr/lib/cfgadm/ib.so.1`
Hardware-specific library for generic InfiniBand device administration

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/library

See Also

`cfgadm(1M)`, `config_admin(3CFGADM)`, `libcfgadm(3LIB)`, `ib(4)`, `attributes(5)`, `ib(7D)`, `ibdm(7D)`, `ibtl(7D)`

InfiniBand Specification Volume 1 (http://www.infinibandta.org)

Notes

Apart from the listing (`cfgadm -l` or `cfgadm -x list_clients`), only the superuser can execute any functions on an attachment point.

Name cfgadm_pci – PCI, CompactPCI, and PCI Express Hotplug hardware specific commands for cfgadm

Synopsis

```
/usr/sbin/cfgadm [-f] [-y | -n] [-v]
     [-o hardware_options] -c function ap_id [ap_id]

/usr/sbin/cfgadm [-f] [-y | -n] [-v]
     [-o hardware_options] -x hardware_function ap_id
     [ap_id]

/usr/sbin/cfgadm [-v] [-s listing_options]
     [-o hardware_options] [-l [ap_id | ap_type]]

/usr/sbin/cfgadm [-v] [-o harware_options] -t ap_id [ap_id]

/usr/sbin/cfgadm [-v] [-o hardware_function] -h
     [ap_id| ap_type]
```

Description The PCI hardware specific library, `/usr/lib/cfgadm/pci.so.1`, provides the support for hotplugging PCI and CompactPCI adapter cards into the respective hotpluggable slots in a system that is hotplug capable, through the `cfgadm` command (see `cfgadm`(1M)). This library does not include support for PCI Express Hotplug or Standard PCI Hotplug adapter cards, which are provided by a different library (see `cfgadm_shp`(1M)). Hotplug administrative models between PCI, CompactPCI remain the same except where noted in this document.

For PCI Hot Plug, each hotplug slot on a specific PCI bus is represented by an attachment point of that specific PCI bus.

An attachment point consist of two parts: a receptacle and an occupant. The `receptacle` under PCI Hot Plug is usually referred to as the physical hotpluggable slot; and the `occupant` is usually referred to as the PCI adapter card that plugs into the slot.

Attachment points are named through `ap_id`s. There are two types of `ap_id`s: logical and physical. The physical `ap_id` is based on the physical pathname, that is, `/devices/pci@1/hpc0_slot3`, whereas the logical `ap_id` is a shorter, and more user-friendly name. For PCI hotpluggable slots, the logical `ap_id` is usually the corresponding hotplug controller driver name plus the logical slot number, that is, `pci0:hpc0slot1`; PCI nexus driver, with hotplug controller driver named `hpc` and slot number `1`. The `ap_type` for PCI Hot Plug is `pci`.

Note that the `ap_type` is not the same as the information in the `Type` field.

See the *Oracle Solaris Administration: Common Tasks* for a detailed description of the hotplug procedure.

Options The following options are supported:

`-c` *function*
The following *function*s are supported for PCI hotpluggable slots:

configure
Configure the PCI device in the slot to be used by Solaris.

connect
Connect the slot to PCI bus.

disconnect
Disconnect the slot from the PCI bus.

insert
Not supported.

remove
Not supported.

unconfigure
Logically remove the PCI device's resources from the system.

`-f`
Not supported.

`-h` *ap_id* | *ap_type*
Print out PCI Hot Plug-specific help message.

`-l` *list*
List the values of PCI Hot Plug slots.

`-o` *hardware_options*
No hardware specific options are currently defined.

`-s` *listing_options*
Same as the generic `cfgadm(1M)`.

`-t` *ap_id*
This command is only supported on platforms which support testing capability on the slot.

`-v`
Execute in verbose mode.

When the `-v` option is used with the `-l` option, the `cfgadm` command outputs information about the attachment point. For PCI Hotplug attachment points located in a PCI PCI Express hierarchy, see `cfgadm_shp(1M)` for details. For PCI Hot Plug attachment points not located in a PCI Express hiearchy, the `Information` field will be the slot's system label, if any. This string will be obtained from the `slot-name` property of the slot's bus node. The information in the `Type` field is printed with or without the `v` option. The occupant `Type` field will describe the contents of the slot. There are 2 possible values:

unknown
The slot is empty. If a card is in the slot, the card is not configured or there is no driver for the device on the card.

subclass/*board*
The card in the slot is either a single-function or multi-function device.

subclass is a string representing the subclass code of the device, for example, SCSI, `ethernet`, `pci-isa`, and so forth. If the card is a multi-functional device, `MULT` will get printed instead.

board is a string representing the board type of the device. For example, hp is the string used for a PCI Hot Plug adapter, hs is used for a Hot Swap Board, nhs for a Non—Hot Swap cPCI Board, bhs for a Basic Hot Swap cPCI Board, and fhs for a Full Hot Swap cPCI Board.

Most PCI cards with more than one device are not multi-function devices, but are implemented as a PCI bridge with arbitrary devices behind them. In those cases, the subclass displayed is that of the PCI bridge. Most commonly, the bridges are `pci-pci,` a generic PCI to PCI bridge or `stpci`, a semi-transparent PCI bridge.

`-x` *hardware_function*
Perform hardware specific function. These hardware specific functions should not normally change the state of a receptacle or occupant.

The following *hardware_functions* are supported:

enable_slot | disable_slot
Change the state of the slot and preserve the state of slot across reboot. Preservation of state across reboot is only supported on select platforms.

`enable_slot` enables the addition of hardware to this slot for hotplugging and at boot time.

`disable_slot` disables the addition of hardware to this slot for hotplugging and at boot time. When a slot is disabled its condition is shown as unusable.

enable_autoconfig | disable_autoconfig
Change the ability to autoconfigure the occupant of the slot. Only platforms that support auto configuration support this feature.

`enable_autoconfig` enables the ability to autoconfigure the slot.

`diable_autoconfig` disables the ability to autoconfigure the slot.

Autoconfiguration is done through the attention button on the PCI Express platforms and through the injector/ejector latch on the CompactPCI platforms. When autoconfiguration is disabled, the attention button or latch mechanism cannot be used to configure the occupant of the slot.

led=[*led_sub_arg*],mode=[*mode_sub_arg*]
Without sub-arguments, print a list of the current LED settings. With sub-arguments, set the mode of a specific LED for a slot.

Specify *led_sub_arg* as `fault`, `power`, `attn`, or `active`.

Specify *mode_sub_arg* as `on`, `off` or `blink`.

Changing the state of the LED does not change the state of the receptacle or occupant. Normally, the LEDs are controlled by the hotplug controller, no user intervention is necessary. Use this command for testing purposes.

Caution: Changing the state of the LED can misrepresent the state of occupant or receptacle.

The following command prints the values of LEDs:

```
example#  cfgadm -x led pci0:hpc0_slot1
Ap_Id             Led
pci0:hpc0_slot1   power=on,fault=off,active=off,attn=off
```

The following command turns on the Fault LED:

```
example# cfgadm -x led=fault,mode=on pci0:hpc0_slot1
```

The following command turns off the Power LED:

```
example# cfgadm -x led=power,mode=off pci0:hpc0_slot0
```

The following command sets the *active* LED to blink to indicate the location of the slot:

```
example# cfgadm -x led=active,mode=on pci0:hpc0_slot3
```

Examples

EXAMPLE 1 Printing out the Value of Each Slot

The following command prints out the values of each slot:

```
example# cfgadm -l
Ap_Id            Type         Receptacle   Occupant       Condition
c0               scsi-bus     connected    configured     unknown
c1               scsi-bus     connected    unconfigured   unknown
c2               scsi-bus     connected    unconfigured   unknown
cpci_slot1       stpci/fhs    connected    configured     ok
cpci_slot2       unknown      empty        unconfigured   unknown
cpci_slot4       stpci/fhs    connected    configured     ok
cpci_slot5       stpci/fhs    connected    configured     ok
```

EXAMPLE 2 Replacing a Card

The following command lists all DR-capable attachment points:

```
example# cfgadm


Type             Receptacle   Occupant     Condition
c0               scsi-bus     connected    configured     unknown
c1               scsi-bus     connected    unconfigured   unknown
```

EXAMPLE 2 Replacing a Card *(Continued)*

```
c2              scsi-bus     connected    unconfigured   unknown
cpci_slot1      stpci/fhs    connected    configured     ok
cpci_slot2      unknown      empty        unconfigured   unknown
cpci_slot4      stpci/fhs    connected    configured     ok
cpci_slot5      stpci/fhs    connected    configured     ok
```

The following command unconfigures and electrically disconnects the card:

```
example# cfgadm -c disconnect cpci_slot4
```

The change can be verified by entering the following command:

```
example# cfgadm cpci_slot4


Ap_Id                  Type         Receptacle   Occupant     Condition
cpci_slot4             unknown      disconnected unconfigured unknown
```

Now the card can be swapped. The following command electrically connects and configures the card:

```
example# cfgadm -c configure cpci_slot4
```

The change can be verifed by entering the following command:

```
example# cfgadm cpci_slot4


Ap_Id                  Type         Receptacle   Occupant     Condition
cpci_slot4             stpcipci/fhs connected    configured   ok
```

Files `/usr/lib/cfgadm/pci.so.1`
Hardware specific library for PCI hotplugging.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/library

See Also `cfgadm(1M)`, `cfgadm_shp(1M)`, `config_admin(3CFGADM)`, `libcfgadm(3LIB)`, `attributes(5)`

Oracle Solaris Administration: Common Tasks

Name cfgadm_sata – SATA hardware-specific commands for cfgadm

Synopsis

```
/usr/sbin/cfgadm [-f] [-y | -n] [-v] [-o hardware_options]
     -c function ap_id...

/usr/sbin/cfgadm [-f] [-y | -n] [-v] [-o hardware_options]
     -x hardware_function ap_id...

/usr/sbin/cfgadm [-v] [-a] [-s listing_options]
     [-o hardware_options] [-l [ap_id | ap_type]...]

/usr/sbin/cfgadm [-v] [-o harware_options] -t ap_id...

/usr/sbin/cfgadm [-v] [-o hardware_options] -h [ap_id]...
```

Description The SATA hardware specific library, `/usr/lib/cfgadm/sata.so.1`, provides the functionality for SATA hot plugging through the `cfgadm` command. `cfgadm` operates on attachment points, which are locations in the system where hardware resources can be dynamically reconfigured. See `cfgadm(1M)` for information regarding attachment points.

Each SATA controller's and port multiplier's device port is represented by an attachment point in the device tree. SATA devices, connected and configured in the system are shown as the attachment point name extension. The terms “attachment point” and “SATA port” are used interchangeably in the following description.

Attachment points are named through `ap_id`s. All the SATA attachment points `ap_id` consist of a string in the following form:

```
sataX/P[.M][::dsk/cXtYd0]
```

where

X	is the SATA controller number
P	is the SATA controller's device port number (0 to 31)
M	is the port multiplier's device port number (0 to 14) the port multiplier host port number (15). It is used only when the port multiplier is attached to the SATA controller's device port.
dev/cXtYd0	identifies the attached SATA device
Y	is a target number

In general, the device identifier is derived from the corresponding logical link for the device in */dev*. Because only one LUN (LUN 0) is supported by the SATA device, the “d” component of the device string will always have number 0 (zero).

For example, the logical `ap_id` of the device port 4 of the port multiplier connected to the device port 5 of the SATA controller 2 would be:

```
sata2/5.4
```

If the SATA disk or CD/DVD device is connected to this attachment point, and the device is configured, the *ap_id* would be:

```
sata2/5.4::dsk/c2t645d0
```

The *cXtYd0* string identifying a device has one-to-one correspondence to the device attachment point.

A simple listing of attachment points in the system will include all SATA device ports and attached devices. For example:

```
#cfgadm -l
Ap_Id                      Type        Receptacle   Occupant     Condition
  sata0/0::dev/c0t0d0      disk        connected    configured   ok
  sata0/1::dev/c0t1d0      disk        connected    configured   ok
  sata0/2::dev/c0t2d0      cd-dvd      connected    configured   ok
  sata0/3                  sata-port   empty        unconfigured ok
  sata1/0                  sata-port   disconnected unconfigured unknown
  sata1/1                  sata port   disconnected unconfigured unknown
  sata1/2                  sata port   empty        unconfigured ok
  sata1/3.15               sata-pmult  connected    configured   ok
  sata1/3.0::dev/c0t512d0 disk         connected    configured   ok
  sata1/3.1                sata-port   empty        unconfigured ok
  sata1/3.2                sata-port   empty        unconfigured ok
  sata1/3.3                sata-port   empty        unconfigured ok
  usb0/1                   unknown     empty        unconfigured ok
  usb0/2                   unknown     empty        unconfigured ok
```

See cfgadm(1M)for more information regarding listing of attachment points.

The receptacle state for attachment point at the SATA port have the following meanings:

empty	The SATA port is powered-on and enabled. No device presence was detected on this port.
disconnected	The SATA port is not enabled or the SATA device presence was detected but no communication with the device was established, or the port has failed.
connected	The SATA device is detected on the port the communication with the device is established.

The occupant (device attached to the SATA port) state have the following meanings:

configured	The attached SATA device is configured and ready to use by the operating system.
unconfigured	No device is attached, or the SATA device attached to the SATA port was not yet configured. To configure it, run the command "`cfgadm -c configure ap_id`".

The attachment point (SATA port) condition have the following meanings:

ok	The SATA port is powered-on and enabled, and is ready for use.
failed	The SATA port failed. It may be disabled and/or powered-off by the system. It is unusable and its condition is unknown. It may be due to the device plugged-in.
unknown	The SATA port is disabled and its condition is unknown.

A "state table" is the combination of an attachment point receptacle state, an occupant state, and an attachment point (SATA port) condition. The valid states are:

empty/unconfigured/ok	The SATA port is enabled and active. No device presence was detected.
disconnected/unconfigured/ok	The SATA port is enabled and a device presence was detected but no communications with the device was established.
disconnected/unconfigured/unknown	The SATA Port is disabled and its condition is unknown.
disconnected/unconfigured/failed	The SATA Port is disabled and unusable. The port was disabled by the system due to a system-detected failure.
connected/unconfigured/ok	The SATA Port is enabled and active. A device presence was detected and the communication with a device was established. The device is not configured to be used by the OS.
connected/configured/ok	The device is present and configured, and is ready to use by the OS.

Options `cfgadm` defines several types of operations besides listing (`-l`). These operations include testing, (`-t`), invoking configuration state changes, (`-c`), invoking hardware specific functions (`-x`), and obtaining configuration administration help messages (`-h`).

`-c` *function*

The following generic *function*s are defined for the SATA hardware specific library. For SATA port attachment point, the following configuration state change operations are supported:

connect

Enable (activate) the SATA port and establish the communication with an attached device. This operation implies powering-on the port if necessary.

disconnect

Unconfigure the attached device, if it is not already unconfigured, and disable (deactivate) the SATA port. A subsequent "`connect`" command enables SATA port

operation but does not bring a device to the "configured" state.

For a SATA device attached to the SATA port following state change operations are supported:

configure — Configure new device for use by the operating system if it is not already configured. This command also implies connect operation, if necessary.

unconfigure — Unconfigure the device connected to the SATA port if it is not already unconfigured.

The `configure` and `unconfigure` operations cannot be used for an attachment point where the port multiplier is connected. Port multipliers are configured and unconfigured automatically by the system. However, configure and unconfigure operations apply to all SATA devices connected to the port multiplier's device ports.

`-f`
Not supported.

`-h` *ap_id*
SATA specific help can be obtained by using the help option with any SATA attachment point.

`-l` [`-v`]
The `-l` option works as described in `cfgadm(1M)`. When paired with the `-v` option, the "Information" field contains the following SATA-specific information:

- Mfg: manufacturer string
- Product: product string
- No: product Serial Number

`-o` *hardware_options*
No hardware specific options are currently defined.

`-s` *listing_options*
Attachment points of class SATA can be listed by using the select suboption. See `cfgadm(1M)`.

`-t` *ap_id*
Perform self-test of the SATA port, if supported by the SATA controller. If a port self-test operation is not supported by the SATA controller, an error message is issued.

`-x` *hardware_function*
Perform hardware specific function.

Some of the following commands used on the SATA ports or the SATA controller may affect any SATA devices that have been attached, as noted. `ap_id` refers to SATA port or the entire SATA controller, as noted. If the operation implies unconfiguring a device, but it cannot be unconfigured (that is, the device contains a mounted filesystem), an error

message is issued and the operation is not performed. An error message will be also issued if the SATA controller does not support specified operation.

sata_reset_device ap_id
: Reset the SATA device attached to `ap_id` SATA port. The SATA port state does not change.

sata_reset_port ap_id
: Reset the SATA port specified by `ap_id`. If a SATA device is attached to the port, it is also reset. This operation may be also performed on the port to which a port multiplier is connected. If a port multiplier is connected to the SATA controller port, the SATA devices attached to the port multiplier may not be reset

sata_reset_all ap_id
: Reset SATA controller specified by the controller number part in `ap_id` and all attached devices and re-enumerate all connected devices, including port multipliers and devices connected to port multipliers' device ports.

  This operations implies unconfiguring all attached devices prior to the operation. Any newly enumerated devices will be left unconfigured.

sata_port_deactivate ap_id
: Force the deactivation of the port when all else fails. This is meant as an emergency step; use with caution.

sata_port_activate ap_id
: Force the activation of a port. This is meant for emergency situations on a port which was deactivated to recover from errors.

sata_port_self_test ap_id
: Perform self-test operation on the SATA controller. This operation implies unconfiguring all devices and resetting the SATA controller.

`-v`
: Execute in verbose mode.

  The following Transitions table reports the state transitions resulting from the `-c` operations and hotplugging actions:

```
current state     operation       possible new state
-------------     ---------       ------------------
empty/
unconfigured/ok   device plug-in  connected/unconfigured/ok, or
                                  disconnected/unconfigured/ok, or
                                  disconnected/unconfigured/failed

empty/
unconfigured/ok   -c unconfigure  error message, no state change

empty/
```

```
unconfigured/ok   -c configure    error message, no state change

empty/
unconfigured/ok   -c connect      error message, no state change

empty/
unconfigured/ok   -c disconnect   disconnected/unconfigured/unknown, or
                                  disconnected/unconfigured/failed

disconnected/
unconfigured/ok   device unplug   no state change

disconnected/
unconfigured/ok   -c unconfigure  error message, no state change

disconnected/
unconfigured/ok   -c configure    error message, no state change

disconnected/
unconfigured/ok   -c connect      error message, no state change

disconnected/
unconfigured/ok   -c disconnect   error message, no state change

disconnected/
unconfigured/
unknown
(no disk plugged) -c configure    error message, state change to
                                  empty/unconfigured/ok, or
                                  disconnected/unconfigured/failed

disconnected/
unconfigured/
unknown           -c configure    state change to
(disk plugged)                    connected/configured/ok or,
                                  connected/unconfigured/ok, or
                                  disconnected/unconfigured/failed and
                                  possible error message

disconnected/
unconfigured/
unknown           -c connect      empty/unconfigured/ok, or
                                  connected/unconfigured/ok, or
                                  disconnected/unconfigured/ok, or
                                  disconnected/unconfigured/unknown, or
                                  disconnected/unconfigured/failed

disconnected/
```

unconfigured/ unknown	-c disconnect	error message, no state change
disconnected/ unconfigured/ failed	any command other than -x commands	error message, no state change
connected/ unconfigured/ok	disk unplug	error message and state: empty/unconfigured/ok, or disconnected/unconfigured/failed
connected/ unconfigured/ok	-c configure	connected/unconfigured/ok, or connected/configured/ok, or disconnected/unconfigured/ok, or disconnected/unconfigured/failed
connected/ unconfigured/ok	-c unconfigure	error message, no state change
connected/ unconfigured/ok	-c connect	error message, no state change
connected/ unconfigured/ok	-c disconnect	disconnected/unconfigured/unknown, or disconnected/unconfigured/failed
connected/ configured/ok	disk unplug	error message and state: empty/unconfigured/ok, or disconnected/unconfigured/failed
connected/ configured/ok	-c configure	error message, no state change
connected/ configured/ok	-c unconfigure	error message, if device cannot be unconfigured, no state change, or connected/unconfigured/ok, or disconnected/unconfigured/ok, or disconnected/unconfigured/failed
connected/ configured/ok	-c connect	error message, no state change

```
connected/
configured/ok     -c disconnect   error message, if device cannot be
                                  unconfigured, no state change, or
                                  disconnected/unconfigured/unknown, or
                                  disconnected/unconfigured/failed
```

Examples

EXAMPLE 1 Configuring a Disk

The following command configures a disk attached to SATA controller 0, port 0:

```
example# cfgadm -c configure sata0/0
```

This command should be issued only when there is a device connected to the SATA port.

EXAMPLE 2 Unconfiguring a Disk

The following command unconfigures a disk attached to SATA controller 0, port 3:

```
example# cfgadm -c unconfigure sata0/3
```

The device identifying string is shown when the attachment point receptacle state is "connected" and occupant state is "configured".

EXAMPLE 3 Encountering a Mounted File System While Unconfiguring a Disk

The following command illustrates encountering a mounted file system while unconfiguring a disk:

```
example# cfgadm -c unconfigure sata1/5
```

The system responds with the following:

```
cfgadm: Component system is busy, try again: failed to offline:
/devices/pci@0,0/pci8086,244e@1e/pci1095,3124@1/sd@5,0
     Resource              Information
------------------  -------------------------
/dev/dsk/c1t5d0s0   mounted filesystem "/mnt"
```

Files

/usr/lib/cfgadm/sata.so.1 Hardware specific library for generic SATA hot plugging.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/library

See Also

`cfgadm(1M)`, `config_admin(3CFGADM)`, `libcfgadm(3LIB)`, `attributes(5)`

Notes

The emergency "sata_port_deactivate" operation is not supported on ports with attached disks containing critical partitions such as root (/), /usr, swap, or /var. The deactivate operation should not be attempted on such ports. Incorrect usage can result in a system hang and require a reboot.

Hotplugging operations are not supported by all SATA controllers.

If SATA connectors are the hot-pluggable type and the SATA controller supports hotplugging, a SATA device can be hotplugged at any time. The system detects the event and establishes the communication with the device. The device has to be configured by the explicit "`cfgadm -c configure ap_id`" command.

If the SATA connectors are the hot-pluggable type and the SATA controller supports hotplugging, unplugging a device without unconfiguring it may result in system hang or data loss. If a device is unconfigured but receptacle state is not in a disconnected state, unplugging a device from the SATA port will result in error message.

Warnings The connectors on some SATA devices do not conform to SATA hotplug specifications. Performing hotplug operations on such devices can cause damage to the SATA controller and/or the SATA device.

Name cfgadm_sbd – `cfgadm` commands for system board administration

Synopsis

```
cfgadm -l [-a] [-o parsable] ap_id...

cfgadm -c function [-f] [-y | -n]
     [-o unassign | nopoweroff] [-v] ap_id...

cfgadm -t [-v] ap_id...

cfgadm -x [-f] [-v] function ap_id...
```

Description The `cfgadm_sbd` plugin provides dynamic reconfiguration functionality for connecting, configuring, unconfiguring, and disconnecting class `sbd` system boards. It also enables you to connect or disconnect a system board from a running system without having to reboot the system.

The `cfgadm` command resides in `/usr/sbin`. See `cfgadm(1M)`. The `cfgadm_sbd` plugin resides `/usr/platform/sun4u/lib/cfgadm`.

Each board slot appears as a single attachment point in the device tree. Each component appears as a dynamic attachment point. You can view the type, state, and condition of each component, and the states and condition of each board slot by using the `-a` option.

The `cfgadm` options perform differently depending on the platform. Additionally, the form of the attachment points is different depending on the platform. See the `Platform Notes` section for more information.

Component Conditions The following are the names and descriptions of the component conditions:

failed
: The component failed testing.

ok
: The component is operational.

unknown
: The component has not been tested.

Component States The following is the name and description of the receptacle state for components:

connected
: The component is connected to the board slot.

The following are the names and descriptions of the occupant states for components:

configured
: The component is available for use by the Solaris operating environment.

unconfigured
: The component is not available for use by the Solaris operating environment.

Board Conditions

The following are the names and descriptions of the board conditions.

failed
The board failed testing.

ok
The board is operational.

unknown
The board has not been tested.

unusable
The board slot is unusable.

Board States

Inserting a board changes the receptacle state from empty to disconnected. Removing a board changes the receptacle state from disconnected to empty.

Caution: Removing a board that is in the connected state or that is powered on and in the disconnected state crashes the operating system and can result in permanent damage to the system.

The following are the names and descriptions of the receptacle states for boards:

connected
The board is powered on and connected to the system bus. You can view the components on a board only after it is in the connected state.

disconnected
The board is disconnected from the system bus. A board can be in the disconnected state without being powered off. However, a board must be powered off and in the disconnected state before you remove it from the slot.

empty
A board is not present.

The occupant state of a disconnected board is always unconfigured. The following table contains the names and descriptions of the occupant states for boards:

configured
At least one component on the board is configured.

unconfigured
All of the components on the board are unconfigured.

Dynamic System Domains

Platforms based on dynamic system domains (DSDs, referred to as domains in this document) divide the slots in the chassis into electrically isolated hardware partitions (that is, DSDs). Platforms that are not based on DSDs assign all slots to the system permanently.

A slot can be empty or populated, and it can be assigned or available to any number of domains. The number of slots available to a given domain is controlled by an available

component list (ACL) that is maintained on the system controller. The ACL is not the access control list provided by the Solaris operating environment.

A slot is visible to a domain only if the slot is in the domain's ACL and if it is not assigned to another domain. An unassigned slot is visible to all domains that have the slot in their ACL. After a slot has been assigned to a domain, the slot is no longer visible to any other domain.

A slot that is visible to a domain, but not assigned, must first be assigned to the domain before any other state changing commands are applied. The assign can be done explicitly using `-x assign` or implicitly as part of a connect. A slot must be unassigned from a domain before it can be used by another domain. The unassign is always explicit, either directly using `-x unassign` or as an option to disconnect using `-o unassign`.

State Change Functions

Functions that change the state of a board slot or a component on the board can be issued concurrently against any attachment point. Only one state changing operation is permitted at a given time. A `Y` in the Busy field in the state changing information indicates an operation is in progress.

The following list contains the functions that change the state:

- configure
- unconfigure
- connect
- disconnect

Availability Change Functions

Commands that change the availability of a board can be issued concurrently against any attachment point. Only one availability change operation is permitted at a given time. These functions also change the information string in the `cfgadm -l` output. A `Y` in the Busy field indicates that an operation is in progress.

The following list contains the functions that change the availability:

- `assign`
- `unassign`

Condition Change Functions

Functions that change the condition of a board slot or a component on the board can be issued concurrently against any attachment point. Only one condition change operation is permitted at a given time. These functions also change the information string in the `cfgadm -l` output. A `Y` in the Busy field indicates an operation is in progress.

The following list contains the functions that change the condition:

- `poweron`
- `poweroff`
- `test`

Unconfigure Process

This section contains a description of the unconfigure process, and illustrates the states of source and target boards at different stages during the process of moving permanent memory.

In the following code examples, the permanent memory on board 0 must be moved to another board in the domain. Thus, board 0 is the source, and board 1 is the target.

A status change operation cannot be initiated on a board while it is marked as busy. For brevity, the CPU information has been removed from the code examples.

The process is started with the following command:

```
# cfgadm -c unconfigure -y SB0::memory &
```

First, the memory on board 1 in the same address range as the permanent memory on board 0 must be deleted. During this phase, the source board, the target board, and the memory attachment points are marked as busy. You can display the status with the following command:

```
# cfgadm -a -s cols=ap_id:type:r_state:o_state:busy SB0 SB1

Ap_Id          Type      Receptacle     Occupant       Busy
SB0            CPU       connected      configured     y
SB0::memory    memory    connected      configured     y
SB1            CPU       connected      configured     y
SB1::memory    memory    connected      configured     y
```

After the memory has been deleted on board 1, it is marked as unconfigured. The memory on board 0 remains configured, but it is still marked as busy, as in the following example.

```
Ap_Id          Type      Receptacle     Occupant       Busy
SB0            CPU       connected      configured     y
SB0::memory    memory    connected      configured     y
SB1            CPU       connected      configured     y
SB1::memory    memory    connected      unconfigured   n
```

The memory from board 0 is then copied to board 1. After it has been copied, the occupant state for the memory is switched. The memory on board 0 becomes unconfigured, and the memory on board 1 becomes configured. At this point in the process, only board 0 remains busy, as in the following example.

```
Ap_Id          Type      Receptacle     Occupant       Busy
SB0            CPU       connected      configured     y
SB0::memory    memory    connected      unconfigured   n
SB1            CPU       connected      configured     n
SB1::memory    memory    connected      configured     n
```

After the entire process has been completed, the memory on board 0 remains unconfigured, and the attachment points are not busy, as in the following example.

```
Ap_Id          Type      Receptacle     Occupant       Busy
SB0            CPU       connected      configured     n
SB0::memory    memory    connected      unconfigured   n
SB1            CPU       connected      configured     n
SB1::memory    memory    connected      configured     n
```

The permanent memory has been moved, and the memory on board 0 has been unconfigured. At this point, you can initiate a new state changing operation on either board.

Platform-Specific Options

You can specify platform-specific options that follow the options interpreted by the system board plugin. All platform-specific options must be preceded by the `platform` keyword. The following example contains the general format of a command with platform-specific options:

command `-o` *sbd_options*`,platform=`*platform_options*

Options

This man page does not include the `-v`, `-a`, `-s`, or `-h` options for the `cfgadm` command. See `cfgadm`(1M) for descriptions of those options. The following options are supported by the `cfgadm_sbd` plugin:

`-c` *function*

Performs a state change function. You can use the following functions:

unconfigure

Changes the occupant state to unconfigured. This function applies to system board slots and to all of the components on the system board.

The `unconfigure` function removes the CPUs from the CPU list and deletes the physical memory from the system memory pool. If any device is still in use, the `cfgadm` command fails and reports the failure to the user. You can retry the command as soon as the device is no longer busy. If a CPU is in use, you must ensure that it is off line before you proceed. See `pbind`(1M), `psradm`(1M) and `psrinfo`(1M).

The `unconfigure` function moves the physical memory to another system board before it deletes the memory from the board you want to unconfigure. Depending of the type of memory being moved, the command fails if it cannot find enough memory on another board or if it cannot find an appropriate physical memory range.

For permanent memory, the operating system must be suspended (that is, quiesced) while the memory is moved and the memory controllers are reprogrammed. If the operating system must be suspended, you will be prompted to proceed with the operation. You can use the `-y` or `-n` options to always answer yes or no respectively.

Moving memory can take several minutes to complete, depending on the amount of memory and the system load. You can monitor the progress of the operation by issuing a status command against the memory attachment point. You can also interrupt the memory operation by stopping the `cfgadm` command. The deleted memory is returned to the system memory pool.

disconnect
Changes the receptacle state to disconnected. This function applies only to system board slots.

If the occupant state is configured, the `disconnect` function attempts to unconfigure the occupant. It then powers off the system board. At this point, the board can be removed from the slot.

This function leaves the board in the assigned state on platforms that support dynamic system domains.

If you specify `-o nopoweroff`, the `disconnect` function leaves the board powered on. If you specify `-o unassign`, the `disconnect` function unassigns the board from the domain.

If you unassign a board from a domain, you can assign it to another domain. However, if it is assigned to another domain, it is not available to the domain from which is was unassigned.

configure
Changes the occupant state to configured. This function applies to system board slots and to any components on the system board.

If the receptacle state is disconnected, the `configure` function attempts to connect the receptacle. It then walks the tree of devices that is created by the `connect` function, and attaches the devices if necessary. Running this function configures all of the components on the board, except those that have already been configured.

For CPUs, the `configure` function adds the CPUs to the CPU list. For memory, the `configure` function ensures that the memory is initialized then adds the memory to the system memory pool. The CPUs and the memory are ready for use after the `configure` function has been completed successfully.

For I/O devices, you must use the `mount` and the `ifconfig` commands before the devices can be used. See `ifconfig(1M)` and `mount(1M)`.

connect
Changes the receptacle state to connected. This function applies only to system board slots.

If the board slot is not assigned to the domain, the `connect` function attempts to assign the slot to the domain. Next, it powers on and tests the board, then it connects the board electronically to the system bus and probes the components.

After the `connect` function is completed successfully, you can use the `-a` option to view the status of the components on the board. The `connect` function leaves all of the components in the unconfigured state.

The assignment step applies only to platforms that support dynamic system domains.

`-f`
Overrides software state changing constraints.

The `-f` option never overrides fundamental safety and availability constraints of the hardware and operating system.

`-l`
Lists the state and condition of attachment points specified in the format controlled by the `-s`, `-v`, and `-a` options as specified in `cfgadm(1M)`. The `cfgadm_sbd` plugin provides specific information in the info field as described below. The format of this information might be altered by the `-o parsable` option.

The parsable `info` field is composed of the following:

cpu
The `cpu` type displays the following information:

`cpuid=#[,#…]`
Where # is a number, and represents the ID of the CPU. If more than one # is present, this CPU has multiple active virtual processors.

`speed=#`
Where # is a number and represents the speed of the CPU in MHz.

`ecache=#`
Where # is a number and represents the size of the ecache in MBytes. If the CPU has multiple active virtual processors, the ecache could either be shared among the virtual processors, or divided between them.

memory
The `memory` type displays the following information, as appropriate:

address=#
Where # is a number, representing the base physical address.

size=#
Where # is a number, representing the size of the memory in KBytes.

permanent=#
Where # is a number, representing the size of permanent memory in KBytes.

unconfigurable
An operating system setting that prevents the memory from being unconfigured.

inter-board-interleave
The board is participating in interleaving with other boards.

source=*ap_id*
Represents the source attachment point.

target=*ap_id*
Represents the target attachment point.

deleted=#
Where # is a number, representing the amount of memory that has already been deleted in KBytes.

remaining=#
Where # is a number, representing the amount of memory to be deleted in KBytes.

io
The `io` type displays the following information:

device=*path*
Represents the physical path to the I/O component.

referenced
The I/O component is referenced.

board
The `board` type displays the following boolean names. If they are not present, then the opposite applies.

assigned
The board is assigned to the domain.

powered-on
The board is powered on.

The same items appear in the `info` field in a more readable format if the `-o parsable` option is not specified.

`-o` parsable
Returns the information in the `info` field as a boolean *name* or a set of `name=value` pairs, separated by a space character.

The `-o parsable` option can be used in conjunction with the `-s` option. See the `cfgadm`(1M) man page for more information about the `-s` option.

`-t`
Tests the board.

Before a board can be connected, it must pass the appropriate level of testing.

Use of this option always attempts to test the board, even if it has already passed the appropriate level of testing. Testing is also performed when a `-c connect` state change function is issued, in which case the test step can be skipped if the board already shows an appropriate level of testing. Thus the `-t` option can be used to explicitly request that the board be tested.

`-x` *function*
Performs an sbd-class function. You can use the following functions:

assign
Assigns a board to a domain.

The receptacle state must be disconnected or empty. The board must also be listed in the domain available component list. See Dynamic System Domains.

unassign
Unassigns a board from a domain.

The receptacle state must be disconnected or empty. The board must also be listed in the domain available component list. See Dynamic System Domains.

poweron
Powers the system board on.

The receptacle state must be disconnected.

poweroff
Powers the system board off.

The receptacle state must be disconnected.

Operands The following operands are supported:

Receptacle *ap_id*
For the Sun Fire high-end systems such as the Sun Fire 15K , the receptacle attachment point ID takes the form `SB`*X* or `IO`*X*, where *X* equals the slot number.

The exact format depends on the platform and typically corresponds to the physical labelling on the machine. See the platform specific information in the `NOTES` section.

Component *ap_id*
The component attachment point ID takes the form *component_typeX*, where *component_type* equals one of the component types described in "Component Types" and *X* equals the component number. The component number is a board-relative unit number.

The above convention does not apply to memory compontents. Any DR action on a memory attachment point affects all of the memory on the system board.

Examples The following examples show user input and system output on a Sun Fire 15K system. User input, specifically references to attachment points and system output might differ on other Sun Fire systems, such as the Sun Fire midrange systems such as the 6800. Refer to the Platform Notes for specific information about using the `cfgadm_sbd` plugin on non-Sun Fire high-end models.

EXAMPLE 1 Listing All of the System Board

```
# cfgadm -a -s "select=class(sbd)"

Ap_Id         Type      Receptacle     Occupant       Condition
```

EXAMPLE 1 Listing All of the System Board *(Continued)*

```
SB0           CPU       connected      configured     ok
SB0::cpu0     cpu       connected      configured     ok
SB0::memory   memory    connected      configured     ok
IO1           HPCI      connected      configured     ok
IO1::pci0     io        connected      configured     ok
IO1::pci1     io        connected      configured     ok
SB2           CPU       disconnected   unconfigured   failed
SB3           CPU       disconnected   unconfigured   unusable
SB4           unknown   empty          unconfigured   unknown
```

This example demonstrates the mapping of the following conditions:

- The board in Slot 2 failed testing.
- Slot 3 is unusable; thus, you cannot hot plug a board into that slot.

EXAMPLE 2 Listing All of the CPUs on the System Board

```
# cfgadm -a -s "select=class(sbd):type(cpu)"

Ap_Id         Type      Receptacle     Occupant       Condition
SB0::cpu0     cpu       connected      configured     ok
SB0::cpu1     cpu       connected      configured     ok
SB0::cpu2     cpu       connected      configured     ok
SB0::cpu3     cpu       connected      configured     ok
```

EXAMPLE 3 Displaying the CPU Information Field

```
# cfgadm -l -s noheadings,cols=info SB0::cpu0

cpuid 16, speed 400 MHz, ecache 8 Mbytes
```

EXAMPLE 4 Displaying the CPU Information Field in Parsable Format

```
# cfgadm -l -s noheadings,cols=info -o parsable SB0::cpu0

cpuid=16 speed=400 ecache=8
```

EXAMPLE 5 Displaying the Devices on an I/O Board

```
# cfgadm -a -s noheadings,cols=ap_id:info -o parsable IO1

IO1       powered-on assigned
IO1::pci0 device=/devices/saf@0/pci@0,2000 referenced
IO1::pci1 device=/devices/saf@0/pci@1,2000 referenced
```

EXAMPLE 6 Monitoring an Unconfigure Operation

In the following example, the memory sizes are displayed in Kbytes.

EXAMPLE 6 Monitoring an Unconfigure Operation *(Continued)*

```
# cfgadm -c unconfigure -y SB0::memory &
# cfgadm -l -s noheadings,cols=info -o parsable SB0::memory SB1::memory

address=0x0 size=2097152 permanent=752592 target=SB1::memory
     deleted=1273680 remaining=823472
address=0x1000000 size=2097152 source=SB0::memory
```

EXAMPLE 7 Assigning a Slot to a Domain

```
# cfgadm -x assign SB2
```

EXAMPLE 8 Unassigning a Slot from a Domain

```
# cfgadm -x unassign SB3
```

Attributes

See `attributes(5)` for a description of the following attribute:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	`system/library/platform`
Stability	See below.

The interface stability is evolving. The output stability is unstable.

See Also

`cfgadm(1M)`, `devfsadm(1M)`, `ifconfig(1M)`, `mount(1M)`, `pbind(1M)`, `psradm(1M)`, `psrinfo(1M)`, `config_admin(3CFGADM)`, `attributes(5)`

Notes

This section contains information on how to monitor the progress of a memory delete operation. It also contains platform specific information.

Memory Delete Monitoring

The following shell script can be used to monitor the progress of a memory delete operation.

```
# cfgadm -c unconfigure -y SB0::memory &
# watch_memdel SB0

#!/bin/sh
# This is the watch_memdel script.

if [ -z "$1" ]; then
        printf "usage:  %s board_id\n" ‘basename $0‘
        exit 1
fi

board_id=$1

cfgadm_info=’cfgadm -s noheadings,cols=info -o parsable’
```

```
eval `$cfgadm_info $board_id::memory`

if [ -z "$remaining" ]; then
        echo no memory delete in progress involving $board_id
        exit 0
fi

echo deleting target $target

while true
do
        eval `$cfgadm_info $board_id::memory`

        if [ -n "$remaining" -a "$remaining" -ne 0 ]
        then
                echo $deleted KBytes deleted, $remaining KBytes remaining
                remaining=
        else
                echo memory delete is done
                exit 0
        fi
        sleep 1
done
exit 0
```

Sun Enterprise 10000 Platform Notes

The following syntax is used to refer to attachment points on the Sun Enterprise 10000 system:

board::*component*

...where *board* refers to the system board; and *component* refers to the individual component. System boards can range from `SB0` (zero) to `SB15`. A maximum of sixteen system boards are available.

The DR 3.0 model running on a Sun Enterprise 10000 domain supports a limited subset of the functionality provided by the `cfgadm_sbd` plugin. The only supported operation is to view the status of attachment points in the domain. This corresponds to the `-l` option and all of its associated options.

Attempting to perform any other operation from the domain will result in an error that states that the operation is not supported. All operations to add or remove a system board must be initiated from the System Service Processor.

Sun Fire High-End System Platform Notes

The following syntax is used to refer to attachment points on the Sun Fire high-end systems:

board::*component*

where *board* refers to the system board or I/O board; and *component* refers to the individual component.

Depending on the system's configuration, system boards can range from `SB0` (zero) through `SB17`, and I/O boards can range from `IO0` (IO zero) through `IO17`. (A maximum of eighteen system and I/O boards are available).

The `-t` and `-x` options behave differently on the Sun Fire high-end system platforms. The following list describes their behavior:

`-t`
: The system controller uses a CPU to test system boards by running `LPOST`, sequenced by the `hpost` command. To test I/O boards, the driver starts the testing in response to the `-t` option, and the test runs automatically without user intervention. The driver unconfigures a CPU and a stretch of contiguous physical memory. Then, it sends a command to the system controller to test the board. The system controller uses the CPU and memory to test the I/O board from inside of a transaction/error cage. You can only use CPUs from system boards (not MCPU boards) to test I/O boards.

`-x assign | unassign`
: In the Sun Fire high-end system administration model, the platform administrator controls the platform hardware through the use of an available component list for each domain. This information is maintained on the system controller. Only the platform administrator can modify the available component list for a domain.

  The domain administrator is only allowed to assign or unassign a board if it is in the available component list for that domain. The platform administrator does not have this restriction, and can assign or unassign a board even if it is not in the available component list for a domain.

Sun Fire 15K Component Types

The following are the names and descriptions of the component types:

`cpu`
: CPU

`io`
: I/O device

`memory`
: Memory

Note: An operation on a memory component affects all of the memory components on the board.

Sun Fire Midrange Systems Platform Notes

References to attachment points are slightly different on Sun Fire midrange servers such as the 6800, 4810, 4800, and 3800 systems than on the Sun Fire high-end systems. The following syntax is used to refer to attachment points on Sun Fire systems other than the Sun Fire 15K:

`N#.`*board*`::`*component*

where **`N#`** refers to the node; *board* refers to the system board or I/O board; and *component* refers to the individual component.

Depending on the system's configuration, system boards can range from `SB0` through `SB5`, and I/O boards can range from `IB6` through `IB9`. (A maximum of six system and four I/O boards are available).

Sun Fire Midrange System Component Types

The following are the names and descriptions of the component types:

`cpu`
CPU

`pci`
I/O device

`memory`
Memory

Note: An operation on a memory component affects all of the memory components on the board.

Name cfgadm_scsi – SCSI hardware specific commands for cfgadm

Synopsis

```
/usr/sbin/cfgadm [-f] [-y | -n] [-v] [-o hardware_option]
     -c function ap_id...

/usr/sbin/cfgadm [-f] [-y | -n] [-v] [-o hardware_option]
     -x hardware_function ap_id...

/usr/sbin/cfgadm [-v] [-a] [-s listing_option] [-o hardware_option]
     [-l [ap_id | ap_type ... ]]

/usr/sbin/cfgadm [-v] [-o hardware_option] -t ap_id...

/usr/sbin/cfgadm [-v] [-o hardware_option] -h [ap_id]...
```

Description The SCSI hardware specific library `/usr/lib/cfgadm/scsi.so.1` provides the functionality for SCSI hot-plugging through the `cfgadm(1M)` command. `cfgadm` operates on attachment points, which are locations in the system where hardware resources can be dynamically reconfigured. Refer to `cfgadm`(1M) for information regarding attachment points.

For SCSI hot-plugging, each SCSI controller is represented by an attachment point in the device tree. In addition, each SCSI device is represented by a dynamic attachment point. Attachment points are named through `ap_ids`. Two types of `ap_ids` are defined: logical and physical. The physical `ap_id` is based on the physical pathname, whereas the logical `ap_id` is a shorter more user-friendly name. For SCSI controllers, the logical `ap_id` is usually the corresponding disk controller number. For example, a typical logical `ap_id` would be `c0`.

SCSI devices are named relative to the controller `ap_id`. Thus if a disk device is attached to controller `c0`, its `ap_id` can be:

```
c0::dsk/c0t0d0
```

where `dsk/c0t0d0` identifies the specific device. In general, the device identifier is derived from the corresponding logical link for the device in `/dev`. For example, a SCSI tape drive logical `ap_id` could be `c0::rmt/0`. Here `c0` is the logical `ap_id` for the `SCSI` controller and `rmt/0` is derived from the logical link for the tape drive in `/dev/rmt`. If an identifier can not be derived from the link in `/dev`, a unique identifier will be assigned to it. For example, if the tape device has no link in `/dev`, it can be assigned an `ap_id` of the form `c0::st3` where `st3` is a unique internally generated identifier.

When a controller is capable of supporting the Solaris I/O multipathing feature (formerly known as MPxIO or the Sun StorEdge Traffic Manager [STMS]) and is enabled, the controller provides a path to a `scsi_vhci(7D)` multipath device. If a device attached to such controller is supported by `scsi_vhci(7D)` its `ap_id` can be:

```
c0::0,0
```

...where `0,0` uniquely identifies the target and logical unit information. The `Type` field for a path of such `ap_ids` indicates if it represent a path to the `scsi_vhci(7D)` multipath devices, along with the type of device that is connected to through the path.

A simple listing of attachment points in the system will include attachment points at SCSI controllers but not SCSI devices. Use the `-a` flag to the list option (`-l`) to list SCSI devices as well. For example:

```
# cfgadm -l
  Ap_Id             Type         Receptacle     Occupant     Condition
  c0                scsi-bus     connected      configured   unknown
  sysctrl0:slot0    cpu/mem      connected      configured   ok
  sysctrl0:slot1    sbus-upa     connected      configured   ok
```

To list SCSI devices in addition to SCSI controllers:

```
# cfgadm -al

Ap_Id                Type         Receptacle   Occupant     Condition
c0                   scsi-bus     connected    configured   unknown
c0::dsk/c0t14d0      disk         connected    configured   unknown
c0::dsk/c0t11d0      disk         connected    configured   unknown
c0::dsk/c0t8d0       disk         connected    configured   unknown
c0::dsk/c0t0d0       disk         connected    configured   unknown
c0::rmt/0            tape         connected    configured   unknown
sysctrl0:slot0       cpu/mem      connected    configured   ok
sysctrl0:slot1       sbus-upa     connected    configured   ok
```

If the controller `c0` was enabled with Solaris I/O multipathing and the connected disk and tape devices are supported by Solaris I/O multipathing the output would be:

```
# cfgadm -al

Ap_Id               Type         Receptacle   Occupant     Condition
c0                  scsi-bus     connected    configured   unknown
c0::11,0            disk-path    connected    configured   unknown
c0::14,0            disk-path    connected    configured   unknown
c0::8,0             disk-path    connected    configured   unknown
c0::0,0             disk-path    connected    configured   unknown
c0::a.0             tape-path    connected    configured   unknown
sysctrl0:slot0      cpu/mem      connected    configured   ok
sysctrl0:slot1      sbus-upa     connected    configured   ok
```

Refer to `cfgadm(1M)` for more information regarding listing attachment points. The receptacle and occupant state for attachment points at the SCSI controller have the following meanings:

`empty`
: not applicable

`disconnected`
: bus quiesced (I/O activity on bus is suspended)

`connected`
: bus active

`configured`
one or more devices on the bus is configured

`unconfigured`
no device on the bus is configured

The corresponding states for individual SCSI devices are:

`empty`
not applicable

`disconnected`
bus to which the device is attached is quiesced

`connected`
bus to which device is attached is active

`configured`
device or path to a multipath SCSI device is configured

`unconfigured`
device or path to a multipath SCSI device is not configured

Options `cfgadm` defines several types of operations besides listing (`-l`).These operations include testing, (`-t`), invoking configuration state changes, (`-c`), invoking hardware specific functions (`-x`), and obtaining configuration administration help messages (`-h`).

`-c` *function*
The following generic commands are defined for the SCSI hardware specific library:

For SCSI controller attachment points, the following configuration state change operations are supported:

`connect`
Unquiesce the SCSI bus.

`disconnect`
Quiesce the bus (suspend I/O activity on bus).

Incorrect use of this command can cause the system to hang. See `NOTES`.

`configure`
Configure new devices on SCSI bus.

`unconfigure`
Unconfigure all devices connected to bus.

The following generic commands are defined for SCSI devices and for paths to multipath SCSI devices:

`configure`
Configure a specific device or a specific path to a multipath SCSI device.

`unconfigure`
Unconfigure a specific device or a specific path to a multipath SCSI device.

`-f`
When used with the `disconnect` command, forces a quiesce of the SCSI bus, if supported by hardware.

Incorrect use of this command can cause the system to hang. See `NOTES`.

`-h` *ap_id*
SCSI specific help can be obtained by using the help option with any SCSI attachment point.

`-o` *hardware_option*
No hardware specific options are currently defined.

`-s` *listing_option*
Attachment points of class `scsi` can be listed by using the `select` sub-option. Refer to the `cfgadm(1M)` man page for additional information.

`-t` *ap_id*
No test commands are available at present.

`-x` *hardware_function*
Some of the following commands can only be used with SCSI controllers and some only with SCSI devices.

In the following, *controller_ap_id* refers to an `ap_id` for a SCSI controller, for example, `c0`. *device_ap_id* refers to an `ap_id` for a SCSI device, for example: `c0::dsk/c0dt3d0`.

The following hardware specific functions are defined:

`insert_device` *controller_ap_id*
Add a new device to the SCSI controller, *controller_ap_id*.

This command is intended for interactive use only.

`remove_device` *device_ap_id*
Remove device *device_ap_id*.

This command is intended for interactive use only.

`replace_device` *device_ap_id*
Remove device *device_ap_id* and replace it with another device of the same kind.

This command is intended for interactive use only.

`reset_device` *device_ap_id*
Reset *device_ap_id*.

`reset_bus` *controller_ap_id*
Reset bus *controller_ap_id* without resetting any devices attached to the bus.

`reset_all` *controller_ap_id*
Reset bus *controller_ap_id* and all devices on the bus.

`locator` [`=on`|`off`] *device_ap_id*
Sets or gets the hard disk locator LED, if it is provided by the platform. If the [on|off] suboption is not set, the state of the hard disk locator is printed.

`led`[`=LED,mode=on`|`off`|`blink`] *device_ap_id*
If no sub-arguments are set, this function print a list of the current LED settings. If sub-arguments are set, this function sets the mode of a specific LED for a slot.

Examples

EXAMPLE 1 Configuring a Disk

The following command configures a disk attached to controller `c0`:

```
# cfgadm -c configure c0::dsk/c0t3d0
```

EXAMPLE 2 Unconfiguring a Disk

The following command unconfigures a disk attached to controller `c0`:

```
# cfgadm -c unconfigure c0::dsk/c0t3d0
```

EXAMPLE 3 Adding a New Device

The following command adds a new device to controller `c0`:

```
# cfgadm -x insert_device c0
```

The system responds with the following:

```
Adding device to SCSI HBA: /devices/sbus@1f,0/SUNW,fas@e,8800000
This operation will suspend activity on SCSI bus c0
Continue (yes/no)?
```

Enter:

```
y
```

The system responds with the following:

```
SCSI bus quiesced successfully.
It is now safe to proceed with hotplug operation.
Enter y if operation is complete or n to abort (yes/no)?
```

Enter:

```
y
```

EXAMPLE 4 Replacing a Device

The following command replaces a device attached to controller `c0`:

```
# cfgadm -x replace_device c0::dsk/c0t3d0
```

EXAMPLE 4 Replacing a Device *(Continued)*

The system responds with the following:

```
Replacing SCSI device: /devices/sbus@1f,0/SUNW,fas@e,8800000/sd@3,0
This operation will suspend activity on SCSI bus: c0
Continue (yes/no)?
```

Enter:

```
y
```

The system responds with the following:

```
SCSI bus quiesced successfully.
It is now safe to proceed with hotplug operation.
Enter y if operation is complete or n to abort (yes/no)?
```

Enter:

```
y
```

EXAMPLE 5 Encountering a Mounted File System While Unconfiguring a Disk

The following command illustrates encountering a mounted file system while unconfiguring a disk:

```
# cfgadm -c unconfigure c1::dsk/c1t0d0
```

The system responds with the following:

```
cfgadm: Component system is busy, try again: failed to offline:
/devices/pci@1f,4000/scsi@3,1/sd@1,0
     Resource             Information
------------------  -------------------------
/dev/dsk/c1t0d0s0   mounted filesystem "/mnt"
```

EXAMPLE 6 Displaying the Value of the Locator for a Disk

The following command displays the value of the locator for a disk. This example is specific to the SPARC Enterprise Server family:

```
# cfgadm -x locator c0::dsk/c0t6d0
```

The system responds with the following:

```
Disk                    Led
c0t6d0                  locator=on
```

EXAMPLE 7 Setting the Value of the Locator for a Disk

The following command sets the value of the locator for a disk. This example is specific to the SPARC Enterprise Server family:

EXAMPLE 7 Setting the Value of the Locator for a Disk *(Continued)*

```
# cfgadm -x locator=off c0::dsk/c0t6d0
```

The system does not print anything in response.

EXAMPLE 8 Configuring a Path to a Multipath SCSI Disk

The following command configures a path connected through controller `c0`:

```
# cfgadm -c configure c0::2,0
```

EXAMPLE 9 Unconfiguring a Path to a Multipath SCSI Disk

The following command unconfigures a path connected through controller `c0`:

```
# cfgadm -c unconfigure c0::2,0
```

Files

`/usr/lib/cfgadm/scsi.so.1`
hardware-specific library for generic SCSI hot-plugging

`/usr/platform/SUNW,SPARC-Enterprise/lib/cfgadm/scsi.so.1`
platform-specific library for generic SCSI hot-plugging

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/library

See Also `cfgadm(1M)`, `luxadm(1M)`, `config_admin(3CFGADM)`, `libcfgadm(3LIB)`, `attributes(5)`, `scsi_vhci(7D)`

Notes The `disconnect` (quiesce) operation is not supported on controllers which control disks containing critical partitions such as `root` (`/`), `/usr`, `swap`, or `/var`. The `disconnect` operation should not be attempted on such controllers. Incorrect usage can result in a system hang and require a reboot.

When a controller is in the disconnected (quiesced) state, there is a potential for deadlocks occurring in the system. The `disconnect` operation should be used with caution. A controller should be kept in the disconnected state for the minimum period of time required to accomplish the DR operation. The `disconnect` command is provided only to allow the replacement of the SCSI cables while the system is running. It should not be used for any other purpose. The only fix for a deadlock (if it occurs) is to reboot the system.

Hotplugging operations are not supported by all SCSI controllers.

Warnings The connectors on some SCSI devices do not conform to SCSI hotplug specifications. Performing hotplug operations on such devices can cause damage to the hardware on the SCSI bus. Refer to your hardware manual for additional information.

Name cfgadm_sdcard – SD/MMC hardware-specific commands for cfgadm

Synopsis

```
/usr/sbin/cfgadm  [-f] [-y | -n] [-o hardware_options]
     -c function ap_id[...]

/usr/sbin/cfgadm  [-f] [-y | -n] [-o hardware_options]
     -c hardware_function ap_id[...]

/usr/sbin/cfgadm  [-v] [-a] [-s listing_options]
     [-o hardware_options] [-l [ap_id | ap_type[...]]]

/usr/sbin/cfgadm  [-v] [-o hardware_options] -h [ap_id[...]]
```

Description The Secure Digital (SD) and MultiMediaCard (MMC) hardware specific library, `/usr/lib/cfgadm/sdcard.so.1`, provides the functionality for SD/MMC hot-plugging through the `cfgadm(1M)` command. The `cfgadm` command operates on attachment points, which are locations in the system where hardware resources can be dynamically reconfigured. See `cfgadm(1M)` for information on attachment points.

Each SD/MMC slot is represented by an attachment point in the device tree. Card devices that are connected and configured in the system are shown as attachment point name extensions. The terms "attachment point" and "SD/MMC slot" are used interchangeably throughout this manpage.

Attachment points are named through ap_ids. All SD/MMC attachment points consist of a string in the following form:

```
sdcardX/[S][::dsk/cXtYd0]
```

Where:

X is the SD/MMC controller number.

S is the slot number on the controller (0 to 8).

`dev/cXtYd0` identifies the inserted memory card.

Y is a target number.

In general, the device identifier is derived from the corresponding logical link for the device in `/dev`. Because only one LUN (LUN 0) is supported by the SD/MMC device, the "d" component of the device string will always have number 0 (zero). For example, the logical *ap_id* of slot 4 of SD/MMC controller 2 would be *sdcard2/4*. If the SD/MMC media card is inserted in this attachment point and the device is configured, the *ap_id* might be *sdcard2/4::dsk/c2t0d0*.

The c*X*t*Y*d*0* string identifying a device has one-to-one correspondence to the device attachment point.

A simple listing of attachment points in the system includes all SD/MMC device slots and attached devices. For example:

```
#cfgadm -l
Ap_Id                     Type       Receptacle   Occupant    Condition
 sdcard0/0::dev/c2t0d0    sdcard      connected    configured     ok
 sata0/1::dev/c0t1d0      disk        connected    configured     ok
 sata0/2::dev/c0t2d0      cd-dvd      connected    configured     ok
 sata0/3                  sata-port   empty        unconfigured   ok
 usb0/1                   unknown     empty        unconfigured   ok
 usb0/2                   unknown     empty        unconfigured   ok
```

See `cfgadm(1M)` for more information regarding listing of attachment points.

The receptacle state for an attachment point at the SD/MMC slot has the following meanings:

empty
: The SD/MMC slot is powered-on and enabled. No device presence was detected for this slot.

disconnected
: The SD/MMC slot is not enabled, or the SD/MMC device presence was detected but no communication with the device was established, or the slot has failed.

connected
: The SD/MMC device is detected in the slot and device communication is established.

The occupant (device inserted in the SD/MMC slot) state has the following meanings:

configured
: The attached SD/MMC device is configured and ready to use by the operating system.

unconfigured
: No device is attached, or the SD/MMC device inserted in the SD/MMC slot is not yet configured. To configure, run the command **`cfgadm -c configure ap_id`**.

The attachment point (SD/MMC slot) condition has the following meanings:

ok
: The SD/MMC slot is powered-on, enabled and ready for use.

failed
: The SD/MMC slot failed. It may be disabled and/or powered-off by the system. It is unusable and its condition is unknown. The failure may be due to the device inserted in the slot.

unknown
: The SD/MMC slot is disabled and its condition is unknown.

Options The `cfgadm` command defines several types of operations besides listing (`-l`). These operations include invoking configuration state changes (`-c`), invoking hardware specific functions (`-x`), and obtaining configuration administration help messages (`-h`).

`-c`: For SD/MMC slot attachment points, the following configuration state change operations are supported:

connect
: Enable (activate) the SD/MMC slot and establish the communication with an attached device. This operation implies powering-on the slot if necessary.

disconnect
: Unconfigure the inserted device if it is not already unconfigured and disable (deactivate) the SD/MMC slot. A subsequent "connect" command enables SD/MMC slot operation but does not bring a device to the "configured" state.

The following state change operations are supported for an SD/MMC card inserted in to the SD/MMC slot:

configure
: Configure new device for use by the operating system if it is not already configured. This command also implies connect operation, if necessary.

unconfigure
: Unconfigure the device inserted in the SD/MMC slot if it is not already unconfigured.

`-f` : Not supported.

`-h` *ap_id*: SD/MMC specific help can be obtained by using the help option with any SD/MMC attachment point.

`-l` [-v]: The -l option works as described in `cfgadm(1M)`. When paired with the -v option, the "Information" field contains the following SD/MMC-specific information:

Mod: product model string
Rev: product revision number (major.minor)
Date: month and year of manufacture
SN: product serial number (hexadecimal)

`-o` hardware_options — No hardware specific options are currently defined.

`-s` listing_options: Attachment points of class SD/MMC can be listed by using the select suboption. See `cfgadm(1M)`.

`-t` *ap_id*: Self-test functionality. Not supported by SD/MMC slots.

`-x` *hardware_function*: Perform hardware specific function. sdcard_reset_slot *ap_id*indicates reset of the SD/MMC slot specified by *ap_id*. If an SD/MMC device is inserted in the slot, it is also reset.

`-v`: Execute in verbose mode.

Examples Example 1 – Configuring an SD/MMC card:

The following command configures a card attached to SD/MMC controller 0, slot 0. It should be issued only when there is a device inserted in the SD/MMC slot.

```
# cfgadm -c configure sdcard0/0
```

Example 2 – Unconfiguring an SD/MMC card:

The following command unconfigures a card inserted in SD/MMC controller 0, slot 3:

```
# cfgadm -c unconfigure sdcard0/3
```

Example 3 — Encountering a mounted file system while unconfiguring a disk:

The following command illustrates encountering a mounted file system while unconfiguring a disk:

```
# cfgadm -c unconfigure sdcard1/5::dsk/c01t35d0
```

The system responds with the following:

```
cfgadm: Component system is busy, try again: failed to offline:
      /devices/pci@0,0/pci8086,244e@1e/pci1095,3124@1/sd@5,0
          Resource             Information
      ------------------  -------------------------
      /dev/dsk/c1t5d0s0   mounted filesystem "/mnt"
```

Files /usr/lib/cfgadm/sdcard.so.1
Hardware specific library for generic SD/MMC hot plugging.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/library

See Also `cfgadm(1M)`, `config_admin(3CFGADM)`, `libcfgadm(3LIB)`, `attributes(5)`, `sda(7D)`, `sdcard(7D)`, `sdhost(7D)`

Notes Under normal operation, SD/MMC media cards are automatically configured when a card is inserted. Most administrators do not find it necessary to use this command under normal use.

Removing an SD/MMC card without first unconfiguring it may result in data loss if the device is being written to when it's being removed. Devices that are mounted read-only can be safely removed at any time.

Devices that have filesystems other than `pcfs(7FS)` on them should always be explicitly unconfigured before removal.

Name cfgadm_shp – PCI Express and Standard PCI Hotplug hardware-specific commands for cfgadm

Synopsis

```
/usr/sbin/cfgadm [-f] [-y | -n] [-v]
     [-o hardware_options] -c function ap_id [ap_id]

/usr/sbin/cfgadm [-f] [-y | -n] [-v]
     [-o hardware_options] -x hardware_function ap_id [ap_id]

/usr/sbin/cfgadm [-v] [-s listing_options]
     [-o hardware_options] -x hardware_function ap_id [ap_id]

/usr/sbin/cfgadm [-v] [-o hardware_options] -tap_id [ap_id]

/usr/sbin/cfgadm [-v] [-o hardware_function]-h [ap_id | ap_type]
```

Description The PCI Express and Standard PCI Hotplug hardware-specific library, `/usr/lib/cfgadm/shp.so.1`, provides support for hotplugging PCI Express and Standard PCI Hotplug adapter cards into the respective hotpluggable slots in a system that is hotplug-capable, through the `cfgadm` command (see `cfgadm(1M)`). Support for the rest PCI Hotplug adapter cards (other than PCI Express and Standard PCI Hotplug cards) are provided by `cfgadm_pci` library (see `cfgadm_pci(1M)`). Hotplug administrative models between PCI Express Hotplug and Standard PCI Hotplug remain the same except where noted in this man page.

For PCI hotplug, each hotplug slot on a specific PCI bus is represented by an attachment point of that PCI bus.

An attachment point consist of two parts: a receptacle and an occupant. The receptacle under PCI hotplug is usually referred to as the physical hot pluggable slot; and the occupant is usually referred to as the PCI adapter card that plugs into the slot.

Attachment points are named through `ap_id`s. There are two types of `ap_id`s: logical and physical. The physical `ap_id` is based on the physical pathname, for example:

```
/devices/pci@7c,0/pci10de,5d@d:pcie2
```

Whereas the logical `ap_id` is a shorter, more user-friendly name, for example, `pcie2`. The `ap_type` for Hotplug PCI is `pci`.

Note that the `ap_type` is not the same as the information in the `Type` field.

PCI Express ap_id Naming For attachment points located in a PCI Express hierarchy (that is, the parent or an ancestor is a PCI Express device), including attachment points that are not PCI Express devices themselves, the naming scheme shown below is used.

Grammar:

`APID` : *absolute-slot-path*
Fundamental term.

absolute-slot-path : *slot-path*[:*slot-path*[:*slotpath* ...]]
...where *fru-id* indicates the chassis FRU, if any, containing the *slot-id*.

fru-id : *fru-type*[*serialid#*]
...where *fru-type* is "iob" for a PCI Express expansion chassis, followed by its serial number *serialid#*, if available

slot-id : *slot-name* | *device-type physical-slot#* | \
nexus-driver-name nexus-driver-instance.\
device-type pci-device-number
...where *slot-name* is a name assigned by the platform or hardware itself. *device-type* is either `pcie` for PCI Express devices or `pci` for PCI devices. *nexus-driver-name* is the driver name for the device component; *physical-slot#* is the hardware slot number; and *pci-device-number* is the PCI device number in standard PCI nomenclature.

First, an *absolute-slot-path* is constructed that attempts to describe the attachment point's topological location in more physically identifiable terms for the user. This *absolute-slot-path* consists of *slot-path* components each separated by a `:` (colon). The leaf or leftmost *slot-path* component describes the device of the attachment point itself, while its right-adjacent *slot-path* component up to the rightmost or topmost *slot-path* component describes the parent up to the root devices, respectively.

Each *slot-path* consists of a *slot-id* optionally preceded by a *fru-id*, which identifies an expansion chassis containing the device described by *slot-id* (detailed below). *fru-id* consists of *fru-type* followed by an optional *serialid#*. *fru-type* is "iob" for PCI Express expansion chassis types, while *serialid#* is either a 64-bit hexadecimal number indicating a raw serial number obtained from the expansion chassis hardware, or an upper-case, ASCII four-character sequence for a Sun-branded expansion chassis.

Each slot-id consists of one of three possible forms:

slot-id form (1)
slot-names

slot-id form (2)
device-type physical-slot#

slot-id form (3)
nexus-driver-name nexus-driver-instance device-type pci-device-number

The precedence of which form to select flows from the lowest form number to the highest form number, or from top to bottom as described above. If a form cannot be successfully constructed, then the next numerically higher form is attempted.

The *slot-names* in *slot-id* form (1) is taken from the `slot-names` property of the corresponding node in the device tree and is a name assigned by hardware or the platform. This format is not predefined or established.

In *slot-id* form (2), *device-type* indicates the device type of the component's slot, and is either `pcie` for PCI Express or `pci` for PCI, while *physical-slot#*, taken from the `physical-slot#` property of its corresponding device node, indicates the hardware slot number of the component.

slot-id form (3) is used when all other forms cannot be successfully constructed, and is considered to be the default form. *nexus-driver-name* is the component's driver name; *nexus-driver-instance* is this driver's instance; *device-type* is the same as described in form (2); *pci-device-number* is the PCI device number as described and used for device configuration cycles in standard PCI nomenclature.

In summary of the *slot-path* component, expanding the optional FRU component that might precede it, *slot-path* will consist one of the following forms in order:

(1) [iob[*serialid#*].]
slot-names

(2) [iob[*serialid#*].]
device_type physical_slot#

(2) [iob[*serialid#*].]
nexus-driver-name nexus-driver-instance.

device_type pci-device-number

Lastly, the final form of the actual `ap_id` name used in `cfgadm` is decided as follows, specified in order of precedence:

`ap_id` form (1)
If the *absolute-slot-path* can fit within the fixed length limit of `cfgadm`'s `ap_id` field, then *absolute-slot-path* itself is used

`ap_id` form (2)
(*absolute-slot-path* exceeds the `ap_id` length limit) If the last *slot_path* component is contained within an expansion chassis, and it contains a *serialid#*, then the last *slot_path* component is used. The requirement for a *serialid#* in this form is to ensure a globally unique `ap_id`.

`ap_id` form (3)
(*absolute-slot-path* exceeds the `ap_id` length limit) The default form, *slot-id* form (3), of the last *slot_path* component is used.

Whichever final `ap_id` name is used, the *absolute-slot-path* is stored in the Information (`info`) field which can be displayed using the `-s` or `-v` options. This information can be used to physically locate any `ap_id`s named using `ap_id` form (2) or `ap_id` form (3). The *absolute-slot-path* is transformed slightly when stored in the information field, by the replacement of a colon (`:`) with forward slashes (`/`) to more closely denote a topological context. The *absolute-slot-path* can include *slot-path* components that are not hotpluggable above the leaf or rightmost *slot-path* component up to the onboard host slot.

See the Examples section for a list of hotpluggable examples.

Options The following options are supported:

`-c` *function*
The following functions are supported for PCI hotpluggable slots:

`configure`
Configure the PCI device in the slot to be used by Solaris.

`connect`
Connect the slot to PCI bus.

`disconnect`
Disconnect the slot from the PCI bus.

`insert`
Not supported.

`remove`
Not supported.

`unconfigure`
Logically remove the PCI device's resources from the system.

`-f`
Not supported.

`-h` *ap_id* | *ap_type*
Display PCI hotplug-specific help message.

`-l` *list*
List the values of PCI Hot Plug slots.

`-o` *hardware_options*
No hardware specific options are currently defined.

`-s` *listing_options*
Same as the generic `cfgadm(1M)`.

`-t` *ap_id*
This command is only supported on platforms that support testing capability on the slot.

`-v`
Execute in verbose mode.

When the `-v` option is used with the `-l` option, the `cfgadm` command outputs information about the attachment point. For attachment points located in a PCI Express hierarchy, the Information field will contain the attachment point's absolute slot path location, including any hardware– or platform-specific labeling information for each component in the slot path. Each component in the slot path will be separated by a `/` (forward slash). See "PCI Express `ap_id` Naming," above. For PCI Hot Plug attachment points not located in a PCI

Express hierarchy, see `cfgadm_pci(1M)`. The information in the `Type` field is printed with or without the `-v` option. The occupant `Type` field will describe the contents of the slot. There are two possible values:

`unknown`
The slot is empty. If a card is in the slot, the card is not configured or there is no driver for the device on the card.

subclass/*board*
The card in the slot is either a single-function or multi-function device.

subclass is a string representing the subclass code of the device, for example, `SCSI`, `ethernet`, `pci-isa`, and so forth. If the card is a multi-functional device, `MULT` will get displayed instead.

board is a string representing the board type of the device. For example, `hp` is the string used for a PCI Hot Plug adapter.

`-x` *hardware_function*
Perform hardware-specific function. These hardware-specific functions should not normally change the state of a receptacle or occupant.

The following *hardware_function* is supported:

`led=`[*led_sub_arg*]`,mode=`[*mode_sub_arg*]
Without subarguments, display a list of the current LED settings. With subarguments, set the mode of a specific LED for a slot.

Specify *led_sub_arg* as `fault`, `power`, `attn`, or `active`.

Specify *mode_sub_arg* as `on`, `off`, or `blink`.

For PCI Express, only the `power` and `attn` LEDs are valid and only the state of the `attn` LED can be changed.

Changing the state of the LED does not change the state of the receptacle or occupant. Normally, the LEDs are controlled by the hotplug controller, no user intervention is necessary. Use this command for testing purposes.

Caution – Changing the state of the LED can misrepresent the state of occupant or receptacle.

The following command displays the values of LEDs:

```
example# cfgadm -x led pcie2
Ap_Id              Led
pcie2     power=on,fault=off,active=off,attn=off
```

The following command sets the `attn` LED to blink to indicate the location of the slot:

```
example# cfgadm -x led=attn,mode=blink pcie2
```

Examples

EXAMPLE 1 Displaying the Value of Each Slot

The following command displays the values of each slot:

```
example# cfgadm -l
Ap_Id            Type         Receptacle   Occupant       Condition
c0               scsi-bus     connected    configured     unknown
c1               scsi-bus     connected    unconfigured   unknown
c2               scsi-bus     connected    unconfigured   unknown
pcie7            etherne/hp   connected    configured     ok
pcie8            unknown      empty        unconfigured   unknown
pcie9            fibre/hp     connected    configured     ok
```

EXAMPLE 2 Replacing a Card

The following command lists all DR-capable attachment points:

```
example# cfgadm
Type             Receptacle   Occupant     Condition
c0               scsi-bus     connected    configured     unknown
c1               scsi-bus     connected    unconfigured   unknown
c2               scsi-bus     connected    unconfigured   unknown
pcie7            etherne/hp   connected    configured     ok
pcie8            unknown      empty        unconfigured   unknown
pcie9            fibre/hp     connected    configured     ok
```

The following command unconfigures and electrically disconnects the card identified by `pcie7`:

```
example# cfgadm -c disconnect pcie7
```

The change can be verified by entering the following command:

```
example# cfgadm pcie7
Ap_Id     Type         Receptacle   Occupant     Condition
pcie7     unknown      disconnected unconfigured unknown
```

At this point the card can be swapped. The following command electrically connects and configures the replacement card:

```
example# cfgadm -c configure pcie7
```

The change can be verified by entering the following command:

```
example# cfgadm pcie7
Ap_Id     Type         Receptacle   Occupant     Condition
pcie7     etherne/hp   connected    configured   ok
```

EXAMPLE 3 Interpreting ApIds in a PCI Express Topology

The following command shows a listing for a topology with both PCI Express and PCI attachment points in an I/O expansion chassis connected to hotpluggable slots at the host level:

EXAMPLE 3 Interpreting ApIds in a PCI Express Topology *(Continued)*

```
example# cfgadm -s cols=ap_id:info
Ap_Id                          Information
iou#0-pci#0                    Location: iou#0-pci#0
iou#0-pci#1                    Location: iou#0-pci#1
iou#0-pci#1:iob.pci3           Location: iou#0-pci#1/iob.pci3
iou#0-pci#1:iob.pci4           Location: iou#0-pci#1/iob.pci4
iou#0-pci#2                    Location: iou#0-pci#2
iou#0-pci#2:iob58071.pcie1     Location: iou#0-pci#2/iob58071.pcie1
iou#0-pci#2:iob58071.special   Location: iou#0-pci#2/iob58071.special
iou#0-pci#3                    Location: iou#0-pci#3
iou#0-pci#3:iobBADF.pcie1      Location: iou#0-pci#3/iobBADF.pcie1
iou#0-pci#3:iobBADF.pcie2      Location: iou#0-pci#3/iobBADF.pcie2
iou#0-pci#3:iobBADF.pcie3      Location: iou#0-pci#3/iobBADF.pcie3
iou#0-pci#3:iobBADF.pci1       Location: iou#0-pci#3/iobBADF.pci1
iou#0-pci#3:iobBADF.pci2       Location: iou#0-pci#3/iobBADF.pci2
```

In this example, the `iou#0-pci#[0-3]` entries represents the topmost hotpluggable slots in the system. Because the `iou#`*n*`-pci#`*n* form does not match any of the forms stated in the grammar specification section described above, we can infer that such a name for the base component in this hotplug topology is derived from the platform through the `slot-names` property.

The slots in the preceding output are described as follows:

Slot `iou#0-pci#0`
: This slot is empty or its occupant is unconfigured.

Slot `iou#0-pci#1`
: This slot contains an expansion chassis with two hotpluggable slots, `pci3` and `pci4`. `pci3` and `pci4` represent two PCI slots contained within that expansion chassis with physical slot numbers `3` and `4`, respectively. The expansion chassis in this case does not have or export a serial-id.

Slot `iou#0-pci#2`
: This slot contains a third-party expansion chassis with a hexadecimal serial-id of `58071`. Within that expansion chassis are two hotpluggable slots, `pcie1` and `special`. `pcie1` represents a PCI Express slot with physical slot number `1`. The slot `special` has a label which is derived from the platform, hardware, or firmware.

Slot `iou#0-pci#3`
: This slot contains a Sun expansion chassis with an FRU identifier of `BADF`. This expansion chassis contains three PCI Express slots, `pcie1`, `pcie2`, and `pcie3` with physical slot numbers `1`, `2`, and `3`, respectively; and two PCI slots, `pci1` and `pci2`, with physical slot numbers 1 and 2, respectively.

The following command shows a listing for a topology with both PCI Express and PCI attachment points in an I/O expansion chassis with connected hotpluggable and non-hotpluggable host slots:

EXAMPLE 3 Interpreting ApIds in a PCI Express Topology *(Continued)*

```
example# cfgadm -s cols=ap_id:info
Ap_Id                          Information
Slot1                          Location: Slot1
Slot2:iob4ffa56.pcie1          Location: Slot2/iob4ffa56.pcie1
Slot2:iob4ffa56.pcie2          Location: Slot2/iob4ffa56.pcie2
Slot5:iob3901.pci1             Location: Slot2/iob3901.pci1
Slot5:iob3901.pci2             Location: Slot2/iob3901.pci2
```

In this example, the host system only has one hotpluggable slot, `Slot1`. We can infer that `Slot2` and `Slot5` are not hotpluggable slots because they do not appear as attachment points themselves in `cfgadm`. However, `Slot2` and `Slot5` each contains a third party expansion chassis with hotpluggable slots.

The following command shows a listing for a topology with attachment points that are lacking in certain device properties:

```
example# cfgadm -s cols=ap_id:info
Ap_Id                          Information
px_pci7.pcie0                  Location: px_pci7.pcie0
px_pci11.pcie0                 Location: px_pci11.pcie0
px_pci11.pcie0:iob.pcie1       Location: px_pci11.pcie0/iob.pcie1
px_pci11.pcie0:iob.pcie2       Location: px_pci11.pcie0/iob.pcie2
px_pci11.pcie0:iob.pcie3       Location: px_pci11.pcie0/iob.pcie3
```

In this example, the host system contains two hotpluggable slots, `px_pci7.pcie0` and `px_pci11.pcie0`. In this case, it uses `slot-id` form (3) (the default form) for the base *slot-path* component in the *absolute-slot-path*, because the framework could not obtain enough information to produce other more descriptive forms of higher precedence.

Interpreting right-to-left, attachment point `px_pci7.pcie0` represents a PCI Express slot with PCIdevice number `0` (which does not imply a physical slot number of the same number), bound to nexus driver `px_pci`, instance `7`. Likewise, attachment point `px_pci11.pcie0` represents a PCI Express slot with PCI device number `0` bound to driver instance `11` of `px_pci`.

Under `px_pci11.pcie0` is a third-party expansion chassis without a serial-id and with three hotpluggable PCI Express slots.

The following command shows a listing for a topology with attachment point paths exceeding the `ApId` field length limit:

```
example# cfgadm -s cols=ap_id:info
Ap_Id                          Information
pcie4                          Location: pcie4
pcie4:iobSUNW.pcie1            Location: pcie4/iobSUNW.pcie1
pcie4:iobSUNW.pcie2            Location: pcie4/iobSUNW.pcie2
```

EXAMPLE 3 Interpreting ApIds in a PCI Express Topology *(Continued)*

```
iob8879c3f3.pci1
                Location: pcie4/iobSUNW.pcie2/iob8879c3f3.pci1
iob8879c3f3.pci2
                Location: pcie4/iobSUNW.pcie2/iob8879c3f3.pci2
iob8879c3f3.pci3
                Location: pcie4/iobSUNW.pcie2/iob8879c3f3.pci3
```

In this example, there is only one hotpluggable slot, `pcie4` in the host. Connected under `pcie4` is a Sun expansion chassis with FRU identifier `SUNW`. Nested under PCI Express slot `pcie2` of that expansion chassis (`ApId` `pcie4:iobSUNW.pcie2`) lies another expansion chassis with three hotpluggable PCI slots.

Because the length of the absolute-slot-path form of:

```
pcie4/iobSUNW.pcie2/iob8879c3f3.pci1...3
```

...exceeds the `ApId` field length limit, and the leaf slot-path component is globally unique, `ap_id` form (2) is used, where the leaf *slot-path* component in the *absolute-slot-path* is used as the final `ApId`.

The following command shows a listing for a topology with attachment point paths exceeding the `ApId` field-length limit and lacking enough information to uniquely identify the leaf *slot-id* on its own (for example, missing the serial-id):

```
example# cfgadm -s cols=ap_id:info
Ap_Id                          Information
pcie4                          Location: pcie4
pcie4:iob4567812345678.pcie3   Location: pcie4/iob4567812345678.pcie3
px_pci20.pcie0
                Location: pcie4/iob4567812345678.pcie3/iob.pcie1
px_pci21.pcie0
                Location: pcie4/iob4567812345678.pcie3/iob.pcie2
```

In this example, there is only one hotpluggable slot, `pcie4` in the host. Connected under `pcie4` is a third-party expansion chassis with hexadecimal serial-id `4567812345678`. Nested under the PCI Express slot `pcie3` of that expansion chassis (`ApId` `pcie4:iob4567812345678.pcie3`), lies another third-party expansion chassis without a serial- id and with two hotpluggable PCI Express slots.

Because the length of the absolute-slot-path form of:

```
pcie4/iob4567812345678.pcie3/iob.pcie1...2
```

exceeds the `ApId` field length limit, and the leaf *slot-path* component is not globally unique, `ap_id` form (3) is used. `ap_id` form (2) is where *slot-id* form (3) (the default form) of the leaf *slot-path* component in the *absolute-slot-path* is used as the final `ApId`.

EXAMPLE 3 Interpreting ApIds in a PCI Express Topology *(Continued)*

The default form or `slot-id` form (3) of the leaf component `.../iob.pcie1` represents a PCI Express slot with device number `0`, bound to driver instance `20` of `px_pci`. Likewise, the default form of the leaf component `.../iob.pcie2` represents a PCI Express slot with device number `0`, bound to driver instance `21` of `px_pci`.

Files

`/usr/lib/cfgadm/shp.so.1`
Hardware-specific library for PCI Express and Standard PCI hotplugging.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/library
Interface Stability	Uncommitted

See Also

`cfgadm(1M)`, `cfgadm_pci(1M)`, `hotplugd(1M)`, `config_admin(3CFGADM)`, `libcfgadm(3LIB)`, `attributes(5)`, `smf(5)`

Notes

The `cfgadm_shp` library is dependent on the `hotplug` service, which is managed by `smf(5)` under FMRI:

```
svc:/system/hotplug:default
```

The service must be enabled for the `cfgadm_shp` library to function properly. See `hotplugd(1M)` for details.

Name cfgadm_sysctrl – EXX00 system board administration

Synopsis

```
/usr/sbin/cfgadm -c function [-f]
     [-o disable-at-boot | enable-at-boot] [-n | -y] sysctrl0:slot# ...

/usr/sbin/cfgadm -x quiesce-test sysctrl0:slot#

/usr/sbin/cfgadm -x insert-test | remove-test sysctrl0:slot# ...

/usr/sbin/cfgadm -x set-condition-test=# sysctrl0:slot# ...

/usr/sbin/cfgadm [-l]
     -o disable-at-boot | enable-at-boot sysctrl0:slot# ...
```

Description The `sysctrl` hardware specific library `/usr/platform/sun4u/lib/cfgadm/sysctrl.so.1` provides dynamic reconfiguration functionality for configuring and disconnecting system boards on `E6X00`, `E5X00`, `E4X00`, and `E3X00` systems. You can insert both I/O and CPU boards into a slot on a running system that is configured for Solaris without rebooting. You can also disconnect and remove both types of boards from a running system without rebooting.

System slots appear as attachment points in the device tree, one attachment point for each actual slot in the system chassis. If a board is not in a slot, the receptacle state is `empty`. If a board is powered-off and ready to remove, the receptacle state is `disconnected`. If a board is powered-on and is connected to the system bus, the receptacle state is `connected`.

The occupant state is `unconfigured` when the receptacle state is `empty` or `disconnected`. The occupant state is either `unconfigured` or `configured` when the receptacle state is `connected`.

In the `configured` state the devices on a board are available for use by Solaris. In the `unconfigured` state, the devices on the board are not.

Inserting a board changes the receptacle state from `empty` to `disconnected`. Removing a board changes the receptacle state from `disconnected` to `empty`. Removing a board that is in the `connected` state crashes the operating system and can result in permanent damage to the system.

Options Refer to `cfgadm(1M)` for a more complete description options.

The following options are supported:

`-c` *function* Perform the state change function. Specify *function* as `connect`, `disconnect`, `configure` or `unconfigure`.

configure Change the occupant state to `configure`.

If the receptacle state is `disconnected`, the `configure` function first attempts to connect the receptacle. The `configure` function walks the OBP device tree created as

part of the `connect` function and creates the Solaris device tree nodes, attaching devices as required. For CPU/Memory boards, `configure` adds CPUs to the CPU list in the `powered-off` state. These are visible to the `psrinfo(1M)` and `psradm(1M)` commands. Two memory attachment points are published for CPU/memory boards. Use `mount(1M)` and`ifconfig(1M)` to use I/O devices on the new board. To use CPUs, use `psradm -n` to on-line the new processors. Use `cfgadm_ac(1M)` to test and configure the memory banks.

connect

Change the receptacle state to `connected`.

Changing the receptacle state requires that the system bus be frozen while the bus signals are connected and the board tested. The bus is frozen by running a `quiesce` operation which stops all process activity and suspends all drivers. Because the `quiesce` operation and the subsequent resume can be time consuming, and are not supported by all drivers, the `-x quiesce-test` is provided. While the system bus is frozen, the board being connected is tested by firmware. This operation takes a short time for I/O boards and a significant time for CPU/Memory boards due to CPU external cache testing. This does not provide memory testing. The user is prompted for confirmation before proceeding with the `quiesce`. Use the `-y` or `-n` option to override the prompt. The `connect` operation is refused if the board is marked as

`disabled-at-boot`, unless either the force flag, `-f`, or the enable at boot flag, `-o enable-at-boot`, is given. See `-l`.

disconnect — Change the receptacle state to `disconnected`.

If the occupant state is `configure`, the `disconnect` function first attempts to unconfigure the occupant. The `disconnect` operation does not require a `quiesce` operation and operates quickly. The board is powered-off ready for removal.

unconfigure — Change the occupant state to `unconfigureed`.

Devices on the board are made invisible to Solaris during this process. The I/O devices on an I/O board are removed from the Solaris device tree. Any device that is still in use stops the unconfigure process and be reported as in use. The unconfigure operation must be retried after the device is made non-busy. For CPU/Memory boards, the memory must have been changed to the unconfigured state prior to issuing the board unconfigure operation. The CPUs on the board are off-lined, powered off and removed from the Solaris CPU list. CPUs that have processes bound to them cannot be off-lined. See `psradm`(1M), `psrinfo(1M)`, `pbind(1M)`, and`p_online(2)` for more information on off-lining CPUs.

`-f` — Force a block on connecting a board marked as `disabled-at-boot` in the non-volatile

	disabled-board-list variable. See *Platform Notes:Sun Enterprise 6x00/5x00/4x00/3x00 Systems*
-l	List options. Supported as described in cfgadm(1M)cfgadm(1M). The *type* field can be one of cpu/mem, mem, dual-sbus, sbus-upa, dual-pci, soc+sbus, soc+upa, disk or unknown. The hardware-specific info field is set as follows: [disabled at boot] [non-detachable] [100 MHz capable] For sbus-upa and soc+upa type boards, the following additional information appears first: [single buffered ffb|double buffered ffb|no ffb installed] For disk type boards, the following additional information appears first: {target: # | no disk} {target: # | no disk}
-o disable-at-boot | enable-at-boot	Modify the state of the non—volatile disabled-board-list variable. Use this the -o option in conjunction with the -c *function* or -l option. Use -o enable-at-boot with the -c connect to override a block on connecting a disabled-at-boot board.
-x insert-test | remove-test	Perform a test. Specify remove-test to change the driver state for the specified slot from disconnected to empty without the need for physically removing the board during automated test sequences. Specify insert-test to change the driver state of a slot made to appear empty using the remove-test command to the disconnected state as if it had been inserted.
-x quiesce-test sysctrl0:slot1	Perform a test. Allows the quiesce operation required for board connect operations to be exercised. The execution of this test confirms that, with the current software and hardware configuration, it is possible to quiesce the

system. If a device or process cannot be quiesced, its name is printed in an error message. Any valid board attachment point can be used with this command, but since all systems have a slot1 the given form is recommended.

-x set-condition-test=# — Perform a test.

Allows the condition of a system board attachment point to be set for testing the policy logic for state change commands. The new setting is given as a number indicating one of the following condition values:

```
0       unknown
1       ok
2       failing
3       failed
4       unusable
```

Operands The following operand is supported:

sysctrl0:slot# — The attachment points for boards on `EXX00` systems are published by instance `0` of the `sysctrl` driver (`sysctrl0`). The names of the attachment points are numbered from `slot0` through `slot15`. Specify # as a number between `0` and `15`, indicating the slot number. This form conforms to the logical `ap_id` specification given in `cfgadm(1M)`. The corresponding physical `ap_id`s are listed in the `FILES` section.

Files /usr/platform/sun4u/lib/cfgadm/sysctrl.so.1
Hardware specific library

/devices/central@1f,0/fhc@0,f8800000/clock-board@0,900000:slot*
Attachment Points

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/library/platform

See Also `cfgadm(1M)`, `cfgadm_ac(1M)`, `ifconfig(1M)`, `mount(1M)`, `pbind(1M)`, `psradm(1M)`, `psrinfo(1M)`, `config_admin(3CFGADM)`, `attributes(5)`

Sun Enterprise 6x00, 5x00, 4x00 and 3x00 Systems Dynamic Reconfiguration User's Guide

Platform Notes:Sun Enterprise 6x00/5x00/4x00/3x00 Systems

Notes Refer to the *Sun Enterprise 6x00, 5x00, 4x00 and 3x00 Systems Dynamic Reconfiguration User's Guide* for additional details regarding dynamic reconfiguration of EXX00 system CPU/Memory boards.

Name cfgadm_usb – USB hardware-specific commands for cfgadm

Synopsis

```
/usr/sbin/cfgadm [-f] [-y | -n] [-v] -c function ap_id...

/usr/sbin/cfgadm -f [-y | -n] [-v] [-o hardware_options]
     -x hardware_function ap_id...

/usr/sbin/cfgadm -v [-a] [-s listing_option]
     [-l [ap_id | ap_type...]]

/usr/sbin/cfgadm -v -h [ap_id]...
```

Description The Universal Serial Bus (USB) hardware-specific library `/usr/lib/cfgadm/usb.so.1` provides the functionality for administering USB devices via the `cfgadm`(1M) command. `cfgadm` operates on attachment points. For details regarding attachment points, refer to `cfgadm`(1M).

For USB administration, the only attachment points supported are the ports of hubs attached to the USB bus.

Attachment points are named through attachment point IDs (*ap_ids*). The USB bus is hierarchical, so the *ap_ids* are as well. USB hubs have ports, numbered from `1` to *n*. All USB *ap_ids* consist of a string of the following form:

`usb`*N*`/`*A*`[.`*B*`[.`*C*`[...]]]`

where

N is the *N*th USB host controller on the system,
A is port #*A* on the root (top) hub.
B is port #*B* of the hub plugged into port #*A* of the hub above it.
C is port #*C* of the hub plugged into port #*B* of the hub above it, and so forth.

For example, the first port on the root hub of USB controller `0` (the only controller), has a logical *ap_id*:

```
usb0/1
```

Similarly, the second port on the first external hub plugged into the first port on the root hub of the first USB controller has a logical *ap_id*:

```
usb0/1.2
```

For example, if the *ap_id* is `usb0/1.4.3.4`, it represents port `4` of the hub plugged into port `3` of the hub plugged into port `4` of the hub plugged into port `1` of the root hub of the first USB host controller on the system.

```
example# cfgadm -l
Ap_Id                Type         Receptacle   Occupant     Condition
usb0/1               USB-hub      connected    configured   ok
usb0/2               unknown      empty        unconfigured ok
```

```
usb0/1.1             USB-storage  connected    configured   ok
usb0/1.2             unknown      empty        unconfigured ok
usb0/1.3             unknown      empty        unconfigured ok
usb0/1.4             USB-device   connected    configured   ok
```

USB2.0 chips have one EHCI host USB2.0 host controller and a number of companion USB 1.*x* host controllers (either OHCI or UHCI host controllers).

When a USB2.0 device has been plugged in, it shows up on the EHCI logical ports which might not have a 1 to 1 mapping to external physical port numbers on the system. When a USB1.*x* device is plugged in, the EHCI host controller reroutes the device to a companion host controller and the device shows up on the companion's logical port number.

The mapping of logical port numbers to physical port numbers can get quite complicated. For example:

```
% cfgadm
Ap_Id                Type         Receptacle   Occupant     Condition
c0                   scsi-bus     connected    configured   unknown
usb0/1               usb-mouse    connected    configured   ok
usb0/2               usb-kbd      connected    configured   ok
usb0/3               unknown      empty        unconfigured ok
usb0/4               usb-hub      connected    configured   ok
usb0/4.1             unknown      empty        unconfigured ok
usb0/4.2             unknown      empty        unconfigured ok
usb0/4.3             unknown      empty        unconfigured ok
usb0/4.4             usb-storage  connected    configured   ok
usb1/1               unknown      empty        unconfigured ok
usb1/2               unknown      empty        unconfigured ok
usb1/3               unknown      empty        unconfigured ok
usb2/1               unknown      empty        unconfigured ok
usb2/2               usb-device   connected    configured   ok
usb3/1               unknown      empty        unconfigured ok
usb3/2               unknown      empty        unconfigured ok
usb3/3               unknown      empty        unconfigured ok
usb3/4               unknown      empty        unconfigured ok
usb3/5               unknown      empty        unconfigured ok
```

In this example usb0 is the onboard USB 1.*x* host controller. usb1 and usb2 are companion OHCI USB1.*x* host controllers and usb3 is an EHCI USB2.0 host controller.

The following table shows the somewhat confusing routing for this USB2.0 chip:

```
logical port number        physical port number
-------------------        --------------------
    usb1/1            internal port 1
    usb1/2            external port 1
    usb1/3            external port 3
```

```
usb2/1            internal port 2
usb2/2            external port 2

usb3/1            internal port 1
usb3/2            internal port 2
usb3/3            external port 1
usb3/4            external port 2
usb3/5            external port 3
```

Unfortunately, the exact routing can often only be determined by experimentation.

The receptacle states for attachment points at the USB port have the following meanings:

`connected`
USB port is powered on and enabled. A USB device is plugged in to the port. The device is logically connected to the USB bus.

`disconnected`
USB port is powered on and enabled. A USB device is plugged into the port. The device has been logically disconnected from the USB bus (using the `cfgadm -c disconnect` command).

`empty`
USB port is powered on, but no device is plugged in to it.

The occupant states for devices at USB port attachment points at the USB port have the following meanings:

`configured`
The USB device at the USB port is configured and usable by Solaris.

`unconfigured`
The USB device at the USB port was explicitly off-lined using `cfgadm -c unconfigure`, or was not successfully configured for use with Solaris, for example, having no driver or a device problem.

The attachment point conditions are:

`ok`
Normal state - ready for use.

`failing`
Not used.

`failed`
Not used.

`unusable`
The user has physically removed a device while an application had the device open (there might be outstanding `I/O`). Users need to reinsert the same physical device and close the application properly before removing the device again. The port cannot configure other inserted devices until this is done.

If the original device cannot be reinserted into the port, see the *Oracle Solaris Administration: Common Tasks* for instructions for clearing this attachment point condition.

`unknown`
Not used.

A USB device can be hotplugged or hotunplugged at any time, and the system detects the event and takes the appropriate action.

It is not necessary to transition a receptacle to the `disconnected` state before removing its device from the USB. However, it is not recommended to hot-remove devices currently in use (such as removable disks currently opened by a volume manager or some other application).

Options `cfgadm` defines several types of operations. These operations include invoking configuration state changes (`-c`), invoking hardware-specific functions (`-x`), and obtaining configuration administration help messages (`-h`).

If any of these operations fail, the device and attachment point might not be in the expected state. Use the `cfgadm -l` command to display the device's current status.

All other options have the same meaning as defined in `cfgadm`(1M).

The following options are supported:

`-c` *function*
The following generic commands are defined for the USB hardware specific library. The following configuration state change operations are supported:

`configure`
If there is a USB device plugged into the port, this command attempts to configure it and set everything up so that it is usable by Solaris. This command does an implied `connect` (reverse of `disconnect`) if necessary. This command accomplishes nothing, and returns an error message, if the device at that port is already configured. After successful execution of this command, the device is ready for use under Solaris.

`disconnect`
Performs an `unconfigure` on the *ap_id* (if it is not already `unconfigured`), and then transitions the receptacle to the `disconnected` state, even though a device is still be plugged into the port. Issuing a `cfgadm -c configure`, or physically hotplugging the device, brings the device back to the `connected` receptacle state, and to the `configured` occupant state, assuming a driver can be found and there are no problems enumerating and configuring the device.

`unconfigure`
Makes the device plugged into the port unusable by Solaris (offline it). If successful, `cfgadm` reports this *ap_id*'s occupant state as *unconfigured*. Issuing a `configure` to the *ap_id* (if successful) brings its occupant back to the `configured` (online) condition, as it physically hotplugging the device on the port.

`-f`
Not supported.

`-h` *ap_id*
USB specific help can be obtained by using the help option with any USB attachment point.

`-l[v]`
The `-l` option works as described in `cfgadm(1M)`. When paired with the `-v` option, the `Information` field contains the following USB-specific information:

- `Mfg`: manufacturer string (`iManufacturer`)
- `Product`: product string (`iProduct`)
- `NConfigs`: total number of configurations the device supports (`bNumConfigurations`).
- `Config`: current configuration setting in decimal (configuration index, not configuration value).
- The configuration string descriptor for the current configuration (`iConfiguration`)

See the Universal Serial Bus specification for a description of these fields.

`-o` *hardware_options*
Hardware options are only supported for the hardware-specific command, `-x usb_config`. See the description of that command below for an explanation of the options available.

`-s` *listing_options*
Attachment points of class USB can be listed by using the `select` sub-option. See `cfgadm(1M)`.

`-x` *hardware_function*
The following hardware-specific functions are defined:

`usb_config -o config=`*n*
This command requires the mandatory `config` value to be specified using the `-o` option.

Sets the USB configuration of a multi-configuration USB device at *ap_id* to configuration index *n*. The device is set to this configuration henceforth and this setting persists across reboots, hot-removes, and unconfigure/configure of the device.

Valid values of *n* range from `0` to (`Nconfigs -1`). The device is reset by a `disconnect` followed by a `configure`. The `configure` causes the device to be configured to the new configuration setting.

If any of these steps fail, the configuration file and the device are restored to their previous state and an error message is issued.

`usb_reset`
Performs a software reset (re-enumeration) of the device. This is the equivalent of removing the device and inserting it back again. The port on the hub is power cycled if the hub supports power cycling of individual ports.

If the connected device is a hub, this function has the effect of resetting that hub and any devices down the tree of which it is the root.

If any of these steps fail, the device is restored to its previous state and an error message is issued.

State table: attachment points state versus commands:

```
Valid states:
    empty/unconfigured         → no device connected

    disconnected/unconfigured  → logically disconnected,
                                 unavailable,
                                 devinfo node removed,
                                 device physically connected

    connected/unconfigured     → logically connected,
                                 unavailable,
                                 devinfo node present

    connected/configured       → connected, available
```

The table below clarifies the state transitions resulting from actions or commands:

```
current state      operation           new state
-------------      ---------           ---------
empty/
unconfigured:
              device plugged in:     connected/configured or
                                     connected/unconfigured
                                     (if enumeration failed)
              device removed:        n/a
              cfgadm -c unconfigure: empty/unconfigured
              cfgadm -c configure:   empty/unconfigured
              cfgadm -c disconnect:  empty/unconfigured
                                     (no-op and error)

disconnected/
unconfigured:
              device plugged in:     n/a
              device removed:        empty/unconfigured
              cfgadm -c unconfigure: disconnected/unconfigured
              cfgadm -c configure:   connected/configured, or
                                     connected/unconfigured
```

```
                                    (if reenumeration failed)
              cfgadm -c disconnect:   disconnected/unconfigured

connected/unconfigured:
              device plugged in:      n/a
              device removed:         empty/unconfigured
              cfgadm -c unconfigure:  connected/unconfigured
              cfgadm -c configure:    connected/configured, or
                                      connected/unconfigured
                                      (if reenumeration failed)
              cfgadm -c disconnect:   disconnected/unconfigured

connected/configured:
              device plugged in:      n/a
              device removed:         empty/unconfigured or
                                      connected/configured,
                                      but with ap condition
                                      'unusable' if device
                                      was open when removed
              cfgadm -c unconfigure:  connected/unconfigured
              cfgadm -c configure:    connected/configured
              cfgadm -c disconnect:   disconnected/unconfigured
```

Examples

EXAMPLE 1 Listing the Status of All USB Devices

The following command lists the status of all USB devices on the system:

```
# cfgadm
Ap_Id           Type         Receptacle   Occupant     Condition
usb0/1          USB-hub      connected    configured   ok
usb0/2          unknown      empty        unconfigured ok
usb0/1.1        USB-storage  connected    configured   ok
usb0/1.2        unknown      empty        unconfigured ok
usb0/1.3        unknown      empty        unconfigured ok
usb0/1.4        USB-device connected    configured   ok
```

Notice that `cfgadm` treats the `USB-device` device at `ap_id` `usb0/1.4` as a single unit, since it cannot currently control individual interfaces.

EXAMPLE 2 Listing the Status of a Port with No Device Plugged In

The following command lists the status of a port with no device plugged in:

```
example# cfgadm -l usb0/1.3
Ap_Id           Type         Receptacle   Occupant     Condition
usb0/1.3        unknown      empty        unconfigured ok
```

EXAMPLE 3 Listing the Status of the Same Port with a Device Plugged In

The following command lists the status of the same port after physically plugging in a device that configures without problems:

EXAMPLE 3 Listing the Status of the Same Port with a Device Plugged In *(Continued)*

```
example# cfgadm -l usb0/1.3
Ap_Id           Type         Receptacle   Occupant     Condition
usb0/1.3        USB-hub      connected    configured   ok
```

EXAMPLE 4 Unconfiguring an Existing USB Device

The following command unconfigures the USB device attached to `usb0/1.3`, then displays the status of the `ap_id`:

```
example# cfgadm -c unconfigure usb0/1.3
Unconfigure the device: /devices/pci@0,0/pci8086,7112@7,2/hub@2:2.3
This operation suspends activity on the USB device
Continue (yes/no)?

Enter:

y

example# cfgadm -l usb0/1.3
Ap_Id           Type         Receptacle   Occupant     Condition
usb0/1.3        unknown      connected    unconfigured ok
```

EXAMPLE 5 Unconfiguring and Logically Disconnecting an Existing USB Device

The following command unconfigures and logically disconnects a USB device attached to `usb0/1.3`:

```
example# cfgadm -c disconnect usb0/1.3
Disconnect the device: /devices/pci@0,0/pci8086,7112@7,2/hub@2:2.3
This operation suspends activity on the USB device
Continue (yes/no)?

Enter:

y

example# cfgadm -l usb0/1.3
Ap_Id         Type         Receptacle     Occupant        Condition
usb0/1.3      unknown      disconnected   unconfigured    ok
```

A `disconnect` implies that `cfgadm` does an `unconfigure` first. The receptacle status now shows `disconnected`, even though the device is still physically connected. In this case, a physical hotplug or using the `cfgadm -c configure` on the `ap_id` brings it back on-line.

EXAMPLE 6 Configuring a Previously Unconfigured USB Device

The following command configures a USB device that was previously attached to `usb0/1.3`:

EXAMPLE 6 Configuring a Previously Unconfigured USB Device *(Continued)*

```
example # cfgadm -yc configure usb0/1.3
example# cfgadm -l usb0/1.3
Ap_Id           Type         Receptacle   Occupant     Condition
usb0/1.3        unknown      connected    configured   ok
```

EXAMPLE 7 Resetting a USB Device

The following command resets a USB device:

```
example# cfgadm -x usb_reset usb0/1.3
Reset the device: /devices/pci@0,0/pci8086,7112@7,2/hub@2:2.3
This operation suspends activity on the USB device
Continue (yes/no)?

Enter:

y
```

EXAMPLE 8 Displaying Detailed Information About a USB Device

The following command displays detailed information about a USB device. This device shows the following USB-specific information in the '`Information`' field:

- Manufacturer string: Iomega
- Product string: `USB Zip 250`
- Number of configurations supported: 1
- Configuration currently active: 0
- Configuration string descriptor for configuration 0: Default

```
example# cfgadm -lv  usb0/1.5
Ap_Id                 Receptacle   Occupant     Condition  Information
When         Type         Busy         Phys_Id
usb0/1.5     connected    configured   ok         Mfg:"Io
mega"  Product:"USB Zip 250"  NConfigs:1  Config:0 : Default

example# cfgadm -l -s "cols=ap_id:info" usb0/1.5
Ap_Id                         Information
usb0/1.5                      Mfg:"Iomega"  Product:"USB Zip 250"
NConfigs:1  Config:0 : Default
```

EXAMPLE 9 Displaying Detailed Information About All USB Devices

The following command displays detailed information about all USB devices on the system:

```
example# cfgadm -l -s "select=class(usb),cols=ap_id:info"
Ap_Id                         Information
usb0/1                        Mfg:<undefined>  Product:<undefined>
NConfigs:1  Config:0 <no cfg str descr>
usb0/2
```

EXAMPLE 9 Displaying Detailed Information About All USB Devices *(Continued)*

```
usb0/1.1                      Mfg:<undefined>  Product:<undefined>
NConfigs:1  Config:0 <no cfg str descr>
usb0/1.2
usb0/1.3
usb0/1.4                      Mfg:"Wizard"  Product:"Modem/ISDN"
NConfigs:3  Config:1 : V.90 Analog Modem
usb0/1.5                      Mfg:"Iomega"  Product:"USB Zip 250"
NConfigs:1  Config:0 : Default
usb0/1.6                      Mfg:"SOLID YEAR"  Product:"SOLID YEAR
USB"NConfigs:1  Config:0 <no cfg str descr>
usb0/1.7
```

Lines containing only an `ap_id` are empty ports. These can be filtered out. This example only lists USB `ap_id`s with connected devices, and information about those devices.

```
example# cfgadm -l -s "select=class(usb),cols=ap_id:info" | grep Mfg
usb0/1                        Mfg:<undefined>  Product:<undefined>
NConfigs:1  Config:0 <no cfg str descr>
usb0/1.1                      Mfg:<undefined>  Product:<undefined>
NConfigs:1  Config:0 <no cfg str descr>
usb0/1.4                      Mfg:"Wizard"  Product:"Modem/ISDN"
NConfigs:3  Config:1 : V.90 Analog Modem
usb0/1.5                      Mfg:"Iomega"  Product:"USB Zip 250"
NConfigs:1  Config:0 : Default
usb0/1.6                      Mfg:"SOLID YEAR"  Product:"SOLID YEAR USB"
Config:0 <no cfg str descr>
```

EXAMPLE 10 Listing Information About a Multi-configuration USB Device

The following example lists information about a multi-configuration USB device.

Notice the `NConfigs` field: the configurations available for this device are `0`, `1`, and `2` (`0` to (*N*`Configs-1`)).

```
example# cfgadm -l -s "cols=ap_id:info" usb0/1.4
Ap_Id                         Information
usb0/1.4                      Mfg:"Wizard"  Product:"Modem/ISDN"
NConfigs:3  Config:1 V.90 Analog Modem"
```

EXAMPLE 11 Setting the Current Configuration of a Multi-configuration USB Device

The following example sets the current configuration of a multi-configuration USB device:

```
example# cfgadm -o config=2 -x usb_config usb0/1.4
Setting the device: /devices/pci@1f,2000/usb@1/device@3
to USB configuration 2
This operation suspends activity on the USB device
Continue (yes/no)?
```

EXAMPLE 11 Setting the Current Configuration of a Multi-configuration USB Device *(Continued)*

```
Enter:

y

USB configuration changed successfully.
```

The device path should be checked to ensure that the right instance of a device is being referred to, in the case where multiple devices of the exact same type are on the same bus. This information is available in the 'Information' field.

Files `/usr/lib/cfgadm/usb.so.1`
Hardware specific library for generic USB device administration

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/library

See Also `cfgadm(1M)`, `config_admin(3CFGADM)`, `attributes(5)`, `scsa2usb(7D)`, `usba(7D)`

Universal Serial Bus 1.1 Specification (`www.usb.org`)

Oracle Solaris Administration: Common Tasks

Notes `cfgadm(1M)` can not unconfigure, disconnect, reset, or change the configuration of any USB device currently opened by any application. These operations also fail on a hub if a device in its hierarchy is opened by an application. See `scsa2usb(7D)` for unconfiguring a USB mass-storage device that is currently in use.

Only super-users can execute any functions on an attachment point. However, one need not be a super-user to list the attachment points.

Name chat – automated conversational exchange tool

Synopsis `chat [options] script`

Description The `chat` program implements a conversational text-based exchange between the computer and any serial device, including (but not limited to) a modem, an ISDN TA, and the remote peer itself, establishing a connection between the Point-To-Point Protocol daemon (`pppd`) and the remote `pppd` process.

Options The `chat` command supports the following options:

`-f` <*chat file*> Read the `chat` script from the `chat` file. This option is mutually exclusive with the `chat` script parameters. You must have `read` access to use the file. Multiple lines are permitted in the file. Use the space or horizontal tab characters to separate the strings.

`-t` <*timeout*> Set the timeout for the expected string to be received. If the string is not received within the time limit, the reply string is not sent. If specified, a 'subexpect' (alternate reply) string can be sent. Otherwise, if no alternate reply strings remain, the `chat` script fails.. A failed script will cause the `chat` program to terminate with a non-zero error code.

`-r` <*report file*> Set the file for output of the report strings. If you use the keyword `REPORT`, the resulting strings are written to this file. If the `-r` option is not used and you use the `REPORT` keyword, the `stderr` file is used for the report strings.

`-e` Start with the echo option turned on. You turn echo on or off at specific points in the `chat` script using the `ECHO` keyword. When echoing is enabled, all output from the modem is echoed to `stderr`.

`-E` Enables environment variable substitution within `chat` scripts using the standard *$xxx* syntax.

`-v` Request that the `chat` script execute in a verbose mode. The `chat` program logs the execution state of the `chat` script as well as all text received from the modem and output strings sent to the modem. The default is to log through `syslog(3C)` with facility `local2`; the logging method is alterable using the `-S` and `-s` options.

`-V` Request that the `chat` script be executed in a `stderr` verbose mode. The `chat` program logs all text received from the modem and output strings sent to the modem to `stderr`. `stderr` is usually the local console at the station running the `chat` or `pppd` program.

`-s` Use `stderr`. Log messages from `-v` and error messages are sent to `stderr`.

`-S`	Do not use syslog. By default, error messages are set to syslog. This option prevents log messages from `-v` and error messages from being sent to syslog.
`-T` <*phone number*>	Pass in an arbitrary string (usually a telephone number) that will be substituted for the `\T` substitution metacharacter in a send string.
`-U` <*phone number 2*>	Pass in a second string (usually a telephone number) that will be substituted for the `\U` substitution metacharacter in a send string. This is useful when dialing an ISDN terminal adapter that requires two numbers.
`script`	If the script is not specified in a file with the `-f` option, the script is included as parameters to the `chat` program.

Extended Description

Chat Script

The `chat` script defines communications. A script consists of one or more "expect-send" pairs of strings separated by spaces, with an optional "subexpect-subsend" string pair, separated by a dash (as in the following example:)

ogin:-BREAK-ogin: ppp ssword: hello2u2

The example indicates that the `chat` program should expect the string "ogin:". If it fails to receive a login prompt within the time interval allotted, it sends a break sequence to the remote and then expects the string "ogin:". If the first "ogin:" is received, the break sequence is not generated.

Upon receiving the login prompt, the `chat` program sends the string ”ppp” and then expects the prompt "ssword:". When the password prompt is received, it sends the password hello2u2.

A carriage return is normally sent following the reply string. It is not expected in the "expect" string unless it is specifically requested by using the `\r` character sequence.

The expect sequence should contain only what is needed to identify the received data. Because it's stored on a disk file, it should not contain variable information. Generally it is not acceptable to look for time strings, network identification strings, or other variable pieces of data as an expect string.

To correct for characters that are corrupted during the initial sequence, look for the string "ogin:" rather than "login:". The leading "l" character may be received in error, creating problems in finding the string. For this reason, scripts look for "ogin:" rather than "login:" and "ssword:" rather than "password:".

An example of a simple script follows:

```
ogin: ppp ssword: hello2u2
```

The example can be intrepreted as: expect ogin:, send ppp, expect ...ssword:, send hello2u2.

When login to a remote peer is necessary, simple scripts are rare. At minimum, you should include sub-expect sequences in case the original string is not received. For example, consider the following script:

```
ogin:--ogin: ppp ssword: hello2u2
```

This script is more effective than the simple one used earlier. The string looks for the same login prompt; however, if one is not received, a single return sequence is sent and then the script looks for login: again. If line noise obscures the first login prompt, send the empty line to generate a login prompt again.

Comments

Comments can be embedded in the `chat` script. Comment lines are ignored by the chat program. A comment starts with the hash ("#") character in column one. If a # character is expected as the first character of the expect sequence, quote the expect string. If you want to wait for a prompt that starts with a # character, write something like this:

```
# Now wait for the prompt and send logout string
'# ' logout
```

Sending Data From A File

If the string to send begins with an at sign ("@"), the remainder of the string is interpreted as the name of the file that contains the string. If the last character of the data read is a newline, it is removed. The file can be a named pipe (or fifo) instead of a regular file. This enables `chat` to communicate with another program, for example, a program to prompt the user and receive a password typed in.

Abort

Many modems report the status of a call as a string. These status strings are often "CONNECTED" or "NO CARRIER" or "BUSY." If the modem fails to connect to the remote, you can terminate the script. Abort strings may be specified in the script using the ABORT sequence. For example:

```
ABORT BUSY ABORT 'NO CARRIER' '' ATZ OK ATDT5551212 CONNECT
```

This sequence expects nothing and sends the string ATZ. The expected response is the string OK. When OK is received, the string ATDT5551212 dials the telephone. The expected string is CONNECT. If CONNECT is received, the remainder of the script is executed. When the modem finds a busy telephone, it sends the string BUSY, causing the string to match the abort character sequence. The script fails because it found a match to the abort string. If the NO CARRIER string is received, it aborts for the same reason.

Clr_Abort

The CLR_ABORT sequence clears previously set ABORT strings. ABORT strings are kept in an array of a pre-determined size; CLR_ABORT reclaims the space for cleared entries, enabling new strings to use that space.

Say The SAY string enables the script to send strings to a user at a terminal via standard error. If `chat` is being run by `pppd` and `pppd` is running as a daemon (detached from its controlling terminal), standard error is normally redirected to the `/etc/ppp/connect-errors` file.

SAY strings must be enclosed in single or double quotes. If carriage return and line feed are required for the output, you must explicitly add them to your string.

The SAY string can provide progress messages to users even with "ECHO OFF." For example, add a line similar to the following to the script:

```
ABORT BUSY
ECHO OFF
SAY "Dialing your ISP...\n"
'' ATDT5551212
TIMEOUT 120
SAY "Waiting up to 2 minutes for connection ..."
CONNECT ''
SAY "Connected, now logging in ...\n"
ogin: account
ssword: pass
$ \c
SAY "Logged in OK ... \n"
```

This sequence hides script detail while presenting the SAY string to the user. In this case, you will see:

```
Dialing your ISP...
Waiting up to 2 minutes for connection...Connected, now logging in...
Logged in OK ...
```

Report REPORT is similar to the ABORT string. With REPORT, however, strings and all characters to the next control character (such as a carriage return), are written to the report file.

REPORT strings can be used to isolate a modem's transmission rate from its CONNECT string and return the value to the `chat` user. Analysis of the REPORT string logic occurs in conjunction with other string processing, such as looking for the expect string. It's possible to use the same string for a REPORT and ABORT sequence, but probably not useful.

Report strings may be specified in the script using the REPORT sequence. For example:

```
REPORT CONNECT
ABORT BUSY
ATDT5551212 CONNECT
ogin: account
```

The above sequence expects nothing, then sends the string ATDT5551212 to dial the telephone. The expected string is CONNECT. If CONNECT is received, the remainder of the

script is executed. In addition, the program writes the string CONNECT to the report file (specified by `-r`) in addition to any characters that follow.

Clr_Report CLR_REPORT clears previously set REPORT strings. REPORT strings are kept in an array of a pre-determined size; CLR_REPORT reclaims the space for cleared entries so that new strings can use that space.

Echo ECHO determines if modem output is echoed to `stderr`. This option may be set with the `-e` option, but can also be controlled by the ECHO keyword. The "expect-send" pair ECHO ON enables echoing, and ECHO OFF disables it. With ECHO, you can select which parts of the conversation should be visible. In the following script:

```
ABORT   'BUSY'
ABORT   'NO CARRIER'
""    AT&F
OK\r\n  ATD1234567
\r\n    \c
ECHO    ON
CONNECT \c
ogin:   account
```

All output resulting from modem configuration and dialing is not visible, but output is echoed beginning with the CONNECT (or BUSY) message.

Hangup The HANGUP option determines if a modem hangup is considered as an error. HANGUP is useful for dialing systems that hang up and call your system back. HANGUP can be ON or OFF. When HANGUP is set to OFF and the modem hangs up (for example, following the first stage of logging in to a callback system), `chat` continues running the script (for example, waiting for the incoming call and second stage login prompt). When the incoming call is connected, use the HANGUP ON string to reinstall normal hang up signal behavior. An example of a simple script follows:

```
ABORT   'BUSY'
 ""  AT&F
 OK\r\n  ATD1234567
 \r\n    \c
 CONNECT \c
 'Callback login:' call_back_ID
 HANGUP OFF
 ABORT "Bad Login"
 'Callback Password:' Call_back_password
 TIMEOUT 120
 CONNECT \c
 HANGUP ON
 ABORT "NO CARRIER"
 ogin:--BREAK--ogin: real_account
```

Timeout The initial timeout value is 45 seconds. Use the `-t` parameter to change the intial timeout value.

To change the timeout value for the next expect string, the following example can be used:

```
''"AT&F
 OK ATDT5551212
 CONNECT \c
 TIMEOUT 10
 ogin:--ogin: username
 TIMEOUT 5
 assword: hello2u2
```

The example changes the timeout to ten seconds when it expects the login: prompt. The timeout is changed to five seconds when it looks for the password prompt.

Once changed, the timeout value remains in effect until it is changed again.

EOT The EOT special reply string instructs the `chat` program to send an EOT character to the remote. This is equivalent to using ^D\c as the reply string. The EOT string normally indicates the end-of-file character sequence. A return character is not sent following the EOT. The EOT sequence can embedded into the send string using the sequence ^D.

BREAK The BREAK special reply string sends a break condition. The break is a special transmitter signal. Many UNIX systems handle break by cycling through available bit rates, and sending break is often needed when the remote system does not support autobaud. BREAK is equivalent to using \K\c as the reply string. You embed the break sequence into the send string using the \K sequence.

Escape Sequences Expect and reply strings can contain escape sequences. Reply strings accept all escape sequences, while expect strings accept most sequences. A list of escape sequences is presented below. Sequences that are not accepted by expect strings are indicated.

`''` Expects or sends a null string. If you send a null string, `chat` sends the return character. If you expect a null string, `chat` proceeds to the reply string without waiting. This sequence can be a pair of apostrophes or quote mark characters.

`\b` Represents a backspace character.

`\c` Suppresses the newline at the end of the reply string. This is the only method to send a string without a trailing return character. This sequence must be at the end of the send string. For example, the sequence hello\c will simply send the characters h, e, l, l, o. (Not valid in expect.)

`\d` Delay for one second. The program uses `sleep(1)` which delays to a maximum of one second. (Not valid in expect.)

`\K` Insert a BREAK. (Not valid in expect.)

`\n`	Send a newline or linefeed character.
`\N`	Send a null character. The same sequence may be represented by \0. (Not valid in expect.)
`\p`	Pause for 1/10th of a second. (Not valid in expect.)
`\q`	Suppress writing the string to syslog. The string `??????` is written to the log in its place. (Not valid in expect.)
`\r`	Send or expect a carriage return.
`\s`	Represents a space character in the string. Can be used when it is not desirable to quote the strings which contains spaces. The sequence 'HI TIM' and HI\sTIM are the same.
`\t`	Send or expect a tab character.
`\T`	Send the phone number string as specified with the `-T` option. (Not valid in expect.)
`\U`	Send the phone number 2 string as specified with the `-U` option. (Not valid in expect.)
`\\`	Send or expect a backslash character.
`\ddd`	Collapse the octal digits (ddd) into a single ASCII character and send that character. (\000 is not valid in an expect string.)
`^C`	Substitute the sequence with the control character represented by C. For example, the character DC1 (17) is shown as ^Q. (Some characters are not valid in expect.)

Environment Variables Environment variables are available within `chat` scripts if the `-E` option is specified on the command line. The metacharacter `$` introduces the name of the environment variable to substitute. If the substition fails because the requested environment variable is not set, nothing is replaced for the variable.

Exit Status The `chat` program terminates with the following completion codes:

`0`	Normal program termination. Indicates that the script was executed without error to normal conclusion.
`1`	One or more of the parameters are invalid or an expect string was too large for the internal buffers. Indicates that the program was not properly executed.
`2`	An error occurred during the execution of the program. This may be due to a read or write operation failing or `chat` receiving a signal such as SIGINT.
`3`	A timeout event occurred when there was an expect string without having a "-subsend" string. This indicates that you may not have programmed the script correctly for the condition or that an unexpected event occurred and the expected string could not be found.

4 The first string marked as an ABORT condition occurred.

5 The second string marked as an ABORT condition occurred.

6 The third string marked as an ABORT condition occurred.

7 The fourth string marked as an ABORT condition occurred.

... The other termination codes are also strings marked as an ABORT condition.

To determine which event terminated the script, use the termination code. It is possible to decide if the string "BUSY" was received from the modem versus "NO DIALTONE." While the first event may be retried, the second probably will not succeed during a retry.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/network/ppp
Interface Stability	Committed

See Also `sleep(1)`, `uucp(1C)`, `pppd(1M)`, `uucico(1M)`, `syslog(3C)`, `attributes(5)`

Additional information on `chat` scripts are available with UUCP documentation. The `chat` script format was taken from scripts used by the `uucico` program.

Name check-hostname – check if sendmail can determine the system's fully-qualified host name

Synopsis

```
/usr/sbin/check-hostname
```

Description The `check-hostname` script is a migration aid for `sendmail(1M)`. This script tries to determine the local host's fully-qualified host name (FQHN) in a manner similar to `sendmail(1M)`. If `check-hostname` is able to determine the FQHN of the local host, it reports success. Otherwise, `check-hostname` reports how to reconfigure the system so that the FQHN can be properly determined.

Files

`/etc/hosts`	Host name database
`/etc/nsswitch.conf`	Name service switch configuration file
`/etc/resolv.conf`	Configuration file for name server routines

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	service/network/smtp/sendmail
Interface Stability	Committed

See Also `domainname(1M)`, `sendmail(1M)`, `hosts(4)`, `attributes(5)`

Name check-permissions – check permissions on mail rerouting files

Synopsis

```
/usr/sbin/check-permissions [login]
```

Description The `check-permissions` script is intended as a migration aid for `sendmail(1M)`. It checks the `/etc/mail/sendmail.cf` file for all configured alias files, and checks the alias files for `:include:` files. It also checks for certain `.forward` files. For each file that `check-permissions` checks, it verifies that none of the parent directories are group- or world-writable. If any directories are overly permissive, it is reported. Otherwise it reports that no unsafe directories were found.

As to which `.forward` files are checked, it depends on the arguments included on the command line. If no argument is given, the current user's home directory is checked for the presence of a `.forward` file. If any arguments are given, they are assumed to be valid logins, and the home directory of each one is checked.

If the special argument `ALL` is given, the `passwd` entry in the `/etc/nsswitch.conf` file is checked, and all password entries that can be obtained through the switch file are checked. In large domains, this can be time-consuming.

Operands The following operands are supported:

login Where *login* is a valid user name, checks the home directory for *login*.

`ALL` Checks the home directory of *all* users.

Files

`/etc/mail/sendmail.cf` Defines environment for `sendmail`

`/etc/mail/aliases` Ascii mail aliases file

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	service/network/smtp/sendmail
Interface Stability	Committed

See Also `getent(1M)`, `sendmail(1M)`, `aliases(4)`, `attributes(5)`

Name chk_encodings – check the label encodings file syntax

Synopsis

```
/usr/sbin/chk_encodings [-a] [-c maxclass] [pathname]
```

Description `chk_encodings` checks the syntax of the label-encodings file that is specified by *pathname*. With the `-a` option, `chk_encodings` also prints a semantic analysis of the label-encodings file that is specified by *pathname*. If *pathname* is not specified, `chk_encodings` checks and analyzes the `/etc/security/tsol/label_encodings` file.

If label-encodings file analysis was requested, whatever analysis can be provided is written to the standard output file even if errors were found.

Options

`-a` Provide a semantic analysis of the label encodings file.

`-c` *maxclass* Accept a maximum classification value of *maxclass* (default 255) in the label encodings file `CLASSIFICATIONS` section.

Exit Status When successful, `chk_encodings` returns an exit status of `0` (true) and writes to the standard output file a confirmation that no errors were found in *pathname*. Otherwise, `chk_encodings` returns an exit status of nonzero (false) and writes an error diagnostic to the standard output file.

Files

`/etc/security/tsol/label_encodings`
The label encodings file contains the classification names, words, constraints, and values for the defined labels of this system.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/trusted
Interface Stability	See below.
Standard	DDS-2600-6216-93, *Compartmented Mode Workstation Labeling: Encodings Format*, September 1993

The command output is Not-an-Interface. The command invocation is Committed for systems that implement the DIA MAC policy.

See Also `label_encodings(4)`, `attributes(5)`, `labels(5)`

"How to Analyze and Verify the label_encodings File" in *Trusted Extensions Label Administration*

Notes The functionality described on this manual page is available only if the system is configured with Trusted Extensions.

This file is part of the Defense Intelligence Agency (DIA) Mandatory Access Control (MAC) policy. This file might not be applicable to other MAC policies that might be developed for future releases of Solaris Trusted Extensions software.

Name chroot – change root directory for a command

Synopsis `/usr/sbin/chroot` *newroot command*

Description The `chroot` utility causes *command* to be executed relative to *newroot*. The meaning of any initial slashes (`/`) in the path names is changed to *newroot* for *command* and any of its child processes. Upon execution, the initial working directory is *newroot*.

Notice that redirecting the output of *command* to a file,

`chroot` *newroot command* `>x`

will create the file `x` relative to the original root of *command*, not the new one.

The new root path name is always relative to the current root. Even if a `chroot` is currently in effect, the *newroot* argument is relative to the current root of the running process.

This command can be run only by the super-user.

Return Values The exit status of `chroot` is the return value of *command*.

Examples **EXAMPLE 1** Using the `chroot` Utility

The `chroot` utility provides an easy way to extract `tar` files (see `tar(1)`) written with absolute filenames to a different location. It is necessary to copy the shared libraries used by `tar` (see `ldd(1)`) to the *newroot* filesystem.

```
example# mkdir /tmp/lib; cd /lib
example# cp ld.so.1 libc.so.1 libcmd.so.1 libdl.so.1 \
         libsec.so.1 /tmp/lib
example# cp /usr/bin/tar /tmp
example# dd if=/dev/rmt/0 | chroot /tmp tar xvf -
```

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also `cd(1)`, `tar(1)`, `chroot(2)`, `ttyname(3C)`, `attributes(5)`

Notes Exercise extreme caution when referencing device files in the new root file system.

References by routines such as `ttyname(3C)` to stdin, stdout, and stderr will find that the device associated with the file descriptor is unknown after `chroot` is run.

Name cimworkshop – start the Sun WBEM CIM WorkShop application

Synopsis

```
/usr/sadm/bin/cimworkshop
```

Description The `cimworkshop` command starts Sun WBEM CIM WorkShop, a graphical user interface that enables you to create, modify, and view the classes and instances that describe the managed resources on your system.

Managed resources are described using a standard information model called Common Information Model (CIM). A CIM class is a computer representation, or model, of a type of managed resource, such as a printer, disk drive, or CPU. A CIM instance is a particular managed resource that belongs to a particular class. Instances contain actual data. Objects can be shared by any WBEM-enabled system, device, or application. CIM objects are grouped into meaningful collections called schema. One or more schemas can be stored in directory-like structures called namespaces.

The CIM WorkShop application displays a Login dialog box. Context help is displayed on the left side of the CIM WorkShop dialog boxes. When you click on a field, the help content changes to describe the selected field.

By default, CIM WorkShop uses the RMI protocol to connect to the CIM Object Manager on the local host, in the default namespace, `root\cimv2`. You can select HTTP if you want to communicate to a CIM Object Manager using the standard XML/HTTP protocol from the Desktop Management Task Force. When a connection is established, all classes contained in the default namespace are displayed in the left side of the CIM WorkShop window.

The name of the current namespace is listed in the tool bar. All programming operations are performed within a namespace. Four namespaces are created in a `root` namespace during installation:

cimv2	Contains the default CIM classes that represent managed resources on your system.
security	Contains the security classes used by the CIM Object Manager to represent access rights for users and namespaces.
system	Contains properties for configuring the CIM Object Manager.
snmp	Contains pre-defined SNMP-related classes and all SNMP MOF files that are compiled.

The `cimworkshop` application allows you to perform the following tasks:

Create, view, and change namespaces.	Use the CIM WorkShop application to view all namespaces. A namespace is a directory-like structure that can store CIM classes and instances.

Create, delete, and view CIM classes.	You cannot modify the unique attributes of the classes that make up the CIM and Solaris Schema. You can create a new instance or subclass of the class and modify the desired attributes in that instance or subclass.
Create, modify, delete, and view CIM instances.	You can add instances to a class and modify its inherited properties or create new properties. You can also change the property values of a CIM instance.
Invoke methods.	You can set input values for a parameter of a method and invoke the method.

When CIM WorkShop connects to the CIM Object Manager in a particular namespace, all subsequent operations occur within that namespace. When you connect to a namespace, you can access the classes and instances in that namespace (if they exist) and in any namespaces contained in that namespace.

When you use CIM WorkShop to view CIM data, the WBEM system validates your login information on the current host. By default, a validated WBEM user is granted read access to the CIM Schema. The CIM Schema describes managed objects on your system in a standard format that all WBEM-enabled systems and applications can interpret.

Read Only	Allows read-only access to CIM Schema objects. Users with this privilege can retrieve instances and classes, but cannot create, delete, or modify CIM objects.
Read/Write	Allows full read, write, and delete access to all CIM classes and instances.
Write	Allows write and delete, but not read access to all CIM classes and instances.
None	Allows no access to CIM classes and instances.

Usage The `cimworkshop` command is not a tool for a distributed environment. Rather, this command is used for local administration on the machine on which the CIM Object Manager is running.

Exit Status The `cimworkshop` utility terminates with exit status `0`.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWwbdev

See Also mofcomp(1M), wbemlogviewer(1M), init.wbem(1M), attributes(5)

Name clear_locks – clear locks held on behalf of an NFS client

Synopsis

```
/usr/sbin/clear_locks [-s] hostname
```

Description The `clear_locks` command removes all file, record, and share locks created by the *hostname* and held on the current host, regardless of which process created or owns the locks.

This command can be run only by the super-user.

This command should only be used to repair the rare case of a client crashing and failing to clear held locks. Clearing locks held by an active client may cause applications to fail in an unexpected manner.

Options

`-s` Remove all locks created by the current machine and held by the server *hostname*.

Operands The following operands are supported:

hostname name of host server

Exit Status

`0` Successful operation.

`1` If not root.

`2` Usage error.

`3` If unable to contact server (RPC).

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also `fcntl(2)`, `attributes(5)`

Name clinfo – display cluster information

Synopsis

```
clinfo [-nh]
```

Description The `clinfo` command displays cluster configuration information about the node from which the command is executed.

Without arguments, `clinfo` returns an exit status of `0` if the node is configured and booted as part of a cluster. Otherwise, `clinfo` returns an exit status of `1`.

Options The following options are supported:

`-h` Displays the highest node number allowed to be configured. This is different from the maximum number of nodes supported in a given cluster. The current highest configured node number can change immediately after the command returns since new nodes can be dynamically added to a running cluster.

For example, `clinfo -h` might return `64`, meaning that the highest number you can use to identify a node is `64`. See the *Sun Cluster 3.0 System Administration Guide* for a description of utilities you can use to determine the number of nodes in a cluster.

`-n` Prints the number of the node from which `clinfo` is executed.

Exit Status The following exit values are returned:

`0` Successful completion.

`1` An error occurred.

This is usually because the node is not configured or booted as part of a cluster.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also `attributes(5)`

Name clri, dcopy – clear inode

Synopsis

```
clri [-F FSType] [-V] special i-number

dcopy [-F FSType] [-V] special i-number
```

Description `clri` writes zeros on the inodes with the decimal *i-number* on the file system stored on *special*. After `clri`, any blocks in the affected file show up as missing in an `fsck(1M)` of *special*.

Read and write permission is required on the specified file system device. The inode becomes allocatable.

The primary purpose of this routine is to remove a file that for some reason appears in no directory. If it is used to zap an inode that does appear in a directory, care should be taken to track down the entry and remove it. Otherwise, when the inode is reallocated to some new file, the old entry will still point to that file. At that point, removing the old entry will destroy the new file. The new entry will again point to an unallocated inode, so the whole cycle is likely to be repeated again and again.

`dcopy` is a symbolic link to `clri`.

Options

`-F` *FSType* Specify the *FSType* on which to operate. The *FSType* should either be specified here or be determinable from `/etc/vfstab` by matching *special* with an entry in the table, or by consulting `/etc/default/fs`.

`-V` Echo the complete command line, but do not execute the command. The command line is generated by using the options and arguments provided by the user and adding to them information derived from `/etc/vfstab`. This option should be used to verify and validate the command line.

Usage See `largefile(5)` for the description of the behavior of `clri` and `dcopy` when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

Files

`/etc/default/fs` Default local file system type

`/etc/vfstab` List of default parameters for each file system

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also `fsck(1M)`, `vfstab(4)`, `attributes(5)`, `largefile(5)`

Notes This command might not be supported for all *FSTypes*.

Name configCCR – configure Oracle Configuration Manager

Synopsis

```
/usr/lib/ocm/ccr/bin/configCCR

configCCR [ -a ] [ -c ] [ -d | [ -C OracleSUpportHubURL] [-s]
     [CSI-number [[MyOracleSupportID] [country]]

configCCR [ -r ] [ -C OracleSUpportHubURL] [-s]

configCCR [-R response_file]

configCCR -D [ -v ] [ -T target_type [ -N target_name[
     -P property_name ]]]
```

Description The `configCCR` command is used to modify Oracle Configuration Manager (OCM) configuration information. This command enables you to modify the registration credentials after Oracle Configuration Manager has been installed. You can also use this command to switch between Connected and Disconnected modes, and to configure Diagnostic Checks properties.

Options The following options are supported:

`-a`
Run with instance data the directory defined by `ORACLE_CONFIG_HOME`.

`-c`
Indicates that the installation will be designated as the central collector.

`-C` *URL*
Defines the Oracle Support Hub used to connect to Oracle.

`-d`
Run in disconnected mode. In this mode data is not collected automatically.

`-D` [`-v`] [`-T` *target_type* [`-N` *target_name* [`-P` *property_name*]]]
Configure missing diagnostic check properties.

`-v`
Verifies that all target properties are properly configured and that no properties are missing.

`-T` *target_type*
Provides the target type to be configured or verified.

`-N` *target_name*
Provides the target name to be configured or verified. Must be used with `-T`.

`-P` *property_name*
Provides the property name for the target to be configured or verified. Must be used with `-N` and `-T`.

`-R` *file*
Reconfigures using the contents of the response file named *file*.

`-r`
Removes the instance data in the directory defined by `ORACLE_CONFIG_HOME`. If not set the `.../ccr/hosts/`*hostname* (where *hostname* is the current hostname) is removed with the associated `crontab(1)` entry.

`-s`
Indicates that you accept the Oracle Configuration Manager License agreement.

Operands

CSI-number
Customer Support Identifier

MyOracleSupportID
My Oracle Support User Name

country
two-letter country code

Files

`/usr/lib/ocm`
Directory to store all OCM related files and data.

`/usr/lib/ocm/ccr/config`
Contains configuration files.

`/usr/lib/ocm/ccr/log`
Contains log files.

`/usr/lib/ocm/ccr/state`
Contains state files and collected data.

`/usr/lib/ocm/ccr/state/review/targetMap.xml`
Summaries of the data that was collected.

`/usr/lib/ocm/ccr/state/review`
Data that was uploaded to the server.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWocm

See Also

`crontab(1)`, `emCCR(1M)`, `emocmrsp(1M)`, `attributes(5)`

Oracle Configuration Manager Installation and Administration Guide

Name consadm – select or display devices used as auxiliary console devices

Synopsis

```
/usr/sbin/consadm

/usr/sbin/consadm [-a device. . .] [-p]

/usr/sbin/consadm [-d device. . .] [-p]

/usr/sbin/consadm [-p]
```

Description `consadm` selects the hardware *device* or devices to be used as auxiliary console devices, or displays the current device. Only superusers are allowed to make or display auxiliary console device selections.

Auxiliary console devices receive copies of `console` messages, and can be used as the console during `single user` mode. In particular, they receive kernel messages and messages directed to `/dev/sysmsg`. On Solaris x86 based systems they can also be used for interaction with the bootstrap.

By default, selecting a display device to be used as an auxiliary console device selects that device for the duration the system remains up. If the administrator needs the selection to persist across reboots the `-p` option can be specified.

`consadm` runs a daemon in the background, monitoring auxiliary console devices. Any devices that are disconnected (hang up, lose carrier) are removed from the auxiliary console device list, though not from the persistent list. While auxiliary console devices may have been removed from the device list receiving copies of `console` messages, those messages will always continue to be displayed by the default console device.

The daemon will not run if it finds there are not any auxiliary devices configured to monitor. Likewise, after the last auxiliary console is removed, the daemon will shut itself down. Therefore the daemon persists for only as long as auxiliary console devices remain active.

See `eeprom(1M)` for instructions on assigning an auxiliary console device as the system console.

Options The following options are supported:

`-a` *device* — Adds *device* to the list of auxiliary console devices. Specify *device* as the path name to the device or devices to be added to the auxiliary console device list.

`-d` *device* — Removes *device* from the list of auxiliary console devices. Specify *device* as the path name to the device or devices to be removed from the auxiliary console device list.

`-p` — Prints the list of auxiliary consoles that will be auxiliary across reboots.

When invoked with the `-a` or `-d` options , tells the application to make the change persist across reboot.

Examples

EXAMPLE 1 Adding to the list of devices that will receive console messages

The following command adds `/dev/term/a` to the list of devices that will receive console messages.

```
example# consadm -a /dev/term/a
```

EXAMPLE 2 Removing from the list of devices that will receive console messages

The following command removes `/dev/term/a` from the list of devices that will receive console messages. This includes removal from the persistent list.

```
example# consadm -d -p /dev/term/a
```

EXAMPLE 3 Printing the list of devices selected as auxiliary console devices

The following command prints the name or names of the device or devices currently selected as auxiliary console devices.

```
example# consadm
```

Environment Variables

See `environ(5)` for descriptions of the following environment variables that affect the execution of `consadm`: `LC_CTYPE`, `LC_MESSAGES`, and `NLSPATH`.

Exit Status

The following exit values are returned:

`0` Successful completion.

>`0` An error occurred.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os
Interface Stability	Committed

See Also

`kmdb(1)`, `svcs(1)`, `eeprom(1M)`, `svcadm(1M)`, `syslogd(1M)`, `environ(5)`, `attributes(5)`, `smf(5)`, `sysmsg(7D)`, `console(7D)`

Notes

Auxiliary console devices are not usable for `kmdb` or firmware I/O, do not receive panic messages, and do not receive output directed to `/dev/console`.

The `consadm` service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/system/consadm
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. The service's status can be queried using the `svcs(1)` command.

Name console-reset – force console login to display

Synopsis

```
svc:/system/console-reset
```

Description The `console-reset` service runs if X is not running on console and the console is ready for interactive use. Under these conditions, `console-reset` turns off the boot-time graphics, forcing the console login prompt to display.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Uncommitted

See Also `svcs(1)`, `svcadm(1M)`, `vbiosd(1M)`, `attributes(5)`, `smf(5)`

Notes The `console-reset` service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/system/console-reset:default
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. The service's status can be queried using the `svcs(1)` command.

Name coreadm – core file administration

Synopsis

```
coreadm [-g pattern] [-G content] [-i pattern] [-I content]
     [-d option]... [-e option]...

coreadm [-p pattern] [-P content] [pid]...
```

Description `coreadm` specifies the name and location of core files produced by abnormally-terminating processes. See `core(4)`.

Only users and roles that belong to the "Maintenance and Repair" RBAC profile can execute the first form of the `SYNOPSIS`. This form configures system-wide core file options, including a global core file name pattern and a core file name pattern for the `init(1M)` process. All settings are saved persistently and will be applied at boot.

Non-privileged users can execute the second form of the `SYNOPSIS`. This form specifies the file name pattern and core file content that the operating system uses to generate a per-process core file.

A core file name pattern is a normal file system path name with embedded variables, specified with a leading `%` character. The variables are expanded from values that are effective when a core file is generated by the operating system. The possible embedded variables are as follows:

`%d`
Executable file directory name, up to a maximum of `MAXPATHLEN` characters

`%f`
Executable file name, up to a maximum of `MAXCOMLEN` characters

`%g`
Effective group-ID

`%m`
Machine name (`uname -m`)

`%n`
System node name (`uname -n`)

`%p`
Process-ID

`%t`
Decimal value of `time(2)`

`%u`
Effective user-ID

`%z`
Name of the zone in which process executed (`zonename`)

`%%`
Literal `%`

For example, the core file name pattern `/var/cores/core.%f.%p` would result, for command `foo` with process-ID `1234`, in the core file name `/var/cores/core.foo.1234`.

A core file content description is specified using a series of tokens to identify parts of a process's binary image:

`anon`
Anonymous private mappings, including thread stacks that are not main thread stacks

`ctf`
CTF type information sections for loaded object files

`data`
Writable private file mappings

`dism`
DISM mappings

`heap`
Process heap

`ism`
ISM mappings

`rodata`
Read-only private file mappings

`shanon`
Anonymous shared mappings

`shfile`
Shared mappings that are backed by files

`shm`
System V shared memory

`stack`
Process stack

`symtab`
Symbol table sections for loaded object files

`text`
Readable and executable private file mappings

In addition, you can use the token `all` to indicate that core files should include all of these parts of the process's binary image. You can use the token `none` to indicate that no mappings are to be included. The `default` token indicates inclusion of the system default content (`stack+heap+shm+ism+dism+text+data+rodata+anon+shanon+ctf+symtab`). The `/proc` file system data structures are always present in core files regardless of the mapping content.

You can use + and - to concatenate tokens. For example, the core file content `default-ism` would produce a core file with the default set of mappings without any intimate shared memory mappings.

The `coreadm` command with no arguments reports the current system configuration, for example:

```
$ coreadm
    global core file pattern: /var/cores/core.%f.%p
    global core file content: all
      init core file pattern: core
      init core file content: default
           global core dumps: enabled
      per-process core dumps: enabled
     global setid core dumps: enabled
per-process setid core dumps: disabled
    global core dump logging: disabled
```

The `coreadm` command with only a list of process-IDs reports each process's per-process core file name pattern, for example:

```
$ coreadm 278 5678
  278:   core.%f.%p default
  5678:  /home/george/cores/%f.%p.%t all-ism
```

Only the owner of a process or a user with the `proc_owner` privilege can interrogate a process in this manner.

When a process is dumping core, up to three core files can be produced: one in the per-process location, one in the system-wide global location, and, if the process was running in a local (non-global) zone, one in the global location for the zone in which that process was running. Each core file is generated according to the effective options for the corresponding location.

When generated, a global core file is created in mode `600` and owned by the superuser. Nonprivileged users cannot examine such files.

Ordinary per-process core files are created in mode `600` under the credentials of the process. The owner of the process can examine such files.

A process that is or ever has been `setuid` or `setgid` since its last exec(2) presents security issues that relate to dumping core. Similarly, a process that initially had superuser privileges and lost those privileges through setuid(2) also presents security issues that are related to dumping core. A process of either type can contain sensitive information in its address space to which the current nonprivileged owner of the process should not have access. If `setid` core files are enabled, they are created mode `600` and owned by the superuser.

Options The following options are supported:

`-d` *option*...
Disable the specified core file option. See the `-e` *option* for descriptions of possible options.

Multiple `-e` and `-d` options can be specified on the command line. Only users and roles belonging to the "Maintenance and Repair" RBAC profile can use this option.

`-e` *option*...
Enable the specified core file option. Specify *option* as one of the following:

global
Allow core dumps that use global core pattern.

global-setid
Allow set-id core dumps that use global core pattern.

log
Generate a `syslog(3C)` message when generation of a global core file is attempted.

process
Allow core dumps that use per-process core pattern.

proc-setid
Allow set-id core dumps that use per-process core pattern.

Multiple `-e` and `-d` options can be specified on the command line. Only users and roles belonging to the "Maintenance and Repair" RBAC profile can use this option.

`-g` *pattern*
Set the global core file name pattern to *pattern*. The pattern must start with a `/` and can contain any of the special `%` variables that are described in the `DESCRIPTION`.

Only users and roles belonging to the "Maintenance and Repair" RBAC profile can use this option.

`-G` *content*
Set the global core file content to content. You must specify content by using the tokens that are described in the `DESCRIPTION`.

Only users and roles belonging to the "Maintenance and Repair" RBAC profile can use this option.

`-i` *pattern*
Set the default per-process core file name to *pattern*. This changes the per-process pattern for any process whose per-process pattern is still set to the default. Processes that have had their per-process pattern set or are descended from a process that had its per-process pattern set (using the `-p` option) are unaffected. This default persists across reboot.

Only users and roles belonging to the "Maintenance and Repair" RBAC profile can use this option.

`-I` *content*
Set the default per-process core file content to *content*. This changes the per-process content for any process whose per-process content is still set to the default. Processes that have had their per-process content set or are descended from a process that had its per-process content set (using the `-P` option) are unaffected. This default persists across reboot.

Only users and roles belonging to the "Maintenance and Repair" RBAC profile can use this option.

`-p` *pattern*
Set the per-process core file name pattern to *pattern* for each of the specified process-IDs. The pattern can contain any of the special `%` variables described in the `DESCRIPTION` and need not begin with `/`. If the pattern does not begin with `/`, it is evaluated relative to the directory that is current when the process generates a core file.

A nonprivileged user can apply the `-p` option only to processes that are owned by that user. A user with the `proc_owner` privilege can apply the option to any process. The per-process core file name pattern is inherited by future child processes of the affected processes. See `fork(2)`.

If no process-IDs are specified, the `-p` option sets the per-process core file name pattern to *pattern* on the parent process (usually the shell that ran `coreadm`).

`-P` *content*
Set the per-process core file content to *content* for each of the specified process-IDs. The content must be specified by using the tokens that are described in the `DESCRIPTION`.

A nonprivileged user can apply the `-p` option only to processes that are owned by that user. A user with the `proc_owner` privilege can apply the option to any process. The per-process core file name pattern is inherited by future child processes of the affected processes. See `fork(2)`.

If no process-IDs are specified, the `-P` option sets the per-process file content to *content* on the parent process (usually the shell that ran `coreadm`).

Operands The following operands are supported:

pid
process-ID

Examples **EXAMPLE 1** Setting the Core File Name Pattern

When executed from a user's `$HOME/.profile` or `$HOME/.login`, the following command sets the core file name pattern for all processes that are run during the login session:

```
example$  coreadm -p core.%f.%p
```

Note that since the process-ID is omitted, the per-process core file name pattern will be set in the shell that is currently running and is inherited by all child processes.

EXAMPLE 2 Dumping a User's Files Into a Subdirectory

The following command dumps all of a user's core dumps into the `corefiles` subdirectory of the home directory, discriminated by the system node name. This command is useful for users who use many different machines but have a shared home directory.

```
example$  coreadm -p $HOME/corefiles/%n.%f.%p 1234
```

EXAMPLE 3 Culling the Global Core File Repository

The following commands set up the system to produce core files in the global repository only if the executables were run from `/usr/bin` or `/usr/sbin`.

```
example# mkdir -p /var/cores/usr/bin
example# mkdir -p /var/cores/usr/sbin
example# coreadm -G all -g /var/cores/%d/%f.%p.%n
```

Files

`/var/cores`
: Directory provided for global core file storage.

Exit Status

The following exit values are returned:

0
: Successful completion.

1
: A fatal error occurred while either obtaining or modifying the system core file configuration.

2
: Invalid command-line options were specified.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also

`gcore(1)`, `pfexec(1)`, `svcs(1)`, `init(1M)`, `svcadm(1M)`, `exec(2)`, `fork(2)`, `setuid(2)`, `time(2)`, `syslog(3C)`, `core(4)`, `prof_attr(4)`, `user_attr(4)`, `attributes(5)`, `smf(5)`

Notes

In a local (non-global) zone, the global settings apply to processes running in that zone. In addition, the global zone's apply to processes run in any zone.

The term *global settings* refers to settings which are applied to the system or zone as a whole, and does not necessarily imply that the settings are to take effect in the global zone.

The `coreadm` service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/system/coreadm:default
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using svcadm(1M). The service's status can be queried using the svcs(1) command.

The -g, -G, -i, -I, -e, and -d options can be also used by a user, role, or profile that has been granted both the solaris.smf.manage.coreadm and solaris.smf.value.coreadm authorizations.

Name cpustat – monitor system behavior using CPU performance counters

Synopsis

```
cpustat -c eventspec [-c eventspec]... [-p period] [-T u | d ]
     [-Dmnst] [-A cor|soc|bins] [-k keys] [-o limit]
     [-I statfile] [-O statfile] [interval [count]]

cpustat -h
```

Description The `cpustat` utility allows CPU performance counters to be used to monitor the overall behavior of the CPUs in the system.

If *interval* is specified, `cpustat` samples activity every *interval* seconds, repeating forever. If a *count* is specified, the statistics are repeated *count* times. If neither are specified, an interval of five seconds is used, and there is no limit to the number of samples that are taken.

Options The following options are supported:

`-A cor`
: Aggregate output by core ID. Data rows having the same core ID are aggregated into one row. The columns are replaced with subtotals, by default. The `-m` option prints column averages, instead.

`-A soc`
: Aggregate output by socket ID. Data rows having the same socket ID are aggregated into one row. The columns are replaced with subtotals, by default. The `-m` option prints column averages, instead.

`-A bins`
: Aggregate the rows into a lesser number of bins within each sampling period, grouping them in the order in which they appear, and print the columnar subtotal over rows for each bin. The `-m` option may be used in order to compute the arithmetic mean instead of the subtotal. The `-k` sorting option may be used to change the row order prior to the binning step. The `sze` column prints the number of CPUs in each bin. The `BIN` column replaces the `CPU` column and prints the ordinal of each `bin`.

`-c` *eventspec*
: Specifies a set of events for the CPU performance counters to monitor. The syntax of these event specifications is:

```
[picn=]eventn[,attr[n][=val]][,[picn=]eventn
     [,attr[n][=val]],...,]
```

You can use the `-h` option to obtain a list of available events and attributes. This causes generation of the usage message. You can omit an explicit counter assignment, in which case `cpustat` attempts to choose a capable counter automatically.

Attribute values can be expressed in hexadecimal, octal, or decimal notation, in a format suitable for `strtoll(3C)`. An attribute present in the event specification without an explicit value receives a default value of `1`. An attribute without a corresponding counter number is applied to all counters in the specification.

The semantics of these event specifications can be determined by reading the CPU manufacturer's documentation for the events.

Multiple `-c` options can be specified, in which case the command cycles between the different event settings on each sample.

`-D`
Enables debug mode.

`-h`
Prints an extensive help message on how to use the utility and how to program the processor-dependent counters.

`-I` *statfile*
Replay data previously saved in *statfile*. Create data files for replay by specifying `-O`. This option is especially useful for analyzing statistics on machines with large numbers of CPUs. The file may be reprocessed multiple times using different sorting and aggregation options.

The `-I` option is incompatible with an interval and count specification.

Read from the standard input if the file name is – (hyphen).

`-k` *key1*,...
Sort rows within each sampling period from highest to lowest by *key1*, then *key2*, and so on. Each key is a comma-separated list of events. There may be multiple `-k` options specified.

When `cpustat` is run with multiple `-c` *event-spec* options it produces a report of alternating *event-spec*s. Specify multiple `-k` options to sort each *event-spec* differently. For each *event-spec*, the first `-k` option whose keys contain a proper subset of the events in the *event-spec* is used.

`-m`
Print the arithmetic mean value rather than the sum when the `-b` or `-i` is used to aggregate data over multiple CPUs.

`-n`
Omits all header output (useful if `cpustat` is the beginning of a pipeline).

`-o` *num*
Print only the first *num* rows within each sampling period, after applying sorting and aggregation options.

`-O` *statfile*
Save all data to *statfile*. This data may be replayed at a later time using `-I`.

Write to the standard output if the file name is – (hyphen).

The purpose of `-O` is to capture all available data. It is incompatible with the data reduction options: `-A`, `-k`, `-m` and `-o`.

`-p` *period*
Causes `cpustat` to cycle through the list of *eventspec*s every *period* seconds. The tool sleeps after each cycle until *period* seconds have elapsed since the first *eventspec* was measured.

When this option is present, the optional *count* parameter specifies the number of total cycles to make (instead of the number of total samples to take). If *period* is less than the number of *eventspec*s times *interval*, the tool acts as it period is `0`.

`-s`
Creates an idle soaker thread to spin while system-only *eventspec*s are bound. One idle soaker thread is bound to each CPU in the current processor set. System-only *eventspec*s contain both the `nouser` and the `sys` tokens and measure events that occur while the CPU is operating in privileged mode. This option prevents the kernel's idle loop from running and triggering system-mode events.

`-T u | d`
Display a time stamp.

Specify `u` for a printed representation of the internal representation of time. See `time(2)`. Specify `d` for standard date format. See `date(1)`.

`-t`
Prints an additional column of processor cycle counts, if available on the current architecture.

Usage A closely related utility, `cputrack(1)`, can be used to monitor the behavior of individual applications with little or no interference from other activities on the system.

The `cpustat` utility must be run by the super-user, as there is an intrinsic conflict between the use of the CPU performance counters system-wide by `cpustat` and the use of the CPU performance counters to monitor an individual process (for example, by `cputrack`.)

Once any instance of this utility has started, no further per-process or per-LWP use of the counters is allowed until the last instance of the utility terminates.

The times printed by the command correspond to the wallclock time when the hardware counters were actually sampled, instead of when the program told the kernel to sample them. The time is derived from the same timebase as `gethrtime(3C)`.

The processor cycle counts enabled by the `-t` option always apply to both user and system modes, regardless of the settings applied to the performance counter registers.

On some hardware platforms running in system mode using the “sys” token, the counters are implemented using 32-bit registers. While the kernel attempts to catch all overflows to synthesize 64-bit counters, because of hardware implementation restrictions, overflows can be lost unless the sampling interval is kept short enough. The events most prone to wrap are those that count processor clock cycles. If such an event is of interest, sampling should occur frequently so that less than 4 billion clock cycles can occur between samples.

The output of cpustat is designed to be readily parseable by nawk(1) and perl(1), thereby allowing performance tools to be composed by embedding cpustat in scripts. Alternatively, tools can be constructed directly using the same APIs that cpustat is built upon using the facilities of libcpc(3LIB). See cpc(3CPC).

The cpustat utility only monitors the CPUs that are accessible to it in the current processor set. Thus, several instances of the utility can be running on the CPUs in different processor sets. See psrset(1M) for more information about processor sets.

Because cpustat uses LWPs bound to CPUs, the utility might have to be terminated before the configuration of the relevant processor can be changed.

Examples

SPARC **EXAMPLE 1** Measuring External Cache References and Misses

The following example measures misses and references in the external cache. These occur while the processor is operating in user mode on an UltraSPARC machine.

```
example% cpustat -c EC_ref,EC_misses 1 3

    time cpu event      pic0      pic1
   1.008   0  tick     69284      1647
   1.008   1  tick     43284      1175
   2.008   0  tick    179576      1834
   2.008   1  tick    202022     12046
   3.008   0  tick     93262       384
   3.008   1  tick     63649      1118
   3.008   2 total    651077     18204
```

x86 **EXAMPLE 2** Measuring Branch Prediction Success on Pentium 4

The following example measures branch mispredictions and total branch instructions in user and system mode on a Pentium 4 machine.

```
example% cpustat -c \
   pic12=branch_retired,emask12=0x4,pic14=branch_retired,\
   emask14=0xf,sys 1 3

    time cpu event      pic12     pic14
   1.010   1  tick       458       684
   1.010   0  tick       305       511
   2.010   0  tick       181       269
   2.010   1  tick       469       684
   3.010   0  tick       182       269
   3.010   1  tick       468       684
   3.010   2 total      2063      3101
```

EXAMPLE 3 Counting Memory Accesses on Opteron

The following example determines the number of memory accesses made through each memory controller on an Opteron, broken down by internal memory latency:

```
cpustat -c \
   pic0=NB_mem_ctrlr_page_access,umask0=0x01, \
   pic1=NB_mem_ctrlr_page_access,umask1=0x02, \
   pic2=NB_mem_ctrlr_page_access,umask2=0x04,sys \
   1

    time cpu event      pic0      pic1      pic2
   1.003   0  tick     41976     53519      7720
   1.003   1  tick      5589     19402       731
   2.003   1  tick      6011     17005       658
   2.003   0  tick     43944     45473      7338
   3.003   1  tick      7105     20177       762
   3.003   0  tick     47045     48025      7119
   4.003   0  tick     43224     46296      6694
   4.003   1  tick      5366     19114       652
```

EXAMPLE 4 Displaying Multiple CPUs with a Filter

The following command displays the three CPUs with the highest `DTLB_miss` rate.

```
example% cpustat -c DTLB_miss -k DTLB_miss -n 3 1 1

 time cpu event DTLB_miss
1.040 115  tick       107
1.006  18  tick        98
1.045 126  tick        31
1.046  96 total       236

event DTLB_miss
total       236
```

EXAMPLE 5 Aggregating Multiple CPUs into Quartiles by a Filter

The following command aggregates 256 CPUs into quartiles by DTLB miss rate.

```
example% cpustat -c DTLB_miss -b 4 -k DTLB_miss -m 1 1

 time bin event DTLB_miss sze
1.032   0  tick        46  24
1.021   1  tick         3  24
1.007   2  tick         2  24
1.022   3  tick         0  24
1.045   4 total        51  24

event DTLB_miss
total        51
```

EXAMPLE 6 Sorting Multiple Events

The following sequence of commands sorts multiple events.

```
example% cpustat -O /tmp/OUT -c ITLB_miss,DTLB_miss -c PAPI_tot_ins 1 2
example% cpustat  -I /tmp/OUT -b 4 -k ITLB_miss -k PAPI_tot_ins

 time bin event ITLB_miss DTLB_miss sze
1.020   0  tick       129       673  24
1.009   1  tick         0        61  24
1.005   2  tick         0        79  24
1.039   3  tick         0        64  24
1.082   4 total       129       877  24

 time bin event PAPI_tot_ins sze
2.073   0  tick        51947  24
2.020   1  tick        14976  24
2.076   2  tick        14976  24
2.004   3  tick        14976  24
2.082   4 total        96875  24

event ITLB_miss DTLB_miss PAPI_tot_ins
total       129       877        96875
```

Warnings By running the `cpustat` command, the super-user forcibly invalidates all existing performance counter context. This can in turn cause all invocations of the `cputrack` command, and other users of performance counter context, to exit prematurely with unspecified errors.

If `cpustat` is invoked on a system that has CPU performance counters which are not supported by Solaris, the following message appears:

```
cpustat: cannot access performance counters - Operation not applicable
```

This error message implies that `cpc_open()` has failed and is documented in `cpc_open(3CPC)`. Review this documentation for more information about the problem and possible solutions.

If a short interval is requested, `cpustat` might not be able to keep up with the desired sample rate. In this case, some samples might be dropped.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	diagnostic/cpu-counters
Interface Stability	Committed

See Also cputrack(1), nawk(1), perl(1), iostat(1M), prstat(1M), psrset(1M), vmstat(1M), cpc(3CPC), cpc_open(3CPC), cpc_bind_cpu(3CPC), gethrtime(3C), strtoll(3C), libcpc(3LIB), attributes(5)

Notes When cpustat is run on a Pentium 4 with HyperThreading enabled, a CPC set is bound to only one logical CPU of each physical CPU. See cpc_bind_cpu(3CPC).

Name croinfo, diskinfo – query and display information about 1) chassis, receptacle, and occupants or 2) disk occupants of bay receptacles

Synopsis

```
croinfo [-h] [-v] [-I cro_db] [-o fields] [-O fields]
        [-P product-id] [-C chassis-id] [-A alias-id]
        [-R receptacle-name] [-T receptacle-type] [-t occupant-type]
        [-D devchassis-path] [-d occupant-devices] [-p occupant-paths]
        [-c occupant-compdev] [-i occupant-devid] [-m occupant-mfg]
        [-e occupant-model] [-n occupant-part] [-s occupant-serial]
        [-f occupant-firm] [-1 occupant-misc-1] [-2 occupant-misc-2]
        [-3 occupant-misc-3]

diskinfo [-h] [-v] [-I cro_db] [-o fields] [-O fields]
        [-P product-id] [-C chassis-id] [-A alias-id]
        [-R receptacle-name] [-T receptacle-type] [-t occupant-type]
        [-D devchassis-path] [-d occupant-devices] [-p occupant-paths]
        [-c occupant-compdev] [-i occupant-devid] [-m occupant-mfg]
        [-e occupant-model] [-n occupant-part] [-s occupant-serial]
        [-f occupant-firm] [-1 occupant-misc-1] [-2 occupant-misc-2]
        [-3 occupant-misc-3]

croinfo -?

diskinfo -?
```

Description The `diskinfo` and `croinfo` utility share the same binary executable. At runtime, the utility checks to see how it was invoked, and adjusts defaults.

The `croinfo` utility allows users to query and display specific aspects of a system's configuration. Queries are performed against a record-oriented dataset that captures the relationship between physical location and various aspects of the device currently at that physical location. This relationship is expressed in terms of Chassis, Receptacle, and Occupant (thus the `cro` prefix).

Records in a CRO dataset are composed of multiple, named fields with each record having a potentially unique field value. An angle-bracket reference, such as *product-id*, is referring to a specific *field-name*. For a given record, a field value is either undefined (empty) or defined with a set of indexed string values. Some defined field values have just one string value, while others can have multiple string values.

Each *field-name* defined is associated with a separate *field-char* character. By convention, uppercase *field-char* characters are associated for chassis and receptacle information fields, and lowercase *field-char* characters are associated with occupant information fields. For each *field-char* character, a separate `-field-char field-name-RE` flag regular expression filter option is provided. This allows the user to customize queries to display information about specific aspects of the configuration. Records that match all regular expressions are selected, in dataset order, for display. For multiple string values, only one index value needs to match for the field to match.

By default, only a minimal number of default output fields are displayed. You can override the default with either `-o` *fields* for human-readable output or with `-O` *fields* for parseable output. In both cases, output fields can be specified using either the short-hand *field-char*[...] notation or in the more descriptive *field-name*[,...] notation.

For human-readable use, by using the `-o` *fields* option, the user can override the default fields, and output any fields, in the specified order, in a column-aligned whitespace separated format. In general, output will be one line of output per matching record with undefined (empty) *field-name* values displayed as a hyphen (–). If, however, a displayed record has a multiple string *value* field, then multiple lines of output are produced with any secondary non-multiple string values fields showing a colon (`:`).

For scripting, by using the `-O` *fields* option, the user can override the default output fields and output any fields, in the specified order, in a parseable colon-separated format with whitespace removed and column headers suppressed. Output will be one line of output per matching record with undefined (empty) field-name values displayed with no value. If a displayed record *field-name* has a multiple string *value* field, then all the values are concatenated, separated by a semicolon. Any occurrence of a colon or a semicolon in a value is escaped with a leading backslash (`\`). To make scripts more legible, use of the `-O` *field-name*[,...]' notation is encouraged.

The `-o` and `-O` options are mutually exclusive.

If the `-h` option is used, or scripting output format is requested by using `-o`, the column headers for output fields are suppressed.

A Chassis is identified by a specific *product-id* and *chassis-id*. The *product-id* relates to a specific chassis-level product, like a system chassis Sun-Fire-X4200-M2 or a storage chassis SUN-Storage-J4410. For a given *product-id* value, the *chassis-id* defines a unique serial number.

A specific *product-id*.*chassis-id* combination can have an "managed" location-oriented *alias-id* defined by the administrator, using `fmadm(1M)` that provides installation-specific location information about where a chassis is physically located. This might include such information as building, room, rack, and U-number range within a rack.

In addition to the managed location-oriented *alias-id* defined using `fmadm(1M)`, system chassis always have one well-known alias called `SYS` that can be used to identify receptacles that are internal to the system chassis.

Within a chassis, each receptacle has a unique *receptacle-name* that should match the physical silk-screen label designation for the receptacle. Each receptacle also has a *receptacle-type*, which helps define acceptable *occupant-types*.

When a receptacle is occupied, use the `-f` flag definitions for available *occupant-information*. Of particular interest is the `-c` *occupant-compdev* occupant information: it describes the common component of the public `/dev` name associated with the occupant device. For disks, this is the whole-disk `c#t#d#` name.

The CRO dataset order is associated with the *devchassis-path* of the record, which corresponds to the `/dev/chassis` name space maintained by `devchassisd(1M)`. That ordering places records associated with the well-known `SYS` internal alias first, and records with `BOOT` receptacle-name first in `SYS`. This is done to ensure that, when applicable, information about the typical boot device is provided first.

For `croinfo`, the default output is in `-o` *Dtc* format, and all CRO records are shown.

For `diskinfo`, the default output is in `-o` *Dc* format, and a `-T` *bay receptacle-type* filter is applied. The meanings of the *occupant-misc-#* fields also take on a disk-specific interpretation: *misc-1* is capacity, and *misc-2* is target-port information. These defaults allow the `diskinfo` command to query the relationship between chassis, *bay* receptacles, and their *disk* occupants, while ignoring other CRO information.

Options For each record *field-name* defined, a separate *-field-char field-name-RE* flag regular expression filter can be specified. For a given *field-name*, if no specific *-field-char field-name-RE* filter is defined, all CRO records match.

This allows the user to customize queries to display information about specific aspects of the configuration. CRO records that match all of the specified *field-name* regular expressions (as in `regex(3C)`) will be selected for display, with specific fields output controlled by means of `-o`, `-O`, or the default.

`-P` *product-id*
The *product-id* specifies the product identifier of an enumerated chassis. The *product-id* might be exposed in the `/dev/chassis` name space. For storage products that do not have an established `fmadm(1M)` managed *alias-id*, the *product-id* is visible in the `devchassis(7FS)` `/dev/chassis` name space.

Example system *product-id* value: `Sun-Fire-X4200-M2`

Example storage *product-id* value: `SUN-Storage-J4410`

`-C` *chassis-id*
The *chassis-id* specifies the serial number of a product chassis. The *chassis-id* might be exposed in the `/dev/chassis` name space. For storage products that do not have an established `fmadm(1M)` managed *alias-id*, the *product-id* is visible in the `devchassis(7FS)` `/dev/chassis` name space.

Example *chassis-id* value: `0818QAJ002`

`-A` *alias-id*
An *alias-id* value can be the well-known alias value of `SYS`, for system internal devices. In addition, an *alias-id* value can be a managed alias, defined by the administrator using `fmadm(1M)`. The intended use of a managed alias is to define the physical location of the *product-id.chassis-id*. The *alias-id* is exposed in the `/dev/chassis` name space.

Example well-known *alias-id* value: `SYS`

Example managed *alias-id* value: `RACK29.U01-04`

`-R` *receptacle-name*
For a specific *product-id*, the unique *receptacle-name* defines location of a specific receptacle in a chassis. The *receptacle-name* should be identical to a silk-screen label on the physical chassis, and should also match product documentation. The *receptacle-name* can have multiple path components, such as `SYS/HD0`. The receptacle-name is exposed in the `/dev/chassis` name space.

Example *receptacle-name* value: `SYS/HD0`

`-T` *receptacle-type*
Example *receptacle-type* value: `bay`

`-t` *occupant-type*
A receptacle without an occupant has an undefined (empty) *occupant-type* value, shown as a hyphen (–).

Example *occupant-type* value: `disk`

`-D` *devchassis-path*
Example *devchassis-path* value: `/dev/chassis/SYS/HD0/disk`

`-d` *occupant-devices*
A receptacle without an occupant has an undefined (empty) *occupant-devices* value, shown as a hyphen (–).

Example *occupant-devices* value:

```
/devices/scsi_vhci/disk@g5000c500101ba0a3
```

`-p` *occupant-paths*
A receptacle without an occupant has an undefined (empty) *occupant-paths* value, shown as a hyphen (–).

Example occupant-paths value:

```
devices/pci@0,0/pci10de,5d@d/pci11f8,8001@0/iport@f/disk@w5000c500101ba0a1,0
```

`-c` *occupant-compdev*
A receptacle without an occupant has an undefined (empty) *occupant-compdev* value, shown as a hyphen (–).

Example *occupant-compdev* value: `c0t5000C500101BA0A3d0`

`-i` *occupant-devid*
A receptacle without an occupant has an undefined (empty) *occupant-devid* value, shown as a hyphen (–).

Example *occupant-devid* value: `id1,sd@n5000c500101ba0a3`

`-m` *occupant-mfg*
A receptacle without an occupant has an undefined (empty) *occupant-mfg* value, shown as a hyphen (–).

Example *occupant-mfg* value: `SEAGATE`

`-e` *occupant-model*
A receptacle without an occupant has an undefined (empty) *occupant-model*, shown as a hyphen (–).

Example *occupant-model* value: `ST32000SSSUN2.0T`

`-n` *occupant-part*
A receptacle without an occupant has an undefined (empty) *occupant-part* value, shown as a hyphen (–).

Example *occupant-part* value: `SEAGATE-ST32000SSSUN2.0T`

`-s` *occupant-serial*
A receptacle without an occupant has an undefined (empty) *occupant-serial* value, shown as a hyphen (–).

Example *occupant-serial* value: `000949L09C8L_____9WM09C8L`

`-f` *occupant-firm*
A receptacle without an occupant has an undefined (empty) *occupant-firm* value, shown as a hyphen (–).

Example *occupant-firm* value: `0313`

`-1` *occupant-misc-1*
A receptacle without an occupant has an undefined (empty) *occupant-misc-1* value, shown as a hyphen (–).

`-2` *occupant-misc-2*
A receptacle without an occupant has an undefined (empty) *occupant-misc-2* value, shown as a hyphen (–).

`-3` *occupant-misc-3*
A receptacle without an occupant has an undefined (empty) *occupant-misc-3* value, shown as a hyphen (–).

`-?`
Display usage information.

Output *field-name* Control Options

`-o` *fields*
Output specified fields, in order, in human-readable format.

For `croinfo`, default output is in `-o` *Dtc* format. For `diskinfo`, default output is in `-O` *Dc* format.

`-O` *fields*
Output specified fields, in order, in parseable format.

`-h`
Do not output *field-name* column headers

`-v`
Display verbose header that includes various information about when the CRO dataset was created. This option is of particular use with `-I` and is used to specify a non-standard source for the CRO dataset.

Dataset Selection Option

`-I` *cro_db*
Data file from which to obtain CRO dataset information.

Examples

In some of the following examples, example output wraps in an 80–character-wide display.

EXAMPLE 1 Determining Where a Disk is Located

The following command determines where a disk is located:

```
# croinfo -c c0t5000C500101BA0A3d0
D:devchassis-path                       t:occupant-type
--------------------------------------  ---------------
/dev/chassis/RACK29.U01-04/DISK_00/disk  disk

c:occupant-compdev
------------------
c0t5000C500101BA0A3d0
```

EXAMPLE 2 Reporting Internal Disks

The following command reports the *receptacle-name* and the *occupant-compdev* of internal disks, that is, disks that are associated with the well-known `SYS` alias.

```
# diskinfo -A SYS -o Rc
R:receptacle-name  c:occupant-compdev
-----------------  ------------------
SYS/HD0            c8t0d0
SYS/HD1            c8t1d0
SYS/HD2            -
SYS/HD3            -
```

Note that the `SYS/HD2` and `SYS/HD3` receptacles are empty.

The same command, in scripting output mode, would produce:

```
# diskinfo -A SYS -O receptacle-name,occupant-compdev
SYS/HD0:c8t0d0
SYS/HD1:c8t1d0
SYS/HD2:
SYS/HD3:
```

EXAMPLE 3 Reporting Disks in a Specific Enclosure

The following command reports all the disks within a chassis with a specific *product-id* and *chassis-id* value.

```
# diskinfo -P SUN-Storage-J4410 -C SUN-Storage-J4410 -o Rc
R:receptacle-name  c:occupant-compdev
-----------------  --------------------
DISK_00            c0t5000C500101BA0A3d0
DISK_01            c0t5000C500101B95BBd0
DISK_02            -
```

EXAMPLE 4 Reporting Physical Path Information

The following command reports physical path information for a specific disk.

```
# croinfo -c c0t5000C500101BA0A3d0 -o cp
c:occupant-compdev
------------------
c0t5000C500101BA0A3d0

p:occupant-paths
-------------------------------------------------
/devices/pci@0,0/pci10de,5d@d/pci11f8,8001@0/iport@f/disk@w5000c500101ba0a1,0
/devices/pci@7b,0/pci10de,5d@d/pci11f8,8001@0/iport@f/disk@w5000c500101ba0a2,0
```

Note that *occupant-paths* has multiple string values.

EXAMPLE 5 Making Inventory of Disks

The following example reports how many of a specific type of disk are available using *occupant-part*.

```
# for i in ‘croinfo -h -o n  | sort -u‘
> do
> echo $i "\t\c";croinfo -h -n $i | wc -l
> done
SEAGATE-ST330055SSUN300G               3
SEAGATE-ST330056SSUN300G              19
SEAGATE-ST345056SSUN450G               5
```

EXAMPLE 6 Locating a Specific Type of Disk

The following command reports where disks of a specific type are located, what their `ctd` name is (by means of *occupant-compdev*), and what firmware level they are at.

```
# croinfo -n SEAGATE-ST330055SSUN300G -o Dcf
D:Devchassis                                    c:component
----------------------------------------------  --------------------
/dev/chassis/RACK29.U29-32/SCSI_Device__11/disk  c0t5000C50007DD49F7d0
/dev/chassis/RACK29.U33-36/SCSI_Device__18/disk  c0t5000C50008F7FB4Fd0
/dev/chassis/RACK29.U33-36/SCSI_Device__19/disk  c0t5000C50007DD412Fd0
```

EXAMPLE 6 Locating a Specific Type of Disk *(Continued)*

```
f:firm
------
0892
0892
0892
```

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os
Interface Stability	See below

The interface stability of `croinfo` and `diskinfo` is Committed. The interface stability of command output is Not-an-Interface.

See Also

`devchassisd(1M)`, `fmadm(1M)`, `fmd(1M)`, `attributes(5)`, `devchassis(7FS)`

The SCSI Storage Interfaces committee website, `http://www.t10.org`

SCSI Primary Commands-4, SPC4; *SCSI Enclosure Services-2*; *SES2, Serial Attached SCSI-2, SAS2*

Notes

`croinfo` representation depends on the ability of `fmd(1M)` to enumerate system topology and accurately represent associated chassis, receptacles, and occupants. These dependencies might extend through `fmd(1M)` and require that connected hardware, and its associated firmware, comply with specific standards. For disk bays, this requires that storage chassis behave in a T10 standards-compliant (SPC4 and SES2) fashion. Storage chassis that do not respond appropriately might not report chassis, bays, or disk nodes correctly. Specifically, `diskinfo` requires that chassis support SES diagnostic page `0xa` (Additional Element Status) and set the Element Index Present (EIP) bit to `1`. Enclosures that do not meet this criterion will not be fully enumerated, and thus will not be properly represented.

Name cron – clock daemon

Synopsis

```
/usr/sbin/cron
```

Description `cron` starts a process that executes commands at specified dates and times.

You can specify regularly scheduled commands to `cron` according to instructions found in `crontab` files in the directory `/var/spool/cron/crontabs`. Users can submit their own `crontab` file using the `crontab(1)` command. Commands which are to be executed only once can be submitted using the `at(1)` command.

`cron` only examines `crontab` or `at` command files during its own process initialization phase and when the `crontab` or `at` command is run. This reduces the overhead of checking for new or changed files at regularly scheduled intervals.

As `cron` never exits, it should be executed only once. This is done routinely by way of the `svc:/system/cron:default` service. The file `/etc/cron.d/FIFO` file is used as a lock file to prevent the execution of more than one instance of `cron`.

`cron` captures the output of the job's `stdout` and `stderr` streams, and, if it is not empty, mails the output to the user. If the job does not produce output, no mail is sent to the user. An exception is if the job is an `at(1)` job and the `-m` option was specified when the job was submitted.

`cron` and `at` jobs are not executed if your account is locked. Jobs and processses execute. The `shadow(4)` file defines which accounts are not locked and will have their jobs and processes executed.

Setting cron Jobs Across Timezones

The timezone of the `cron` daemon sets the system-wide timezone for `cron` entries. This, in turn, is by set by default system-wide using `/etc/default/init`. The timezone for `cron` entries can be overridden in a user's crontab file; see `crontab(1)`.

If some form of *daylight savings* or *summer/winter time* is in effect, then jobs scheduled during the switchover period could be executed once, twice, or not at all.

Setting cron Defaults

To keep a log of all actions taken by `cron`, you must specify `CRONLOG=YES` in the `/etc/default/cron` file. If you specify `CRONLOG=NO`, no logging is done. Keeping the log is a user configurable option since `cron` usually creates huge log files.

You can specify the `PATH` for `user` cron jobs by using `PATH=` in `/etc/default/cron`. You can set the `PATH` for `root` cron jobs using `SUPATH=` in `/etc/default/cron`. Carefully consider the security implications of setting `PATH` and `SUPATH`.

Example `/etc/default/cron` file:

```
CRONLOG=YES
PATH=/usr/bin:/usr/ucb:
```

This example enables logging and sets the default `PATH` used by non-root jobs to `/usr/bin:/usr/ucb:`. Root jobs continue to use `/usr/sbin:/usr/bin`.

The cron log file is periodically rotated by logadm(1M).

Files

/etc/cron.d	Main cron directory
/etc/cron.d/FIFO	Lock file
/etc/default/cron	cron default settings file
/var/cron/log	cron history information
/var/spool/cron	Spool area
/etc/cron.d/queuedefs	Queue description file for at, batch, and cron
/etc/logadm.conf	Configuration file for logadm

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also svcs(1), at(1), crontab(1), sh(1), logadm(1M), svcadm(1M), queuedefs(4), shadow(4), attributes(5), rbac(5), smf(5), smf_security(5)

Notes The cron service is managed by the service management facility, smf(5), under the service identifier:

```
svc:/system/cron:default
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using svcadm(1M). The service's status can be queried using the svcs(1) command. Most administrative actions may be delegated to users with the solaris.smf.manage.cron authorization (see rbac(5) and smf_security(5)).

Diagnostics A history of all actions taken by cron is stored in /var/cron/log and possibly in /var/cron/olog.

Name cryptoadm – cryptographic framework administration

Synopsis

```
cryptoadm list [-mpv] [provider=provider-name]
     [mechanism=mechanism-list]

cryptoadm disable
     provider=provider-name mechanism=mechanism-list | random | all

cryptoadm enable
     provider=provider-name mechanism=mechanism-list | random | all

cryptoadm install provider=provider-name

cryptoadm install provider=provider-name
     [mechanism=mechanism-list]

cryptoadm uninstall provider=provider-name

cryptoadm unload provider=provider-name

cryptoadm disable fips-140

cryptoadm enable fips-140

cryptoadm list fips-140

cryptoadm refresh

cryptoadm start

cryptoadm stop

cryptoadm --help
```

Description The `cryptoadm` utility displays cryptographic provider information for a system, configures the mechanism policy for each provider, and installs or uninstalls a cryptographic provider. The cryptographic framework supports three types of providers: a user-level provider (a PKCS11 shared library), a kernel software provider (a loadable kernel software module), and a kernel hardware provider (a cryptographic hardware device).

For kernel software providers, the `cryptoadm` utility provides the `unload` subcommand. This subcommand instructs the kernel to unload a kernel software providers.

For the cryptographic framework's metaslot, the `cryptoadm` utility provides subcommands to enable and disable the metaslot's features, list metaslot's configuration, specify alternate persistent object storage, and configure the metaslot's mechanism policy.

The `cryptoadm` utility provides subcommands to enable and disable FIPS-140 mode in the Cryptographic Framework. It also provides a `list` subcommand to display the current status of FIPS-140 mode.

Administrators will find it useful to use `syslog` facilities (see `syslogd(1M)` and `logadm(1M)`) to maintain the cryptographic subsystem. Logging can be especially useful under the following circumstances:

- If kernel-level daemon is dead, all applications fail. You can learn this from syslog and use `svcadm(1M)` to restart the `svc:/system/cryptosvc` service.
- If there are bad providers plugged into the framework, you can learn this from syslog and remove the bad providers from the framework.

With the exception of the subcommands or options listed below, the `cryptoadm` command needs to be run by a privileged user.

- subcommand `list`, any options
- subcommand `--help`

Options The `cryptoadm` utility has the various combinations of subcommands and options shown below.

`cryptoadm list`
Display the list of installed providers.

`cryptoadm list metaslot`
Display the system-wide configuration for metaslot.

`cryptoadm list -m [ provider=`*provider-name* `| metaslot ]`
Display a list of mechanisms that can be used with the installed providers or metaslot. If a provider is specified, display the name of the specified provider and the mechanism list that can be used with that provider. If the metaslot keyword is specified, display the list of mechanisms that can be used with metaslot.

`cryptoadm list -p [ provider=`*provider-name* `| metaslot ]`
Display the mechanism policy (that is, which mechanisms are available and which are not) for the installed providers. Also display the provider feature policy or metaslot. If a provider is specified, display the name of the provider with the mechanism policy enforced on it only. If the metaslot keyword is specified, display the mechanism policy enforced on the metaslot.

`cryptoadm list -v provider=`*provider-name* `| metaslot`
Display details about the specified provider if a provider is specified. If the metaslot keyword is specified, display details about the metaslot.

`-v`
For the various `list` subcommands described above (except for `list -p`), the `-v` (verbose) option provides details about providers, mechanisms and slots.

`cryptoadm disable provider=`*provider-name*
[mechanism=*mechanism-list* | *provider-feature* `... | all`]
Disable the mechanisms or provider features specified for the provider. See OPERANDS for a description of *mechanism*, *provider-feature*, and the `all` keyword.

`cryptoadm [ mechanism=`*mechanism-list* `] [ auto-key-migrate ]`
Disable the metaslot feature in the cryptographic framework or disable some of metaslot's features. If no operand is specified, this command disables the metaslot feature in the

cryptographic framework. If a list of mechanisms is specified, disable mechanisms specified for metaslot. If all mechanisms are disabled for metaslot, the metaslot will be disabled. See OPERANDS for a description of mechanism. If the `auto-key-migrate` keyword is specified, it disables the migration of sensitive token objects to other slots even if it is necessary for performing crypto operations. See OPERANDS for a description of `auto-key-migrate`.

`cryptoadm enable provider=`*provider-name*
[mechanism=*mechanism-list* | *provider-feature* `...` | `all`]

Enable the mechanisms or provider features specified for the provider. See OPERANDS for a description of *mechanism*, *provider-feature*, and the `all` keyword.

`cryptoadm enable metaslot [ mechanism=`*mechanism-list* `] |`
`[ [ token=`*token-label*`] [ slot=`*slot-description*`] |`
`default-keystore ] | [ auto-key-migrate ]`

If no operand is specified, this command enables the metaslot feature in the cryptographic framework. If a list of mechanisms is specified, it enables only the list of specified mechanisms for metaslot. If *token-label* is specified, the specified token will be used as the persistent object store. If the *slot-description* is specified, the specified slot will be used as the persistent object store. If both the *token-label* and the *slot-description* are specified, the provider with the matching token label and slot description is used as the persistent object store. If the `default-keystore` keyword is specified, metaslot will use the default persistent object store. If the `auto-key-migrate` keyword is specified, sensitive token objects will automatically migrate to other slots as needed to complete certain crypto operations. See OPERANDS for a description of mechanism, token, slot, `default-keystore`, and `auto-key-migrate`.

`cryptoadm install provider=`*provider-name*

Install a user-level provider into the system. The *provider* operand must be an absolute pathname of the corresponding shared library. If there are both 32–bit and 64–bit versions for a library, this command should be run once only with the path name containing `$ISA`. Note that `$ISA` is not a reference to an environment variable. Note also that `$ISA` must be quoted (with single quotes [for example, `'$ISA'`]) or the `$` must be escaped to keep it from being incorrectly expanded by the shell. The user-level framework expands `$ISA` to an empty string or an architecture-specific directory, for example, `sparcv9`.

The preferred way of installing a user-level provider is to build a package for the provider. For more information, see the *Solaris Security for Developer's Guide*.

`cryptoadm install provider=`*provider-name*
mechanism=*mechanism-list*

Install a kernel software provider into the system. The provider should contain the base name only. The *mechanism-list* operand specifies the complete list of mechanisms to be supported by this provider, or the keyword `all`.

The preferred way of installing a kernel software provider is to build a package for providers. For more information, see the *Solaris Security for Developer's Guide*.

`cryptoadm uninstall provider=`*provider-name*
Uninstall the specified *provider* and the associated mechanism policy from the system. This subcommand applies only to a user-level provider or a kernel software provider.

`cryptoadm unload provider=`*provider-name*
Unload the kernel software module specified by *provider*.

`cryptoadm disable fips-140`
Disable FIPS-140 mode in the Cryptographic Framework and for hardware providers. A reboot is required following this command to complete the disabling. Alternatively, you could boot a BE that does not have FIPS-140 mode enabled.

`cryptoadm enable fips-140`
Enable FIPS-140 mode in the Cryptographic Framework and for hardware providers. This subcommand does not disable the non-FIPS approved algorithms from the user-level `pkcs11_softtoken` library and the kernel software providers. It is the consumers of the framework that are responsible for using only FIPS-approved algorithms.

Before you enable FIPS-140 mode, it is important for you to first create, activate, and boot to a new Boot Environment (BE), using the `beadm(1M)` command. As some of the FIPS-140 required tests will cause a panic upon failure, it is important to have another BE you can boot to get your system up and running while issues with the FIPS-140 boundary are diagnosed.

Upon completion of this subcommand, a message is issued to inform the administrator that any plugins added that are not within the boundary might invalidate FIPS compliance and to check the Security Policies for those plugins.

The system will require a reboot to perform Power-Up Self Tests that include a cryptographic algorithm test and a software integrity test.

`cryptoadm list fips-140`
Display the current setting of FIPS-140 mode in the Cryptographic Framework and for hardware providers. The status of FIPS-140 mode is `enabled` or `disabled`. The default FIPS-140 mode is `disabled`.

`cryptoadm refresh`
`cryptoadm start`
`cryptoadm stop`
Private interfaces for use by `smf(5)`, these must not be used directly.

`cryptoadm -help`
Display the command usage.

Operands provider=*provider-name*
A user-level provider (a PKCS11 shared library), a kernel software provider (a loadable kernel software module), or a kernel hardware provider (a cryptographic hardware device).

A valid value of the *provider* operand is one entry from the output of a command of the form: `cryptoadm` *list*. A *provider* operand for a user-level provider is an absolute pathname of the corresponding shared library. A *provider* operand for a kernel software provider contains a base name only. A *provider* operand for a kernel hardware provider is in a "*name*/*number*" form.

mechanism=*mechanism-list*
A comma separated list of one or more PKCS #11 mechanisms. A process for implementing a cryptographic operation as defined in PKCS #11 specification. You can substitute `all` for *mechanism-list*, to specify all mechanisms on a provider. See the discussion of the `all` keyword, below.

provider-feature
A cryptographic framework feature for the given provider. Currently only `random` is accepted as a feature. For a user-level provider, disabling the random feature makes the PKCS #11 routines `C_GenerateRandom` and `C_SeedRandom` unavailable from the provider. For a kernel provider, disabling the random feature prevents `/dev/random` from gathering random numbers from the provider.

`all`
The keyword `all` can be used with with the `disable`, `enable`, and `install` subcommands to operate on all provider features.

`token=`*token-label*
The label of a token in one of the providers in the cryptographic framework.

A valid value of the token operand is an item displayed under "Token Label" from the output of the command `cryptoadm list -v`.

`slot=`*slot-description*
The description of a slot in one of the providers in the cryptographic framework.

A valid value of the slot operand is an item displayed under "Description" from the output of the command `cryptoadm list -v`.

`default-keystore`
The keyword `default-keystore` is valid only for metaslot. Specify this keyword to set the persistent object store for metaslot back to using the default store.

`auto-key-migrate`
The keyword auto-key-migrate is valid only for metaslot. Specify this keyword to configure whether metaslot is allowed to move sensitive token objects from the token object slot to other slots for performing cryptographic operations.

The keyword `all` can be used in two ways with the `disable` and `enable` subcommands:

- You can substitute `all` for `mechanism=`*mechanism-list* and any other provider-features, as in:

```
# cryptoadm enable provider=dca/0 all
```

This command enables the mechanisms on the provider *and* any other provider-features, such as `random`.

- You can also use `all` as an argument to `mechanism`, as in:

  ```
  # cryptoadm enable provider=des mechanism=all
  ```

  ...which enables all mechanisms on the provider, but enables no other provider–features, such as `random`.

Examples

EXAMPLE 1 Display List of Providers Installed in System

The following command displays a list of all installed providers:

```
example% cryptoadm list
user-level providers:
/usr/lib/security/$ISA/pkcs11_kernel.so
/usr/lib/security/$ISA/pkcs11_softtoken.so
/opt/lib/libcryptoki.so.1
/opt/system/core-osonn/lib/$ISA/libpkcs11.so.1

kernel providers:
    des
    aes
    bfish
    sha1
    md5
    dca/0
```

EXAMPLE 2 Display Mechanism List for md5 Provider

The following command is a variation of the `list` subcommand:

```
example% cryptoadm list -m provider=md5
md5: CKM_MD5,CKM_MD5_HMAC,CKM_MD5_HMAC_GENERAL
```

EXAMPLE 3 Disable Specific Mechanisms for Kernel Software Provider

The following command disables mechanisms `CKM_DES3_ECB` and `CKM_DES3_CBC` for the kernel software provider `des`:

```
example# cryptoadm disable provider=des
```

EXAMPLE 4 Display Mechanism Policy for a Provider

The following command displays the mechanism policy for the `des` provider:

```
example% cryptoadm list -p provider=des
des: All mechanisms are enabled, except CKM_DES3_ECB, CKM_DES3_CBC
```

EXAMPLE 5 Enable Specific Mechanism for a Provider

The following command enables the `CKM_DES3_ECB` mechanism for the kernel software provider `des`:

```
example# cryptoadm enable provider=des mechanism=CKM_DES3_ECB
```

EXAMPLE 6 Install User-Level Provider

The following command installs a user-level provider:

```
example# cryptoadm install provider=/opt/lib/libcryptoki.so.1
```

EXAMPLE 7 Install User-Level Provider That Contains 32– and 64–bit Versions

The following command installs a user-level provider that contains both 32–bit and 64–bit versions:

```
example# cryptoadm install \
provider=/opt/system/core-osonn/lib/'$ISA'/libpkcs11.so.1
```

EXAMPLE 8 Uninstall a Provider

The following command uninstalls the `md5` provider:

```
example# cryptoadm uninstall provider=md5
```

EXAMPLE 9 Disable metaslot

The following command disables the metaslot feature in the cryptographic framework.

```
example# cryptoadm disable metaslot
```

EXAMPLE 10 Specify metaslot to Use Specified Token as Persistent Object Store

The following command specifies that metaslot use the Venus token as the persistent object store.

```
example# cryptoadm enable metaslot token="SUNW,venus"
```

Exit Status

The following exit values are returned:

0
Successful completion.

>0
An error occurred.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	See below.

The `start`, `stop`, and `refresh` options are Private interfaces. All other options and the utility name are Committed.

See Also `beadm`(1M), `logadm`(1M), `svcadm`(1M), `syslogd`(1M), `libpkcs11`(3LIB), `exec_attr`(4), `prof_attr`(4), `attributes`(5), `smf`(5), `random`(7D)

Oracle Solaris 11.1 Administration: Security Services

Solaris Security for Developer's Guide

Notes If a hardware provider's policy was made explicitly (that is, some of its mechanisms were disabled) and the hardware provider has been detached, the policy of this hardware provider is still listed.

`cryptoadm` assumes that, minimally, a 32–bit shared object is delivered for each user-level provider. If both a 32–bit and 64–bit shared object are delivered, the two versions must provide the same functionality. The same mechanism policy applies to both.

Name datadm – maintain DAT static registry file

Synopsis

```
/usr/bin/datadm [-v] [-u] [-a service_provider.conf]
     [-r service_provider.conf]
```

Description The `datadm` utility maintains the DAT static registry file, `dat.conf(4)`.

This administrative configuration program allows uDAPL service providers to add and remove themselves to the `dat.conf` file.

You can add or remove interface adapters that a service provider supports from a system after its installation. You can use `datadm` to update the `dat.conf` file to reflect the current state of the system. A new set of interface adapters for all the service providers currently installed is regenerated.

Options The following options are supported:

`-a service_provider.conf`
Enumerate each device entry in the `service_provider.conf(4)` file into a list of interface adapters, that is, interfaces to external network that are available to uDAPL consumers.

`-r service_provider.conf`
Remove the list of interface adapters that corresponds to the device entry in the `service_provider.conf(4)` file.

`-u`
Update the `dat.conf` to reflect the current state of the system with an up to date set of interface adapters for the service providers that are currently listed in the DAT static registry.

`-v`
Display the DAT static registry file, `dat.conf`.

Examples **EXAMPLE 1** Enumerating a Device Entry

The following example enumerates a device entry in the `service_provider.conf(4)` file into interface adapters in the `dat.conf(4)` file.

Assume that `SUNW` has a service provider library that supports the device `hermon`. It has a `service_provider.conf(4)` file installed in the directory `/usr/share/dat/SUNWudaplt.conf` with a single entry as follows:

```
driver_name=hermon u1.2 nonthreadsafe default\
    udapl_tavor.so.1 SUNW.1.0 ""
```

`hermon` is an Infiniband Host Channel Adapter with two ports. Both IB ports exist in a single IB partition, `0x8001`. If an IB partition is created and plumbed to each port (with the names `p8001.ibd0` and `p8001.ibd1`), there will be two IB partition instances. See `dladm(1M)` for more information on creating IB partition data links.

EXAMPLE 1 Enumerating a Device Entry *(Continued)*

```
# dladm show-part

LINK            PKEY      OVER       STATE      FLAGS
p8001.ibd0      8001      ibd0       unknown    ----
p8001.ibd1      8001      ibd1       unknown    ----
```

Running the command:

```
# datadm -a /usr/share/dat/SUNWudaplt.conf
```

...appends two new entries (if they do not already exist) in the `/etc/dat/dat.conf` file:

```
p8001.ibd0 u1.2 nonthreadsafe default udapl_tavor.so.1 SUNW.1.0 ""
"driver_name=hermon"
p8001.ibd1 u1.2 nonthreadsafe default udapl_tavor.so.1 SUNW.1.0 ""
"driver_name=hermon"
```

EXAMPLE 2 Updating the `dat.conf` to Reflect the Current State of the System

A new IB partition, `0x8002`, is added to the above example covering port 1 of the Host Channel Adapter. If a new IB partition is created on the port 1/partition `0x8002` with the partition link name specified as `p8002.ibd0`, there will be a third IB partition instance: `p8002.ibd0`.

```
# dladm show-part
LINK            PKEY      OVER       STATE      FLAGS
p8001.ibd0      8001      ibd0       unknown    ----
p8001.ibd1      8001      ibd1       unknown    ----
p8002.ibd0      8000      ibd0       unknown    ----
```

Running `datadm -u` command, updates the `/etc/dat/dat.conf` file with a new entry added reflecting the current state of the system.

`datadm -v` shows that there are now three entries in the `/etc/dat/dat.conf` file:

```
p8001.ibd0 u1.2 nonthreadsafe default udapl_tavor.so.1 SUNW.1.0 ""
"driver_name=hermon"
p8001.ibd1 u1.2 nonthreadsafe default udapl_tavor.so.1 SUNW.1.0 ""
"driver_name=hermon"
p8002.ibd0 u1.2 nonthreadsafe default udapl_tavor.so.1 SUNW.1.0 ""
"driver_name=hermon"
```

Files `/etc/dat/dat.conf`
DAT static registry file

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/io/infiniband/udapl
Interface Stability	Committed

See Also dladm(1M), pkgadd(1M), pkgrm(1M), libdat(3LIB), dat.conf(4), service_provider.conf(4), attributes(5)

Name dcs – domain configuration server

Synopsis

```
/usr/lib/dcs [-s sessions]
     [ [-a auth] [-e encr] [-u esp_auth] ] [-l]
```

Description The Domain Configuration Server (DCS) is a daemon process that runs on Sun servers that support remote Dynamic Reconfiguration (DR) clients. It is started by the Service Management Facility (see smf(5)) when the first DR request is received from a client connecting to the network service `sun-dr`. After the DCS accepts a DR request, it uses the libcfgadm(3LIB) interface to execute the DR operation. After the operation is performed, the results are returned to the client.

The DCS listens on the network service labeled `sun-dr`. Its underlying protocol is TCP. It is invoked as a server program by the SMF using the TCP transport. The fault management resource identifier (FMRI) for DCS is:

```
svc:/platform/sun4u/dcs:default
```

If you disable this service, DR operations initiated from a remote host fail. There is no negative impact on the server.

Security for the DCS connection is provided differently based upon the architecture of the system. The SMF specifies the correct options when invoking the DCS daemon, based upon the current architecture. For all architectures, security is provided on a per-connection basis.

The DCS daemon has no security options that are applicable when used on a Sun Enterprise 10000 system. So there are no options applicable to that architecture.

The security options for Sun Fire high-end systems are based on IPsec options defined as SMF properties. These options include the `-a` *auth*, `-e` *encr*, and `-u` *esp_auth* options, and can be set using the svccfg(1M) command. These options must match the IPsec policies defined for DCS on the system controller. Refer to the `kmd(1M)` man page in the *System Management Services (SMS) Reference Manual*. The `kmd(1M)` man page is not part of the SunOS man page collection.

Security on SPARC Enterprise Servers is not configurable. The DCS daemon uses a platform-specific library to configure its security options when running on such systems. The `-l` option is provided by the SMF when invoking the DCS daemon on SPARC Enterprise Servers. No other security options to the DCS daemon should be used on SPARC Enterprise Servers.

Options The following options are supported:

`-a` *auth* Controls the IPsec Authentication Header (AH) algorithm. *auth* can be one of `none`, `md5`, or `sha1`.

`-e` *encr* Controls the IPsec Encapsulating Security Payload (ESP) encryption algorithm. *encr* can be one of `none`, `des`, or `3des`.

`-l`	Enables the use of platform-specific security options on SPARC Enterprise Servers.
`-s` *sessions*	Sets the number of active sessions that the DCS allows at any one time. When the limit is reached, the DCS stops accepting connections until active sessions complete the execution of their DR operation. If this option is not specified, a default value of 128 is used.
`-u` *esp_auth*	Controls the IPsec Encapsulating Security Payload (ESP) authentication algorithm. *esp_auth* can be one of `none`, `md5`, or `sha1`.

Examples

EXAMPLE 1 Setting an IPSec Option

The following command sets the Authentication Header algorithm for the DCS daemon to use the HMAC-MD5 authentication algorithm. These settings are only applicable for using the DCS daemon on a Sun Fire high-end system.

```
# svccfg -s svc:/platform/sun4u/dcs setprop dcs/ah_auth = "md5"
# svccfg -s svc:/platform/sun4u/dcs setprop dcs/esp_encr = "none"
# svccfg -s svc:/platform/sun4u/dcs setprop dcs/esp_auth = "none"
# svcadm refresh svc:/platform/sun4u/dcs
```

Errors

The DCS uses `syslog(3C)` to report status and error messages. All of the messages are logged with the `LOG_DAEMON` facility. Error messages are logged with the `LOG_ERR` and `LOG_NOTICE` priorities, and informational messages are logged with the `LOG_INFO` priority. The default entries in the `/etc/syslog.conf` file log all of the DCS error messages to the `/var/adm/messages` log.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/domain-configuration/sparc-enterprise, SUNWdcsr
Interface Stability	Committed

See Also

`svcs(1)`, `cfgadm_sbd(1M)`, `svcadm(1M)`, `svccfg(1M)`, `syslog(3C)`, `config_admin(3CFGADM)`, `libcfgadm(3LIB)`, `syslog.conf(4)`, `attributes(5)`, `smf(5)`, `dr(7d)`

Notes

The `dcs` service is managed by the service management facility, `smf(5)`, under the fault management resource identifier (FMRI):

```
svc:/platform/sun4u/dcs:default
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. The service's status can be queried using the `svcs(1)` command.

Name dd – convert and copy a file

Synopsis `/usr/bin/dd [operand=value]...`

Description The `dd` utility copies the specified input file to the specified output with possible conversions. The standard input and output are used by default. The input and output block sizes may be specified to take advantage of raw physical I/O. Sizes are specified in bytes; a number may end with `k`, `b`, or `w` to specify multiplication by 1024, 512, or 2, respectively. Numbers may also be separated by `x` to indicate multiplication.

The `dd` utility reads the input one block at a time, using the specified input block size. `dd` then processes the block of data actually returned, which could be smaller than the requested block size. `dd` applies any conversions that have been specified and writes the resulting data to the output in blocks of the specified output block size.

`cbs` is used only if `ascii`, `asciib`, `unblock`, `ebcdic`, `ebcdicb`, `ibm`, `ibmb`, or `block` conversion is specified. In the first two cases, `cbs` characters are copied into the conversion buffer, any specified character mapping is done, trailing blanks are trimmed, and a `NEWLINE` is added before sending the line to output. In the last three cases, characters up to `NEWLINE` are read into the conversion buffer and blanks are added to make up an output record of size `cbs`. ASCII files are presumed to contain `NEWLINE` characters. If `cbs` is unspecified or `0`, the `ascii`, `asciib`, `ebcdic`, `ebcdicb`, `ibm`, and `ibmb` options convert the character set without changing the input file's block structure. The `unblock` and `block` options become a simple file copy.

After completion, `dd` reports the number of whole and partial input and output blocks.

Operands The following operands are supported:

`if=`*file* — Specifies the input path. Standard input is the default.

`of=`*file* — Specifies the output path. Standard output is the default. If the `seek=expr` conversion is not also specified, the output file will be truncated before the copy begins, unless `conv=notrunc` is specified. If `seek=expr` is specified, but `conv=notrunc` is not, the effect of the copy will be to preserve the blocks in the output file over which `dd` seeks, but no other portion of the output file will be preserved. (If the size of the seek plus the size of the input file is less than the previous size of the output file, the output file is shortened by the copy.)

`ibs=`*n* — Specifies the input block size in *n* bytes (default is `512`).

`obs=`*n* — Specifies the output block size in *n* bytes (default is `512`).

`bs=`*n* — Sets both input and output block sizes to *n* bytes, superseding `ibs=` and `obs=`. If no conversion other than `sync`, `noerror`, and `notrunc` is specified, each input block is copied to the output as a single block without aggregating short blocks.

`cbs=`*n*
: Specifies the conversion block size for `block` and `unblock` in bytes by *n* (default is `0`). If `cbs=` is omitted or given a value of `0`, using `block` or `unblock` produces unspecified results.

  This option is used only if ASCII or EBCDIC conversion is specified. For the `ascii` and `asciib` operands, the input is handled as described for the `unblock` operand except that characters are converted to ASCII before the trailing `SPACE` characters are deleted. For the `ebcdic`, `ebcdicb`, `ibm`, and `ibmb` operands, the input is handled as described for the `block` operand except that the characters are converted to EBCDIC or IBM EBCDIC after the trailing `SPACE` characters are added.

`files=`*n*
: Copies and concatenates *n* input files before terminating (makes sense only where input is a magnetic tape or similar device).

`skip=`*n*
: Skips *n* input blocks (using the specified input block size) before starting to copy. On seekable files, the implementation reads the blocks or seeks past them. On non-seekable files, the blocks are read and the data is discarded.

`iseek=`*n*
: Seeks *n* blocks from beginning of input file before copying (appropriate for disk files, where `skip` can be incredibly slow).

`oseek=`*n*
: Seeks *n* blocks from beginning of output file before copying.

`seek=`*n*
: Skips *n* blocks (using the specified output block size) from beginning of output file before copying. On non-seekable files, existing blocks are read and space from the current end-of-file to the specified offset, if any, is filled with null bytes. On seekable files, the implementation seeks to the specified offset or reads the blocks as described for non-seekable files.

`count=`*n*
: Copies only *n* input blocks.

`conv=`*value*[`,`*value*`...`]
: Where *value*s are comma-separated symbols from the following list:

  `ascii`
  : Converts EBCDIC to ASCII.

  `asciib`
  : Converts EBCDIC to ASCII using BSD-compatible character translations.

  `ebcdic`
  : Converts ASCII to EBCDIC. If converting fixed-length ASCII records without NEWLINEs, sets up a pipeline with `dd conv=unblock` beforehand.

  `ebcdicb`
  : Converts ASCII to EBCDIC using BSD-compatible character translations. If converting fixed-length

ASCII records without NEWLINEs, sets up a pipeline with dd conv=unblock beforehand.

ibm — Slightly different map of ASCII to EBCDIC. If converting fixed-length ASCII records without NEWLINEs, sets up a pipeline with dd conv=unblock beforehand.

ibmb — Slightly different map of ASCII to EBCDIC using BSD-compatible character translations. If converting fixed-length ASCII records without NEWLINEs, sets up a pipeline with dd conv=unblock beforehand.

The ascii (or asciib), ebcdic (or ebcdicb), and ibm (or ibmb) values are mutually exclusive.

block — Treats the input as a sequence of NEWLINE-terminated or EOF-terminated variable-length records independent of the input block boundaries. Each record is converted to a record with a fixed length specified by the conversion block size. Any NEWLINE character is removed from the input line. SPACE characters are appended to lines that are shorter than their conversion block size to fill the block. Lines that are longer than the conversion block size are truncated to the largest number of characters that will fit into that size. The number of truncated lines is reported.

unblock — Converts fixed-length records to variable length. Reads a number of bytes equal to the conversion block size (or the number of bytes remaining in the input, if less than the conversion block size), delete all trailing SPACE characters, and append a NEWLINE character.

The block and unblock values are mutually exclusive.

lcase — Maps upper-case characters specified by the LC_CTYPE keyword tolower to the corresponding lower-case character. Characters for which no mapping is specified are not modified by this conversion.

ucase — Maps lower-case characters specified by the LC_CTYPE keyword toupper to the corresponding upper-case character. Characters for which no mapping is specified are not modified by this conversion.

The `lcase` and `ucase` symbols are mutually exclusive.

`swab` Swaps every pair of input bytes. If the current input record is an odd number of bytes, the last byte in the input record is ignored.

`noerror` Does not stop processing on an input error. When an input error occurs, a diagnostic message is written on standard error, followed by the current input and output block counts in the same format as used at completion. If the `sync` conversion is specified, the missing input is replaced with null bytes and processed normally. Otherwise, the input block will be omitted from the output.

`notrunc` Does not truncate the output file. Preserves blocks in the output file not explicitly written by this invocation of `dd`. (See also the preceding `of=`*file* operand.)

`sync` Pads every input block to the size of the `ibs=` buffer, appending null bytes. (If either `block` or `unblock` is also specified, appends `SPACE` characters, rather than null bytes.)

If operands other than `conv=` are specified more than once, the last specified `operand=`*value* is used.

For the `bs=`, `cbs=`, `ibs=`, and `obs=` operands, the application must supply an expression specifying a size in bytes. The expression, `expr`, can be:

1. a positive decimal number
2. a positive decimal number followed by `k`, specifying multiplication by 1024
3. a positive decimal number followed by `b`, specifying multiplication by 512
4. two or more positive decimal numbers (with or without `k` or `b`) separated by `x`, specifying the product of the indicated values.

All of the operands will be processed before any input is read.

Usage See `largefile(5)` for the description of the behavior of `dd` when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

Examples **EXAMPLE 1** Copying from one tape drive to another

The following example copies from tape drive `0` to tape drive `1`, using a common historical device naming convention.

EXAMPLE 1 Copying from one tape drive to another *(Continued)*

```
example% dd if=/dev/rmt/0h  of=/dev/rmt/1h
```

EXAMPLE 2 Stripping the first 10 bytes from standard input

The following example strips the first 10 bytes from standard input:

```
example% dd ibs=10  skip=1
```

EXAMPLE 3 Reading a tape into an ASCII file

This example reads an EBCDIC tape blocked ten 80-byte EBCDIC card images per block into the ASCII file x:

```
example% dd if=/dev/tape of=x ibs=800 cbs=80 conv=ascii,lcase
```

EXAMPLE 4 Using conv=sync to write to tape

The following example uses `conv=sync` when writing to a tape:

```
example% tar cvf - . | compress | dd obs=1024k of=/dev/rmt/0 conv=sync
```

Environment Variables

See `environ(5)` for descriptions of the following environment variables that affect the execution of `dd`: `LANG`, `LC_ALL`, `LC_CTYPE`, `LC_MESSAGES`, and `NLSPATH`.

Exit Status

The following exit values are returned:

`0` The input file was copied successfully.

`>0` An error occurred.

If an input error is detected and the `noerror` conversion has not been specified, any partial output block will be written to the output file, a diagnostic message will be written, and the copy operation will be discontinued. If some other error is detected, a diagnostic message will be written and the copy operation will be discontinued.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os
Interface Stability	Committed
Standard	See `standards(5)`.

See Also

`cp(1)`, `sed(1)`, `tr(1)`, `attributes(5)`, `environ(5)`, `largefile(5)`, `standards(5)`

Diagnostics `f+p records in(out)` numbers of full and partial blocks read(written)

Notes Do not use `dd` to copy files between file systems having different block sizes.

Using a blocked device to copy a file will result in extra nulls being added to the file to pad the final block to the block boundary.

When `dd` reads from a pipe, using the `ibs=X` and `obs=Y` operands, the output will always be blocked in chunks of size Y. When `bs=Z` is used, the output blocks will be whatever was available to be read from the pipe at the time.

When using `dd` to copy files to a tape device, the file size must be a multiple of the device sector size (for example, 512 Kbyte). To copy files of arbitrary size to a tape device, use `tar(1)` or `cpio(1)`.

For `SIGINT`, `dd` writes status information to standard error before exiting. It takes the standard action for all other signals.

Name ddu – GUI-based device driver utility

Synopsis

```
/usr/bin/ddu [--silent]
```

Description The GUI-based device driver utility, `ddu`, provides information about devices on a system running the Oracle Solaris operating system. `ddu` enables a user to connect to the Image Packaging System (IPS) and search device drivers for the devices that do not have drivers attached to them. By doing this, `ddu` gives the user an opportunity to install missing drivers.

`ddu` is a GUI, with an easy-to-use interface. It has a text-based counterpart, `ddu-text(1M)`, which is described in its own man page.

Options The following options is supported:

`--silent`
: `ddu` runs silently. If it finds devices that are missing drivers a notification is posted. Clicking on the notification displays the DDU GUI.

Examples

EXAMPLE 1 Launching the `ddu` GUI

The following command launches the `ddu` GUI.

```
% ddu
```

EXAMPLE 2 Launching `ddu` in Silent Mode

The following command launches `ddu` in silent mode.

```
% ddu --silent
```

Exit Status

`0`
: Application exited successfully.

`>0`
: Application exited with a failure.

Files

`/usr/bin/ddu`
: Executable for `ddu`.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	diagnostic/ddu
Interface Stability	External

Additional availability attribute values are: `diagnostic/ddu/data`, `diagnostic/ddu/library`, and `diagnostic/ddu/locale`.

See Also ddu-text(1M), attributes(5)

Name ddu-text – text-based device driver utility

Synopsis

```
ddu-text
```

Description The text-based device driver utility, `ddu-text`, provides information about devices on a system running the Oracle Solaris operating system. `ddu-text` enables a user to connect to the Image Packaging System (IPS) and search device drivers for the devices that do not have drivers attached to them. By doing this, `ddu-text` gives the user an opportunity to install missing drivers.

`ddu-text` is text-based. It has a GUI counterpart, ddu(1M), which is described in its own man page.

`ddu-text` has no command-line options.

Examples **EXAMPLE 1** Invoking the `ddu-text`

The following command invokes `ddu-text`.

```
% ddu-text
```

Exit Status

`0`
Application exited successfully.

`>0`
Application exited with a failure.

Files

`/usr/bin/ddu-text`
Executable for `ddu-text`.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	diagnostic/ddu
Interface Stability	External

Additional availability attribute values are: `diagnostic/ddu/data`, `diagnostic/ddu/library`, and `diagnostic/ddu/locale`.

See Also ddu(1M), attributes(5)

Name devchassisd – devchassis daemon

Synopsis

```
/usr/lib/devchassis/devchassisd [-d]
```

Description The `devchassis` daemon, `devchassisd`, provides user-level services for the management of the `/dev/chassis` name space. It is a system daemon started by the Service Management Facility (see `smf(5)`). Its fault management resource identifier (FMRI) is:

```
svc:/system/devchassis:daemon
```

Note that `devchassisd` is a Consolidation Private interface. See `attributes(5)`.

Options The following option is supported:

`-d`, `--debug`
: Run the daemon in standalone debug mode. Messages will be displayed on the controlling terminal instead of to syslog. And increased verbosity will be enabled to display more details about the internal operations of the daemon.

Examples

EXAMPLE 1 Enabling the devchassis Service

The following command enables the `devchassis` service:

```
# svcadm enable svc:/system/devchassis
```

EXAMPLE 2 Disabling the devchassis Service

The following command disables the `devchassis` service:

```
# svcadm disable svc:/system/devchassis
```

Errors The `devchassisd` daemon uses `syslog(3C)` to report status and error messages. All of the messages are logged with the `LOG_DAEMON` facility. Error messages are logged with the `LOG_ERR` and `LOG_NOTICE` priorities, and informational messages are logged with the `LOG_INFO` priority. The default entries in the `/etc/syslog.conf` file log all of the `devchassisd` daemon error messages to the `/var/adm/messages` log.

Files

`/usr/lib/devchassis/devchassisd`
: `devchassisd` daemon binary.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os
Interface Stability	Consolidation Private

See Also svcs(1), svcadm(1M), syslog(3C), syslog.conf(4), attributes(5), smf(5), devchassis(7FS)

Notes The devchassis service is managed by the service management facility, smf(5), under the fault management resource identifier (FMRI):

```
svc:/system/devchassis
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using svcadm(1M). The service's status can be queried using the svcs(1) command.

The devchassis service is enabled by default.

Name devfsadm, devfsadmd – administration command for /dev

Synopsis

```
/usr/sbin/devfsadm [-C] [-c device_class] [-i driver_name]
     [ -n] [-r root_dir] [-s] [-t table_file] [-u] [-v]

/usr/lib/devfsadm/devfsadmd
```

Description `devfsadm` maintains the `/dev` namespace. It replaces the previous suite of `devfs` administration tools including `drvconfig(1M)`, `disks(1M)`, `tapes(1M)`, `ports(1M)`, `audlinks`(1M), and `devlinks(1M)`.

The default operation is to attempt to load every driver in the system and attach to all possible device instances. Next, `devfsadm` creates logical links to device nodes in `/dev` and `/devices` and loads the device policy.

`devfsadmd(1M)` is the daemon version of `devfsadm`(1M). The daemon is started during system startup and is responsible for handling both reconfiguration boot processing and updating `/dev` and `/devices` in response to dynamic reconfiguration event notifications from the kernel.

For compatibility purposes, `drvconfig`(1M), `disks`(1M), `tapes`(1M), `ports`(1M), `audlinks`(1M), and `devlinks`(1M) are implemented as links to `devfsadm`.

In addition to managing `/dev`, `devfsadm` also maintains the `path_to_inst(4)` database.

Options The following options are supported:

`-C`
Cleanup mode. Prompt `devfsadm` to cleanup dangling `/dev` links that are not normally removed. If the `-c` option is also used, `devfsadm` only cleans up for the listed devices' classes.

`-c` *device_class*
Restrict operations to devices of class *device_class*. Solaris defines the following values for *device_class*: `disk`, `tape`, `port`, `audio`, and `pseudo`. This option might be specified more than once to specify multiple device classes.

`-i` *driver_name*
Configure only the devices for the named driver, *driver_name*.

`-n`
Do not attempt to load drivers or add new nodes to the kernel device tree.

`-s`
Suppress any changes to `/dev`. This is useful with the `-v` option for debugging.

`-t` *table_file*
Read an alternate `devlink.tab` file. `devfsadm` normally reads `/etc/devlink.tab`.

`-u`
Activate and attach devices for drivers added with `add_drv(1M) -u`. Cannot be used together with `-n` or `-r`.

`-r` *root_dir*
Presume that the `/dev` directory trees are found under *root_dir*, not directly under `root` (`/`). No other use or assumptions are made about *root_dir*.

`-v`
Print changes to `/dev` in verbose mode.

Exit Status The following exit values are returned:

`0`
Successful completion.

`1`
An error occurred.

Files
`/devices`
device nodes directory

`/dev`
logical symbolic links to /devices

`/usr/lib/devfsadm/devfsadmd`
devfsadm daemon

`/dev/.devfsadm_dev.lock`
update lock file

`/dev/.devfsadm_daemon.lock`
daemon lock file

`/etc/security/device_policy`
device policy file

`/etc/security/extra_privs`
additional device privileges

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also `svcs(1)`, `add_drv(1M)`, `modinfo(1M)`, `modload(1M)`, `modunload(1M)`, `rem_drv(1M)`, `svcadm(1M)`, `tapes(1M)`, `path_to_inst(4)`, `attributes(5)`, `privileges(5)`, `smf(5)`, `devfs(7FS)`

Notes This document does not constitute an API. The `/devices` directory might not exist or might have different contents or interpretations in a future release. The existence of this notice does not imply that any other documentation that lacks this notice constitutes an API.

`devfsadm` no longer manages the `/devices` name space. See `devfs`(7FS).

As a daemon to support hot-plug and synchronous device naming, `devfsadm` is managed by the service management facility, `smf`(5), under the service identifier:

```
svc:/system/devfsadm:default
```

The status of the service can be queried using the `svcs`(1) command.

Name device_allocate – enable and disable device allocation

Synopsis

```
svc:/system/device/allocate:default
```

Description The `device_allocate` service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/system/device/allocate:default
```

An administrator with the Devices Security Rights Profile can enable or disable this service to provide for or remove the device allocation functionality as described in `allocate(1)`.

Examples **EXAMPLE 1** Enabling Device Allocation

The following command enables device allocation.

```
# svcadm enable svc:/system/device/allocate:default
```

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Volatile

See Also `allocate(1)`, `deallocate(1)`, `list_devices(1)`, `devfsadm(1M)`, `svcadm(1M)`, `attributes(5)`, `smf(5)`

Name device_remap – administer the Solaris I/O remapping feature

Synopsis `/usr/platform/sun4v/sbin/device_remap [-v | -R` *dir*`]`

Description Certain multi-node sun4v platforms, such as T5440 and T5240 servers, have an integrated PCI topology that cause the I/O device paths to change in a CPU node failover condition. The device remapping script, `device_remap`, remaps the device paths in `/etc/path_to_inst` file and the symlinks under `/dev` to match the hardware.

Options The following options are supported:

`-v`
Displays the `/etc/path_to_inst` and `/dev` symlink changes.

`-R` *dir*
Perform remapping on the `/etc/path_to_inst` and `/etc/path_to_inst` files in the root image at *dir*.

Usage The primary function of `device_remap` is to remap the device paths in the `/etc/path_to_inst` file and the symlinks under `/dev` in a CPU node failover condition to match the hardware.

After adding CPU node(s) or removing CPU node(s), boot the system to the OBP prompt and use the following procedure:

1. Boot either the failsafe miniroot using: `boot -F failsafe` , or an install miniroot using `boot net -s` or similar command.
2. Mount the root disk as `/mnt`.
3. Change directory to the mounted root disk:

   ```
   # cd /mnt
   ```

4. Run `device_remap` script:

   ```
   # /mnt/usr/platform/sun4v/sbin/device_remap
   ```

5. Boot the system from disk.

All the error messages are self-explanatory, except for the error message “missing ioaliases node” which means the firmware on the system does not support device remapping.

Examples **EXAMPLE 1** Displaying Changes Following Failover

The following command displays the `path_to_inst` and `/dev` changes following a CPU node failover.

```
# device_remap -v
```

EXAMPLE 2 Changing Directory Prior to Any Changes

The following command changes the directory on which the boot image is mounted prior to making any changes.

```
# device_remap -R /newroot
```

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/library/platform
Interface Stability	Uncommitted

See Also

`boot(1M)`, `attributes(5)`

Name devinfo – print device specific information

Synopsis

```
/usr/sbin/devinfo -i device
/usr/sbin/devinfo -p device
```

Description The `devinfo` command is used to print device specific information about disk devices on standard out. The command can only be used by the superuser.

Options

`-i` Prints the following device information:

- Device name
- Software version (not supported and prints as `0`)
- Drive id number (not supported and prints as `0`)
- Device blocks per cylinder
- Device bytes per block
- Number of device partitions with a block size greater than zero

`-p` Prints the following device partition information:

- Device name
- Device major and minor numbers (in hexadecimal)
- Partition start block
- Number of blocks allocated to the partition
- Partition flag
- Partition tag

This command is used by various other commands to obtain device specific information for the making of file systems and determining partition information. If the device cannot be opened, an error message is reported.

Operands *device* Device name.

Exit Status

`0` Successful operation.

`2` Operation failed.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also `prtvtoc(1M)`, `attributes(5)`

Name devlinks – adds /dev entries for miscellaneous devices and pseudo-devices

Synopsis

```
/usr/sbin/devlinks [-d] [-r rootdir] [-t table-file]
```

Description devfsadm(1M) is now the preferred command for /dev and /devices and should be used instead of devlinks.

devlinks creates symbolic links from the /dev directory tree to the actual block- and character-special device nodes under the /devices directory tree. The links are created according to specifications found in the *table-file* (by default /etc/devlink.tab).

devlinks is called each time the system is reconfiguration-booted, and can only be run after drvconfig(1M) is run.

The *table-file* (normally /etc/devlink.tab) is an ASCII file, with one line per record. Comment lines, which must contain a hash character ('#') as their first character, are allowed. Each entry must contain at least two fields, but may contain three fields. Fields are separated by single TAB characters.

The fields are:

devfs-spec Specification of devinfo nodes that will have links created for them. This specification consists of one or more keyword-value pairs, where the keyword is separated from the value by an equal-sign ('='), and keyword-value pairs are separated from one another by semicolons.

The possible keywords are:

type The devinfo device type. Possible values are specified in ddi_create_minor_node(9F)

name The name of the node. This is the portion of the /devices tree entry name that occurs before the first '@' or ':' character.

addr[*n*] The address portion of a node name. This is the portion of a node name that occurs between the '@' and the ':' characters. It is possible that a node may have a name without an address part, which is the case for many of the pseudo-device nodes. If a number is given after the *addr* it specifies a match of a particular comma-separated subfield of the address field: *addr1* matches the first subfield, *addr2* matches the second, and so on. *addr0* is the same as *addr* and matches the whole field.

minor[*n*] The minor portion of a node name – the portion of the name after the ':'. As with *addr* above, a number after the *minor* keyword specifies a subfield to match.

Of these four specifications, only the *type* specification must always be present.

name Specification of the `/dev` links that correspond to the devinfo nodes. This field allows `devlinks` to determine matching `/dev` names for the `/devices` nodes it has found. The specification of this field uses escape-sequences to allow portions of the `/devices` name to be included in the `/dev` name, *or* to allow a counter to be used in creating node names. If a counter is used to create a name, the portion of the name before the counter must be specified absolutely, and all names in the `/dev/`-subdirectory that match (up to and including the counter) are considered to be subdevices of the same device. This means that they should all point to the same directory, `name` and `address` under the `/devices/`-tree

The possible escape-sequences are:

`\D` Substitute the `device-name` (name) portion of the corresponding devinfo node-name.

`\A`*n* Substitute the *n*th component of the `address` component of the corresponding devinfo node name. Sub-components are separated by commas, and sub-component `0` is the whole `address` component.

`\M`*n* Substitute the *n*th sub-component of the `minor` component of the corresponding devinfo node name. Sub-components are separated by commas, and sub-component `0` is the whole `minor` component.

`\N`*n* Substitute the value of a 'counter' starting at *n*. There can be only one counter for each dev-spec, and counter-values will be selected so they are as low as possible while not colliding with already-existing link names.

In a dev-spec the `counter` sequence should not be followed by a digit, either explicitly or as a result of another escape-sequence expansion. If this occurs, it would not be possible to correctly match already-existing links to their counter entries, since it would not be possible to unambiguously parse the already-existing `/dev`-name.

extra-dev-link Optional specification of an extra `/dev` link that points to the initial `/dev` link (specified in field 2). This field may contain a `counter` escape-sequence (as described for the *dev-spec* field) but may not contain any of the other escape-sequences. It provides a way to specify an alias of a particular `/dev` name.

Options The following options are supported:

`-d` Debugging mode – print out all `devinfo` nodes found, and indicate what links would be created, but do not do anything.

`-r` *rootdir* Use *rootdir* as the root of the `/dev` and `/devices` directories under which the device nodes and links are created. Changing the root directory does not change the location of the `/etc/devlink.tab` default table, nor is the root directory applied to the filename supplied to the `-t` option.

`-t` *table-file* Set the table file used by `devlinks` to specify the links that must be created. If this option is not given, `/etc/devlink.tab` is used. This option gives a way to instruct `devlinks` just to perform a particular piece of work, since just the links-types that `devlinks` is supposed to create can be specified in a command-file and fed to `devlinks`.

Errors If `devlinks` finds an error in a line of the *table-file* it prints a warning message on its standard output and goes on to the next line in the *table-file* without performing any of the actions specified by the erroneous rule.

If it cannot create a link for some filesystem-related reason it prints an error-message and continues with the current rule.

If it cannot read necessary data it prints an error message and continues with the next *table-file* line.

Examples **EXAMPLE 1** Using the /etc/devlink.tab Fields

The following are examples of the `/etc/devlink.tab` fields:

```
type=pseudo;name=win    win\M0
type=ddi_display    framebuffer/\M0    fb\N0
```

The first example states that all devices of type `pseudo` with a name component of `win` will be linked to `/dev/win`*x,* where *x* is the minor-component of the *devinfo-name* (this is always a single-digit number for the `win` driver).

The second example states that all devinfo nodes of type `ddi_display` will be linked to entries under the `/dev/framebuffer` directory, with names identical to the entire minor component of the `/devices` name. In addition an extra link will be created pointing from `/dev/fb`*n* to the entry under `/dev/framebuffer`. This entry will use a counter to end the name.

Files

`/dev`	entries for the miscellaneous devices for general use
`/devices`	device nodes
`/etc/devlink.tab`	the default rule-file

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also devfsadm(1M), attributes(5), devfs(7FS), ddi_create_minor_node(9F)

Bugs It is very easy to construct mutually-contradictory link specifications, or specifications that can never be matched. The program does not check for these conditions.

Name devnm – device name

Synopsis

```
/usr/sbin/devnm name [name]...
```

Description The `devnm` command identifies the special file associated with the mounted file system where the argument *name* resides. One or more *name* can be specified.

Examples **EXAMPLE 1** Using the devnm Command

Assuming that `/usr` is mounted on `/dev/dsk/c0t3d0s6`, the following command :

```
/usr/sbin/devnm /usr
```

produces:

```
/dev/dsk/c0t3d0s6 /usr
```

Files

```
/dev/dsk/*
```

```
/etc/mnttab
```

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also `mnttab(4)`, `attributes(5)`

Name devprop – display device properties

Synopsis

```
/usr/sbin/devprop [-n device-path] [-c separator] [-vq]
     [-{e|b|i|l|s}] [property[...]]
```

Description The `devprop` command displays named device properties from the device tree.

If a device path is specified on the command line, `devprop` displays device properties for that device node.

The boolean property prints out true (with the `-e` option) if it exists; it is otherwise false. Byte, `int`, and `int_64` property values display in hex format if one specifies type by means of options `-b`, `-i`, or `-l`; otherwise these values display in decimal format. Array property values are separated by a user-defined char.

Options The options below are supported. Note that the `-e`, `-b`, `-i`, `-l`, and `-s` options are mutually exclusive.

`-b`
The properties to be output are sequences of bytes (`DI_PROP_TYPE_BYTES`).

`-c` *separator*
Specifies the separator in an array property. Use double quotation marks (`" "`) to specify a space. The default separator is the plus sign (`+`) for the string type and period (`.`) for others.

`-e`
The properties to be output are booleans (`DI_PROP_TYPE_BOOLEAN`).

`-i`
The properties to be output are integers (`DI_PROP_TYPE_INT`).

`-l`
The properties to be output are 64–bit integers (`DI_PROP_TYPE_INT64`).

`-n` *device-path*
The path to a target device node for which properties are displayed. The default path is that of the root node (equivalent to specifying `-n /`).

`-q`
Specifies quoted output mode, in which string properties are output surrounded by double quotation marks (`"`).

`-s`
The properties to be output are strings (`DI_PROP_TYPE_STRING`) (the default).

`-v`
Specifies verbose mode, in which the name of the property is output before its value.

Operands The following operand is supported:

property...
Name of the property to be displayed.

Exit Status

`0`
No error occurred.

`non-zero`
An error occurred.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os
Interface Stability	See below.

The command invocation and output are both Volatile. The location of the utility is Committed.

See Also `prtconf(1M)`, `libdevinfo(3LIB)`, `attributes(5)`

Name df – displays number of free disk blocks and free files

Synopsis

```
df [-F FSType] [-abeghklntPVvZ]
     [-o FSType-specific_options]
     [block_device | directory | file | resource ...]
```

Description The `df` utility displays the amount of disk space occupied by mounted or unmounted file systems, the amount of used and available space, and how much of the file system's total capacity has been used. The file system is specified by device, or by referring to a file or directory on the specified file system.

Used without operands or options, `df` reports on all mounted file systems.

`df` may not be supported for all *FSTypes*.

If `df` is run on a networked mount point that the automounter has not yet mounted, the file system size will be reported as zero. As soon as the automounter mounts the file system, the sizes will be reported correctly.

Options The following options are supported:

`-a`
Reports on all file systems including ones whose entries in `/etc/mnttab` (see `mnttab(4)`) have the `ignore` option set.

`-b`
Prints the total number of kilobytes free.

`-e`
Prints only the number of files free.

`-F` *FSType*
Specifies the *FSType* on which to operate. The `-F` option is intended for use with unmounted file systems. The *FSType* should be specified here or be determinable from `/etc/vfstab` (see `vfstab(4)`) by matching the *directory*, *block_device*, or *resource* with an entry in the table, or by consulting `/etc/default/fs`. See `default_fs(4)`.

`-g`
Prints the entire `statvfs(2)` structure. This option is used only for mounted file systems. It can not be used with the `-o` option. This option overrides the `-b`, `-e`, `-k`, `-n`, `-P`, and `-t` options.

`-h`
Like `-k`, except that sizes are in a more human readable format. The output consists of one line of information for each specified file system. This information includes the file system name, the total space allocated in the file system, the amount of space allocated to existing files, the total amount of space available for the creation of new files by unprivileged users, and the percentage of normally available space that is currently allocated to all files on the file system. All sizes are scaled to a human readable format, for example, `14K`, `234M`, `2.7G`, or `3.0T`. Scaling is done by repetitively dividing by `1024`.

This option overrides the `-b`, `-e`, `-g`, `-k`, `-n`, `-t`, and `-V` options. This option only works on mounted filesystems and can not be used together with `-o` option.

`-k`
Prints the allocation in kbytes. The output consists of one line of information for each specified file system. This information includes the file system name, the total space allocated in the file system, the amount of space allocated to existing files, the total amount of space available for the creation of new files by unprivileged users, and the percentage of normally available space that is currently allocated to all files on the file system. This option overrides the `-b`, `-e`, `-n`, and `-t` options and may not be used together with the `-v` option.

`-l`
Reports on local file systems only. This option is used only for mounted file systems. It can not be used with the `-o` option.

`-n`
Prints only the *FSType* name. Invoked with no operands, this option prints a list of mounted file system types. This option is used only for mounted file systems. It can not be used with the `-o` option.

`-o` *FSType-specific_options*
Specifies *FSType-specific* options. These options are comma-separated, with no intervening spaces. See the manual page for the *FSType-specific* command for details.

`-t`
Prints full listings with totals. This option overrides the `-b`, `-e`, and `-n` options.

`-P`
Same as `-h` except in 512-byte units.

`-V`
Echoes the complete set of file system specific command lines, but does not execute them. The command line is generated by using the options and operands provided by the user and adding to them information derived from `/etc/mnttab`, `/etc/vfstab`, or `/etc/default/fs`. This option may be used to verify and validate the command line.

`-v`
Like `-k`, except that sizes are displayed in multiples of the smallest block size supported by each specified file system.

This option may not be used with the `-k` option.

The output consists of one line of information for each file system. This one line of information includes the following:

- the file system's mount point
- the file system's name
- the total number of blocks allocated to the file system
- the number of blocks allocated to existing files

- the number of blocks available for the creation of new files by unprivileged users
- the percentage of blocks in use by files

`-Z`
Displays mounts in all visible zones. By default, `df` displays mounts located only within the current zone. This option has no effect in a non-global zone.

Operands The `df` utility interprets operands according to the following precedence: *block_device*, *directory*, *file*, *resource*. The following operands are supported:

block_device
Represents a block special device (for example, `/dev/dsk/c1d0s7`).

directory
Represents a valid directory name. `df` reports on the file system that contains *directory*.

file
Represents a valid file name. `df` reports on the file system that contains *file*.

resource
Represents an NFS resource name.

Usage See `largefile(5)` for the description of the behavior of `df` when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

Examples **EXAMPLE 1** Executing the `df` command

The following example shows the `df` command and its output:

```
example% /usr/bin/df

/                  (/dev/dsk/c0t0d0s0 ):  287530 blocks    92028 files
/system/contract   (ctfs              ):       0 blocks 2147483572 files
/system/object     (objfs             ):       0 blocks 2147483511 files
/usr               (/dev/dsk/c0t0d0s6 ): 1020214 blocks   268550 files
/proc              (/proc             ):       0 blocks      878 files
/dev/fd            (fd                ):       0 blocks        0 files
/etc/mnttab        (mnttab            ):       0 blocks        0 files
/var/run           (swap              ):  396016 blocks     9375 files
/tmp               (swap              ):  396016 blocks     9375 files
/opt               (/dev/dsk/c0t0d0s5 ):  381552 blocks    96649 files
/export/home       (/dev/dsk/c0t0d0s7 ):  434364 blocks   108220 files
```

where the columns represent the mount point, device (or "filesystem", according to `df -k`), free blocks, and free files, respectively. For contract file systems, `/system/contract` is the mount point, `ctfs` is the contract file system (used by SMF) with 0 free blocks and 2147483582(INTMAX-1) free files. For object file systems, `/system/object` is the mount point, `objfs` is the object file system (see `objfs(7FS)`) with 0 free blocks and 2147483511 free files.

EXAMPLE 2 Writing Portable Information About the `/usr` File System

The following example writes portable information about the `/usr` file system:

```
example% /usr/bin/df -P /usr
```

EXAMPLE 3 Writing Portable Information About the `/usr/src` file System

Assuming that `/usr/src` is part of the `/usr` file system, the following example writes portable information :

```
example% /usr/bin/df -P /usr/src
```

EXAMPLE 4 Using `df` to Display Inode Usage

The following example displays inode usage on all `ufs` file systems:

```
example%/usr/bin/df -F ufs -o i
```

Environment Variables

When set, any header which normally displays `files` will now display `nodes`. See `environ(5)` for descriptions of the following environment variables that affect the execution of `df`: `LANG`, `LC_ALL`, `LC_CTYPE`, `LC_MESSAGES`, and `NLSPATH`.

Exit Status

The following exit values are returned:

`0`
: Successful completion.

`>0`
: An error occurred.

Files

`/dev/dsk/*`
: Disk devices

`/etc/default/fs`
: Default local file system type. Default values can be set for the following flags in `/etc/default/fs`. For example: `LOCAL=ufs`, where `LOCAL` is the default partition for a command if no FSType is specified.

`/etc/mnttab`
: Mount table

`/etc/vfstab`
: List of default parameters for each file system

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os
Interface Stability	Committed

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Standard	See standards(5).

See Also find(1), df_ufs(1M), mount(1M), statvfs(2), default_fs(4), mnttab(4), vfstab(4), attributes(5), environ(5), largefile(5), standards(5), objfs(7FS)

Notes If UFS logging is enabled on a file system, the disk space used for the log is reflected in the df report. The log is allocated from free blocks on the file system, and it is sized approximately 1 Mbyte per 1 Gbyte of file system, up to 256 Mbytes. The log size may be larger (up to a maximum of 512 Mbytes) depending on the number of cylinder groups present in the file system.

Name dfmounts – display mounted resource information

Synopsis

```
dfmounts [-F FSType] [-h] [-o specific_options]
     [restriction]...
```

Description `dfmounts` shows the local resources shared through a distributed file system *FSType* along with a list of clients that have the resource mounted. If *restriction* is not specified, `dfmounts` shows file systems that are currently shared on any NFS server. *specific_options* as well as the availability and semantics of *restriction* are specific to particular distributed file system types.

If `dfmounts` is entered without arguments, remote resources currently mounted on the local system are displayed, regardless of file system type. However, the `dfmounts` command does not display the names of NFS Version 4 clients.

`dfmounts` Output The output of `dfmounts` consists of an optional header line (suppressed with the `-h` flag) followed by a list of lines containing whitespace-separated fields. For each resource, the fields are:

resource server pathname clients ...

where:

resource Specifies the resource name that must be given to the `mount(1M)` command.

server Specifies the system from which the resource was mounted.

pathname Specifies the pathname that must be given to the `share(1M)` command.

clients Is a comma-separated list of systems that have mounted the resource. Clients are listed in the form *domain.*, *domain.system*, or *system*, depending on the file system type.

A field can be null. Each null field is indicated by a hyphen (–) unless the remainder of the fields on the line are also null, in which case the hyphen can be omitted.

Fields with whitespace are enclosed in quotation marks (`" "`).

Options

`-F` *FSType* Specify filesystem type. Defaults to the first entry in `/etc/dfs/fstypes`. `Note`: currently the only valid *FSType* is `nfs`.

`-h` Suppress header line in output.

`-o` *specific_options* Specify options specific to the filesystem provided by the `-F` option. `Note`: currently no options are supported.

Files

`/etc/dfs/fstypes` file system types

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also dfshares(1M), mount(1M), share(1M), unshare(1M), attributes(5)

Name dfmounts_nfs – display mounted NFS resource information

Synopsis

```
dfmounts [-F nfs] [-h] [server]...
```

Description `dfmounts` shows the local resources shared through NFS, along with the list of clients that have mounted the resource. The `-F` flag may be omitted if NFS is the only file system type listed in the file `/etc/dfs/fstypes`.

`dfmounts` without options, displays all remote resources mounted on the local system, regardless of file system type.

The output of `dfmounts` consists of an optional header line (suppressed with the `-h` flag) followed by a list of lines containing whitespace-separated fields. For each resource, the fields are:

resource server pathname clients ...

where

resource Does not apply to NFS. Printed as a hyphen (-).

server Specifies the system from which the resource was mounted.

pathname Specifies the pathname that must be given to the `share(1M)` command.

clients Is a comma-separated list of systems that have mounted the resource.

Options

`-F nfs` Specifies the `nfs`-FSType.

`-h` Suppress header line in output.

server Displays information about the resources mounted from each server, where *server* can be any system on the network. If no server is specified, the *server* is assumed to be the local system.

Files `/etc/dfs/fstypes`

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/file-system/nfs

See Also `mount(1M)`, `share(1M)`, `unshare(1M)`, `attributes(5)`

Name dfshares – list available resources from remote or local systems

Synopsis

```
dfshares [-F FSType] [-h] [-o specific_options] [server]...
```

Description `dfshares` provides information about resources available to the host through a distributed file system of type *FSType*. *specific_options* as well as the semantics of *server* are specific to particular distributed file systems.

If `dfshares` is entered without arguments, all resources currently shared on the local system are displayed, regardless of file system type.

The output of `dfshares` consists of an optional header line (suppressed with the `-h` flag) followed by a list of lines containing whitespace-separated fields. For each resource, the fields are:

resource server access transport

where

resource Specifies the resource name that must be given to the `mount(1M)` command.

server Specifies the name of the system that is making the resource available.

access Specifies the access permissions granted to the client systems, either `ro` (for read-only) or `rw` (for read/write). If `dfshares` cannot determine access permissions, a hyphen (–) is displayed.

transport Specifies the transport provider over which the resource is shared.

A field may be null. Each null field is indicated by a hyphen (–) unless the remainder of the fields on the line are also null; in which case, the hyphen may be omitted.

Options

`-F` *FSType* Specify filesystem type. Defaults to the first entry in `/etc/dfs/fstypes`.

`-h` Suppress header line in output.

`-o` *specific_options* Specify options specific to the filesystem provided by the `-F` option.

Files `/etc/dfs/fstypes`

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also `dfmounts(1M)`, `mount(1M)`, `share(1M)`, `unshare(1M)`, `attributes(5)`

Name dfshares_nfs – list available NFS resources from remote systems

Synopsis

```
dfshares [-F nfs] [-h] [server]...
```

Description `dfshares` provides information about resources available to the host through NFS. The `-F` flag may be omitted if NFS is the first file system type listed in the file `/etc/dfs/fstypes`.

The query may be restricted to the output of resources available from one or more servers.

`dfshares` without arguments displays all resources shared on the local system, regardless of file system type.

Specifying *server* displays information about the resources shared by each server. *Server* can be any system on the network. If no server is specified, then *server* is assumed to be the local system.

The output of `dfshares` consists of an optional header line (suppressed with the `-h` flag) followed by a list of lines containing whitespace-separated fields. For each resource, the fields are:

resource server access transport

where

resource Specifies the resource name that must be given to the `mount(1M)` command.

server Specifies the system that is making the resource available.

access Specifies the access permissions granted to the client systems; however, `dfshares` cannot determine this information for an NFS resource and populates the field with a hyphen (-).

transport Specifies the transport provider over which the *resource* is shared; however, `dfshares` cannot determine this information for an NFS resource and populates the field with a hyphen (-).

A field may be null. Each null field is indicated by a hyphen (-) unless the remainder of the fields on the line are also null; in which case, the hyphen may be omitted.

Options

`-F nfs` Specify the NFS file system type

`-h` Suppress header line in output.

Files `/etc/dfs/fstypes`

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/file-system/nfs

See Also mount(1M), share(1M), unshare(1M), attributes(5)

Name df_ufs – report free disk space on ufs file systems

Synopsis `df -F ufs [`*generic_options*`] [-o i] [`*directory* | *special*`]`

Description `df` displays the amount of disk space occupied by `ufs` file systems, the amount of used and available space, and how much of the file system's total capacity has been used.The amount of space reported as used and available is less than the amount of space in the file system; this is because the system reserves a fraction of the space in the file system to allow its file system allocation routines to work well. The amount reserved is typically about 10%; this can be adjusted using `tunefs(1M)`. When all the space on the file system except for this reserve is in use, only the superuser can allocate new files and data blocks to existing files. When the file system is overallocated in this way, `df` might report that the file system is more than 100% utilized.If neither *directory* nor *special* is specified, `df` displays information for all mounted ufs file systems.

Options The following options are supported:

generic_options Options supported by the generic `df` command. See `df(1M)` for a description of these options.

`-o` Specify `ufs` file system specific options. The available option is:

`i` Report the number of used and free inodes. This option can not be used with *generic_options*.

Files `/etc/mnttab` list of file systems currently mounted

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os, system/xopen/xcu4

See Also `df(1M)`, `fsck(1M)`, `fstyp(1M)`, `tunefs(1M)`, `mnttab(4)`, `attributes(5)`, `ufs(7FS)`,

Notes `df` calculates its results differently for mounted and unmounted file systems. For unmounted systems, the numbers reflect the 10% reservation. This reservation is not reflected in `df` output for mounted file systems. For this reason, the available space reported by the generic command can differ from the available space reported by this module.

`df` might report remaining capacity even though `syslog` warns `filesystem full`. This issue can occur because `df` only uses the available fragment count to calculate available space, but the file system requires contiguous sets of fragments for most allocations.

If you suspect that you have exhausted contiguous fragments on your file system, you can use the `fstyp(1M)` utility with the `-v` option. In the `fstyp` output, look at the `nbfree` (number of blocks free) and `nffree` (number of fragments free) fields. On unmounted filesystems, you can use `fsck(1M)` and observe the last line of output, which reports, among other items, the number of fragments and the degree of fragmentation. See `fsck(1M)`.

Name dhcpagent – Dynamic Host Configuration Protocol (DHCP) client daemon

Synopsis

```
dhcpagent [-a] [ -d n] [-f] [-v]
```

Description `dhcpagent` implements the client half of the Dynamic Host Configuration Protocol (DHCP) for machines running Solaris software.

The `dhcpagent` daemon obtains configuration parameters for the client (local) machine's network interfaces from a DHCP server. These parameters may include a lease on an IP address, which gives the client machine use of the address for the period of the lease, which may be infinite. If the client wishes to use the IP address for a period longer than the lease, it must negotiate an extension using DHCP. For this reason, `dhcpagent` must run as a daemon, terminating only when the client machine powers down.

For IPv4, the `dhcpagent` daemon is controlled through `ifconfig(1M)`.

`dhcpagent` can be invoked as a user process, albeit one requiring root privileges, but this is not necessary, as `ifconfig(1M)` will start it automatically.

For IPv6, the `dhcpagent` daemon is invoked automatically by `in.ndpd(1M)`. It can also be controlled through `ifconfig(1M)`, if necessary.

When invoked, `dhcpagent` enters a passive state while it awaits instructions from `ifconfig(1M)` or `in.ndpd(1M)`. When it receives a command to configure an interface, it brings up the interface (if necessary) and starts DHCP. Once DHCP is complete, `dhcpagent` can be queried for the values of the various network parameters. In addition, if DHCP was used to obtain a lease on an address for an interface, it configures the address for use. When a lease is obtained, it is automatically renewed as necessary. If the lease cannot be renewed, `dhcpagent` will unconfigure the address, but the interface will be left up and `dhcpagent` will attempt to acquire a new address lease. `dhcpagent` monitors system suspend/resume events and will validate any non-permanent leases with the DHCP server upon resume. Similarly, `dhcpagent` monitors link up/down events and will validate any non-permanent leases with the DHCP server when the downed link is brought back up. The lease validation mechanism will restart DHCP if the server indicates that the existing lease is no longer valid. If the server cannot be contacted, then the existing lease will continue. This behavior can be modified with the `VERIFIED_LEASE_ONLY` parameter in the `/etc/default/dhcpagent` file. See the description of this parameter below.

For IPv4, if the configured interface is found to be unplumbed, or to have a different IP address, subnet mask, or broadcast address from those obtained from DHCP, the interface is abandoned from DHCP control.

For IPv6, `dhcpagent` automatically plumbs and unplumbs logical interfaces as necessary for the IPv6 addresses supplied by the server. The IPv6 prefix length (netmask) is not set by the DHCPv6 protocol, but is instead set by `in.ndpd(1M)` using prefix information obtained by Router Advertisements. If any of the logical interfaces created by `dhcpagent` is unplumbed, or

configured with a different IP address, it will be abandoned from DHCP control. If the link-local interface is unplumbed, then all addresses configured by DHCP on that physical interface will be removed.

In addition to DHCP, `dhcpagent` also supports BOOTP (IPv4 only). See *RFC 951, Bootstrap Protocol*. Configuration parameters obtained from a BOOTP server are treated identically to those received from a DHCP server, except that the IP address received from a BOOTP server always has an infinite lease.

DHCP also acts as a mechanism to configure other information needed by the client, for example, the domain name and addresses of routers. Aside from the IP address, and for IPv4 alone, the netmask, broadcast address, and default router, the agent does not directly configure the workstation, but instead acts as a database which may be interrogated by other programs, and in particular by `dhcpinfo(1)`.

On clients with a single interface, this is quite straightforward. Clients with multiple interfaces may present difficulties, as it is possible that some information arriving on different interfaces may need to be merged, or may be inconsistent. Furthermore, the configuration of the interfaces is asynchronous, so requests may arrive while some or all of the interfaces are still unconfigured. To handle these cases, one interface may be designated as primary, which makes it the authoritative source for the values of DHCP parameters in the case where no specific interface is requested. See `dhcpinfo(1)` and `ifconfig(1M)` for details.

All DHCP packets sent by `dhcpagent` include a vendor class identifier (RFC 2132, option code 60; RFC 3315, option code 16). This identifier is the same as the platform name returned by the `uname -i` command, except:

- Any commas in the platform name are changed to periods.
- If the name does not start with a stock symbol and a comma, it is automatically prefixed with `SUNW`.

Messages

The `dhcpagent` daemon writes information and error messages in five categories:

critical
: Critical messages indicate severe conditions that prevent proper operation.

errors
: Error messages are important, sometimes unrecoverable events due to resource exhaustion and other unexpected failure of system calls; ignoring errors may lead to degraded functionality.

warnings
: Warnings indicate less severe problems, and in most cases, describe unusual or incorrect datagrams received from servers, or requests for service that cannot be provided.

informational
: Informational messages provide key pieces of information that can be useful to debugging a DHCP configuration at a site. Informational messages are generally controlled by the `-v` option. However, certain critical pieces of information, such as the IP address obtained, are always provided.

debug
: Debugging messages, which may be generated at two different levels of verbosity, are chiefly of benefit to persons having access to source code, but may be useful as well in debugging difficult DHCP configuration problems. Debugging messages are only generated when using the `-d` option.

When `dhcpagent` is run without the `-f` option, all messages are sent to the system logger `syslog(3C)` at the appropriate matching priority and with a facility identifier `LOG_DAEMON`. When `dhcpagent` is run with the `-f` option, all messages are directed to standard error.

DHCP Events and User-Defined Actions

If an executable (binary or script) is placed at `/etc/dhcp/eventhook`, the `dhcpagent` daemon will automatically run that program when any of the following events occur:

`BOUND` and `BOUND6`
: These events occur during interface configuration. The event program is invoked when `dhcpagent` receives the DHCPv4 ACK or DHCPv6 Reply message from the DHCP server for the lease request of an address, indicating successful initial configuration of the interface. (See also the `INFORM` and `INFORM6` events, which occur when configuration parameters are obtained without address leases.)

`EXTEND` and `EXTEND6`
: These events occur during lease extension. The event program is invoked just after `dhcpagent` receives the DHCPv4 ACK or DHCPv6 Reply from the DHCP server for the DHCPv4 REQUEST (renew) message or the DHCPv6 Renew or Rebind message.

  Note that with DHCPv6, the server might choose to remove some addresses, add new address leases, and ignore (allow to expire) still other addresses in a given Reply message. The `EXTEND6` event occurs when a Reply is received that leaves one or more address leases still valid, even if the Reply message does not extend the lease for any address. The event program is invoked just before any addresses are removed, but just after any new addresses are added. Those to be removed will be marked with the `IFF_DEPRECATED` flag.

`EXPIRE` and `EXPIRE6`
: These events occur during lease expiration. For DHCPv4, the event program is invoked just before the leased address is removed from an interface. For DHCPv6, the event program is invoked just before the last remaining leased addresses are removed from the interface.

`DROP` and `DROP6`
: These events occur during the period when an interface is dropped. The event program is invoked just before the interface is removed from DHCP control. If the interface has been

abandoned due the user unplumbing the interface, then this event will occur after the user's action has taken place. The interface might not be present.

`INFORM` and `INFORM6`
: These events occur when an interface acquires new or updated configuration information from a DHCP server by means of the DHCPv4 `INFORM` or the DHCPv6 Information-Request message. These messages are sent using an `ifconfig(1M)` `dhcp inform` command or when the DHCPv6 Router Advertisement `O` (letter 0) bit is set and the `M` bit is not set. Thus, these events occur when the DHCP client does not obtain an IP address lease from the server, and instead obtains only configuration parameters.

`LOSS6`
: This event occurs during lease expiration when one or more valid leases still remain. The event program is invoked just before expired addresses are removed. Those being removed will be marked with the `IFF_DEPRECATED` flag.

  Note that this event is not associated with the receipt of the Reply message, which occurs only when one or more valid leases remain, and occurs only with DHCPv6. If all leases have expired, then the EXPIRE6 event occurs instead.

`RELEASE` and `RELEASE6`
: This event occurs during the period when a leased address is released. The event program is invoked just before `dhcpagent` relinquishes the address on an interface and sends the DHCPv4 `RELEASE` or DHCPv6 Release packet to the DHCP server.

The system does not provide a default event program. The file `/etc/dhcp/eventhook` is expected to be owned by root and have a mode of 755.

The event program will be passed two arguments, the interface name and the event name, respectively. For DHCPv6, the interface name is the name of the physical interface.

The event program can use the `dhcpinfo(1)` utility to fetch additional information about the interface. While the event program is invoked on every event defined above, it can ignore those events in which it is not interested. The event program runs with the same privileges and environment as `dhcpagent` itself, except that `stdin`, `stdout`, and `stderr` are redirected to `/dev/null`. Note that this means that the event program runs with root privileges.

If an invocation of the event program does not exit after 55 seconds, it is sent a `SIGTERM` signal. If does not exit within the next three seconds, it is terminated by a `SIGKILL` signal.

See EXAMPLES for an example event program.

Options The following options are supported:

`-a`
: Adopt a configured IPv4 interface. This option is for use with diskless DHCP clients. In the case of diskless DHCP, DHCP has already been performed on the network interface

providing the operating system image prior to running `dhcpagent`. This option instructs the agent to take over control of the interface. It is intended primarily for use in boot scripts.

The effect of this option depends on whether the interface is being adopted.

If the interface is being adopted, the following conditions apply:

`dhcpagent` uses the client id specified in `/chosen`:<*client_id*>, as published by the PROM or as specified on a `boot`(1M) command line. If this value is not present, the client id is undefined. The DHCP server then determines what to use as a client id. It is an error condition if the interface is an Infiniband interface and the PROM value is not present.

If the interface is not being adopted:

`dhcpagent` uses the value stored in `/etc/default/dhcpagent`. If this value is not present, the client id is undefined. If the interface is Infiniband and there is no value in `/etc/default/dhcpagent`, a client id is generated as described by the draft document on DHCP over Infiniband, available at:

`http://www.ietf.org`

`-d` *n*
Set debug level to *n*. Two levels of debugging are currently available, 1 and 2; the latter is more verbose.

`-f`
Run in the foreground instead of as a daemon process. When this option is used, messages are sent to standard error instead of to `syslog`(3C).

`-v`
Provide verbose output useful for debugging site configuration problems.

Examples

EXAMPLE 1 Example Event Program

The following script is stored in the file `/etc/dhcp/eventhook`, owned by root with a mode of 755. It is invoked upon the occurrence of the events listed in the file.

```
#!/bin/sh

(
echo "Interface name: " $1
echo "Event: " $2

case $2 in
"BOUND")
     echo "Address acquired from server "\
         `/usr/sbin/dhcpinfo -i $1 ServerID`
     ;;
"BOUND6")
```

EXAMPLE 1 Example Event Program *(Continued)*

```
     echo "Addresses acquired from server " \
         `/usr/sbin/dhcpinfo -v6 -i $1 ServerID`
     ;;
"EXTEND")
    echo "Lease extended for " \
         `/usr/sbin/dhcpinfo -i $1 LeaseTime`" seconds"
     ;;
"EXTEND6")
    echo "New lease information obtained on $i"
     ;;
"EXPIRE" | "DROP" | "RELEASE")
     ;;

esac
) >/var/run/dhcp_eventhook_output 2>&1
```

Note the redirection of stdout and stderr to a file.

Files

`/etc/dhcp/`*if*`.dhc`
`/etc/dhcp/`*if*`.dh6`
: Contains the configuration for interface. The mere existence of this file does not imply that the configuration is correct, since the lease might have expired. On start-up, `dhcpagent` confirms the validity of the address using REQUEST (for DHCPv4) or Confirm (DHCPv6).

`/etc/dhcp/duid`
`/etc/dhcp/iaid`
: Contains persistent storage for DUID (DHCP Unique Identifier) and IAID (Identity Association Identifier) values. The format of these files is undocumented, and applications should not read from or write to them.

`/etc/default/dhcpagent`
: Contains default values for tunable parameters. All values may be qualified with the interface they apply to by prepending the interface name and a period (".") to the interface parameter name.

  To configure IPv6 parameters, place the string `.v6` between the interface name (if any) and the parameter name. For example, to set the global IPv6 parameter request list, use `.v6.PARAM_REQUEST_LIST`. To set the `CLIENT_ID` (`DUID`) on `hme0`, use `hme0.v6.CLIENT_ID`.

  The parameters include:

  `VERIFIED_LEASE_ONLY`
  : Indicates that a `RELEASE` rather than a `DROP` should be performed on managed interfaces when the agent terminates. Release causes the client to discard the lease, and the server to make the address available again. Drop causes the client to record the lease in

`/etc/dhcp/`*interface*`.dhc` or `/etc/dhcp/`*interface*`.dh6` for later use. In addition, when the link status changes to `up` or when the system is resumed after a suspend, the client will verify the lease with the server. If the server is unreachable for verification, then the old lease will be discarded (even if it has time remaining) and a new one obtained.

Enabling this option is often desirable on mobile systems, such as laptops, to allow the system to recover quickly from moves.

`OFFER_WAIT`
Indicates how long to wait between checking for valid `OFFER`s after sending a `DISCOVER`. For DHCPv6, sets the time to wait between checking for valid Advertisements after sending a Solicit.

`CLIENT_ID`
Indicates the value that should be used to uniquely identify the client to the server. This value can take one of three basic forms:

decimal`,`*data*`...`
`0xHHHHH...`
"*string*`...."`

The first form is an RFC 3315 DUID. This is legal for both IPv4 DHCP and DHCPv6. For IPv4, an RFC 4361 Client ID is constructed from this value. In this first form, the format of *data*... depends on the decimal value. The following formats are defined for this first form:

1,*hwtype*,*time*,*lla*
Type 1, DUID-LLT. The *hwtype* value is an integer in the range 0-65535, and indicates the type of hardware. The *time* value is the number of seconds since midnight, January 1st, 2000 UTC, and can be omitted to use the current system time. The *lla* value is either a colon-separated MAC address or the name of a physical interface. If the name of an interface is used, the *hwtype* value can be omitted. For example: `1,,,hme0`

2,*enterprise*,*hex*...
Type 2, DUID-EN. The *enterprise* value is an integer in the range 0-4294967295 and represents the SMI Enterprise number for an organization. The *hex* string is an even-length sequence of hexadecimal digits.

3,*hwtype*,*lla*
Type 3, DUID-LL. This is the same as DUID-LLT (type 1), except that a time stamp is not used.

*,*hex*
Any other type value (0 or 4-65535) can be used with an even-length hexadecimal string.

The second and third forms of `CLIENT_ID` are legal for IPv4 only. These both represent raw Client ID (without RFC 4361), in hex, or NVT ASCII string format. Thus, "`Sun`" and `0x53756E` are equivalent.

`PARAM_REQUEST_LIST`
Specifies a list of comma-separated integer values of options for which the client would like values, or symbolic `Site` or `Option` option names. Symbolic option names for IPv4 are resolved through `/etc/dhcp/inittab`. Option names for IPv6 are resolved by means of `/etc/dhcp/inittab6`.

`PARAM_IGNORE_LIST`
Specifies a list of options (constructed in the same manner as `PARAM_REQUEST_LIST`) that the DHCP client will ignore. Ignored options are treated as though the server did not return the options specified. Ignored options are not visible using `dhcpinfo(1)` or acted on by the client. This parameter can be used, for example, to disable an unwanted client name or default router.

`/etc/dhcp/eventhook`
Location of a DHCP event program.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os
Interface Stability	Committed

See Also `dhcpinfo(1)`, `ifconfig(1M)`, `init(1M)`, `in.mpathd(1M)`, `in.ndpd(1M)`, `syslog(3C)`, `attributes(5)`, `dhcp(5)`

System Administration Guide: IP Services

Croft, B. and Gilmore, J.,*Bootstrap Protocol (BOOTP)*RFC 951, Network Working Group, September 1985.

Droms, R., *Dynamic Host Configuration Protocol*, RFC 2131, Network Working Group, March 1997.

Lemon, T. and B. Sommerfeld. *RFC 4361, Node-specific Client Identifiers for Dynamic Host Configuration Protocol Version Four (DHCPv4)*. Nominum and Sun Microsystems. February 2006.

Droms, R. *RFC 3315, Dynamic Host Configuration Protocol for IPv6 (DHCPv6)*. Cisco Systems. July 2003.

Notes The `dhcpagent` daemon can be used on IPv4 logical interfaces, just as with physical interfaces. When used on a logical interface, the daemon automatically constructs a Client ID value based on the DUID and IAID values, according to RFC 4361. The `/etc/default/dhcpagent` `CLIENT_ID` value, if any, overrides this automatic identifier.

Unlike physical IPv4 interfaces, `dhcpagent` does not add or remove default routes associated with logical interfaces.

DHCP can be performed on IPMP IP interfaces to acquire and maintain IPMP data addresses. Because an IPMP IP interface has no hardware address, the daemon automatically constructs a Client ID using the same approach described above for IPv4 logical interfaces. In addition, the lack of a hardware address means the daemon must set the "broadcast" flag in all `DISCOVER` and `REQUEST` messages on IPMP IP interfaces. Some DHCP servers may refuse such requests.

DHCP can be performed on IP interfaces that are part of an IPMP group (to acquire and maintain test addresses). The daemon will automatically set the `NOFAILOVER` and `DEPRECATED` flags on each test address. Additionally, the daemon will not add or remove default routes in this case. Note that the actual DHCP packet exchange may be performed over any active IP interface in the IPMP group. It is strongly recommended that test addresses have infinite leases. Otherwise, an extended network outage detectable only by probes may cause test address leases to expire, causing `in.mpathd(1M)` to revert to link-based failure detection and trigger an erroneous repair.

Name dhcpconfig – DHCP service configuration utility

Synopsis

```
dhcpconfig -D -r resource -p path [-u uninterpreted]
     [-l lease_length] [-n ] [-d DNS_domain]
     [-a DNS_server_addresses] [-h hosts_resource]
     [-y hosts_domain]

dhcpconfig -R server_addresses

dhcpconfig -U [-f] [-x] [-h]

dhcpconfig -N network_address [-m subnet_mask] [-b ]
     [-t router_addresses] [-y NIS-domain]
     [-a NIS_server_addresses] [-g]

dhcpconfig -C -r resource -p path [-f] [-k]
     [-u uninterpreted]

dhcpconfig -X filename [-m macro_list] [-o option_list]
     [-a network_addresses] [-f] [-x] [-g]

dhcpconfig -I filename [-f] [-g]

dhcpconfig -P [parameter[=value],]...

dhcpconfig -S [-f] [-e | -d | -r | -q]
```

Description The `dhcpconfig` command is used to configure and manage the Dynamic Host Configuration Protocol (DHCP) service or BOOTP relay services. It is intended for use by experienced Solaris system administrators and is designed for ease of use in scripts. The `dhcpmgr` utility is recommended for less experienced administrators or those preferring a graphical utility to configure and manage the DHCP service or BOOTP relay service.

The `dhcpconfig` command can be run by root, or by other users assigned to the DHCP Management profile. See `rbac(5)` and `user_attr(4)`.

`dhcpconfig` requires one of the following function flags: `-D`, `-R`, `-U`, `-N`, `-C`, `-X`, `-I`, `-P` or `-S`.

The `dhcpconfig` menu driven mode is supported in Solaris 8 and previous versions of Solaris.

Where dhcpconfig Obtains Configuration Information

`dhcpconfig` scans various configuration files on your Solaris machine for information it can use to assign values to options contained in macros it adds to the `dhcptab` configuration table. The following table lists information `dhcpconfig` needs, the source used, and how the information is used:

Information	*Source*	*Where Used*
Timezone	System date, timezone settings	`Locale` macro
DNS parameters	`nsswitch.conf`, `/etc/resolv.conf`	Server macro

NIS parameters	System domainname, `nsswitch.conf`, NIS	Network macros
Subnetmask	Network interface, `netmasks` table in nameservice	Network macros

If you have not set these parameters on your server machine, you should do so before configuring the DHCP server with `dhcpconfig`. Note that if you specify options with the `dhcpconfig -D` command line, the values you supply override the values obtained from the system files.

The `dhcpconfig` utility is obsolete and is subject to removal in a future release of Oracle Solaris.

Options The following options are supported:

`-C`
Convert to using a new data store, recreating the DHCP data tables in a format appropriate to the new data store, and setting up the DHCP server to use the new data store.

The following sub-options are required:

`-p` *path_to_data*
The paths for `SUNWfiles` and `SUNWbinfiles` must be absolute UNIX pathnames. See `dhcp_modules(5)`.

`-r` *data_resource*
New data store resource. One of the following must be specified: `SUNWfiles` or `SUNWbinfiles`. See `dhcp_modules(5)`.

The following sub-options are optional:

`-f`
Do not prompt for confirmation. If `-f` is not used, a warning and confirmation prompt are issued before the conversion starts.

`-k`
Keep the old DHCP data tables after successful conversion. If any problem occurs during conversion, tables are not deleted even if `-k` sub-option is not specified.

`-u` *uninterpreted*
Data which is ignored by `dhcpconfig`, but passed on to the datastore for interpretation. The private layer provides for module-specific configuration information through the use of the `RESOURCE_CONFIG` keyword. Uninterpreted data is stored within `RESOURCE_CONFIG` keyword of `dhcpsvc.conf(4)`. The `-u` sub-option is not used with the `SUNWfiles` and `SUNWbinfiles` data stores. See `dhcp_modules(5)`.

`-D`
Configure the DHCP service.

The following sub-options are required:

`-r` *data_resource*
One of the following must be specified: `SUNWfiles` or `SUNWbinfiles`. Other data stores may be available. See `dhcp_modules(5)`.

`-p` *path*
The paths for `SUNWfiles` and `SUNWbinfiles` must be absolute UNIX pathnames. See `dhcp_modules(5)`.

The following sub-options are optional:

`-a` *DNS_servers*
IP addresses of DNS servers, separated with commas.

`-d` *DNS_domain*
DNS domain name.

`-h` *hosts_resource*
Resource in which to place hosts data. Usually, the name service in use on the server. Valid values are `files` or `dns`.

`-l` *seconds*
Lease length used for addresses not having a specified lease length, in seconds.

`-n`
Non-negotiable leases

`-y` *hosts_domain*
DNS domain name to be used for hosts data. Valid only if `dns` is specified for `-h` sub-option.

`-u` *uninterpreted*
Data which is ignored by `dhcpconfig`, but passed on to the datastore for interpretation. The private layer provides for module-specific configuration information through the use of the `RESOURCE_CONFIG` keyword. Uninterpreted data is stored within `RESOURCE_CONFIG` keyword of `dhcpsvc.conf(4)`. The `-u` sub-option is not used with the `SUNWfiles` and `SUNWbinfiles` data stores. See `dhcp_modules(5)`.

`-I` *filename*
Import data from *filename*, containing data previously exported from a Solaris DHCP server. Note that after importing, you may have to edit macros to specify the correct domain names, and edit network tables to change the owning server of addresses in imported networks. Use `dhtadm` and `pntadm` to do this.

The following sub-options are supported:

`-f`
Replace any conflicting data with the data being imported.

`-g`
Signal the daemon to reload the `dhcptab` once the import has been completed.

`-N` *net_address*
Configure an additional network for DHCP service.

The following sub-options are supported:

`-a` *NIS_server_addresses*
List of IP addresses of NIS servers.

`-b`
Network is a point-to-point (PPP) network, therefore no broadcast address should be configured. If `-b` is not used, the network is assumed to be a LAN, and the broadcast address is determined using the network address and subnet mask.

`-g`
Signal the daemon to reload the `dhcptab`.

`-m` *xxx.xxx.xxx.xxx*
Subnet mask for the network; if `-m` is not used, subnet mask is obtained from netmasks.

`-t` *router_addresses*
List of router IP addresses; if not specified, router discovery flag is set.

`-y` *NIS_domain_name*
If NIS is used on this network, specify the NIS domain name.

`-P`
Configure the DHCP service parameters. Each parameter and value are specified by the following pattern:

parameter`[=`*value*`],...`

Where parameter and value are:

parameter
One of the DHCP service parameters listed in `dhcpsvc.conf(4)`. If the corresponding *value* is not specified, the current parameter value is displayed. If *parameter* is not specified, all parameters and current values are displayed.

value
Optional string to set the servers parameter to if the value is acceptable. If the value is missing or is empty (`""`), the parameter and its current value are deleted.

After a parameter has changed the DHCP server requires re-starting before you can use new parameter values.

`-R` *server_addresses*
Configure the BOOTP relay service. BOOTP or DHCP requests are forwarded to the list of servers specified.

server_addresses is a comma separated list of hostnames and/or IP addresses.

`-S`
Control the DHCP service.

The following sub-options are supported:

`-d`
Disable and stop the DHCP service.

`-e`
Enable and start the DHCP service.

`-q`
Display the state of the DHCP service. The state is encoded into the exit status.

```
0    DHCP service disabled and stopped
1    DHCP service enabled and stopped
2    DHCP service disabled and running
3    DHCP service enabled and running
```

`-r`
Enable and restart the DHCP service.

`-U`
Unconfigure the DHCP service or BOOTP relay service.

The following sub-options are supported:

`-f`
Do not prompt for confirmation. If `-f` is not used, a warning and confirmation prompt is issued.

`-h`
Delete hosts entries from name service.

`-x`
Delete the `dhcptab` and network tables.

`-X` *filename*
Export data from the DHCP data tables, saving to *filename*, to move the data to another Solaris DHCP server.

The following sub-options are optional:

`-a` *networks_to_export*
List of networks whose addresses should be exported, or the keyword ALL to specify all networks. If `-a` is not specified, no networks are exported.

`-g`
Signal the daemon to reload the `dhcptab` after the export has been completed.

`-m` *macros_to_export*
List of macros to export, or the keyword `ALL` to specify all macros. If `-m` is not specified, no macros are exported.

`-o` *options_to_export*
List of options to export, or the keyword `ALL` to specify all options. If `-o` is not specified, no options are exported.

`-x`
Delete the data from this server after it is exported. If `-x` is not specified you are in effect copying the data.

Examples

EXAMPLE 1 Configuring DHCP Service with Binary Files Data Store

The following command configures DHCP service, using the binary files data store, in the DNS domain `acme.eng`, with a lease time of 28800 seconds (8 hours),

```
example# dhcpconfig -D -r SUNWbinfiles -p /var/dhcp -l 28800\
            -d acme.eng -a 120.30.33.4 -h dns -y acme.eng
```

EXAMPLE 2 Configuring BOOTP Relay Agent

The following command configures the DHCP daemon as a BOOTP relay agent, which forwards BOOTP and DHCP requests to the servers having the IP addresses 120.30.33.7 and 120.30.42.132:

```
example# dhcpconfig -R 120.30.33.7,120.30.42.132
```

EXAMPLE 3 Unconfiguring DHCP Service

The following command unconfigures the DHCP service, with confirmation, and deletes the DHCP data tables and host table entries:

```
example# dhcpconfig -U -x -h
```

EXAMPLE 4 Configuring a Network for DHCP Service

The following command configures an additional LAN network for DHCP service, specifying that clients should use router discovery and providing the NIS domain name and NIS server address:

```
example# dhcpconfig -N 120.30.171.0 -y east.acme.eng.com\
            -a 120.30.33.4
```

EXAMPLE 5 Exporting a Network, Macros, and Options from a DHCP Server

The following command exports one network (`120.30.171.0`) and its addresses, the macro `120.30.171.0`, and the options `motd` and `PSptr`from a DHCP server, saves the exported data in file `/export/var/120301710_data`, and deletes the exported data from the server.

```
example# dhcpconfig -X /var/dhcp/120301710_export
     -a 120.30.171.0 -m 120.30.171.0 -o motd,PSptr
```

EXAMPLE 6 Importing Data on a DHCP Server

The following command imports DHCP data from a file, `/net/golduck/export/var/120301710_data`, containing data previously exported from a Solaris DHCP server, overwrites any conflicting data on the importing server, and signals the daemon to reload the `dhcptab` once the import is completed:

```
example# dhcpconfig -I /net/golduck/export/var/120301710_data -f -g
```

EXAMPLE 7 Setting DHCP Server Parameters

The following command sets the number of minutes that the DHCP server waits before timing out when updating DNS information on DHCP clients to five minutes.

```
example# example# dhcpconfig -P UPDATE_TIMEOUT=5
```

EXAMPLE 8 Re-starting the DHCP server

The following command stops and re-starts the DHCP server.

```
example#  example# dhcpconfig -S -r
DHCP server stopped
DHCP server started
```

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	service/network/dhcp
Interface Stability	Obsolete

See Also

`dhcpmgr(1M)`, `dhtadm(1M)`, `in.dhcpd(1M)`, `pntadm(1M)`, `dhcp_network(4)`, `dhcptab(4)`, `dhcpsvc.conf(4)`, `nsswitch.conf(4)`, `resolv.conf(4)`, `user_attr(4)`, `attributes(5)`, `dhcp(5)`, `dhcp_modules(5)`, `rbac(5)`

System Administration Guide: IP Services

Name dhcpmgr – graphical interface for managing DHCP service

Synopsis

```
/usr/sadm/admin/bin/dhcpmgr
```

Description `dhcpmgr` is a graphical user interface which enables you to manage the Dynamic Host Configuration Protocol (DHCP) service on the local system. It performs the functions of the `dhcpconfig`, `dhtadm`, and `pntadm` command line utilities. You must be `root` to use `dhcpmgr`. The `dhcpmgr` Help, available from the `Help` menu, contains detailed information about using the tool.

The `dhcpmgr` utility is obsolete and is subject to removal in a future release of Oracle Solaris.

Usage You can perform the following tasks using `dhcpmgr`:

Configure DHCP service	Use `dhcpmgr` to configure the DHCP daemon as a DHCP server, and select the data store to use for storing network configuration tables..
Configure BOOTP relay service	Use `dhcpmgr` to configure the DHCP daemon as a BOOTP relay.
Manage DHCP or BOOTP relay service	Use `dhcpmgr` to start, stop, enable, disable or unconfigure the DHCP service or BOOTP relay service, or change DHCP server parameters.
Manage DHCP addresses	Use `dhcpmgr` to add, modify, or delete IP addresses leased by the DHCP service.
Manage DHCP macros	Use `dhcpmgr` to add, modify or delete macros used to supply configuration parameters to DHCP clients.
Manage DHCP options	Use `dhcpmgr` to add, modify or delete options used to define parameters deliverable through DHCP.
Convert to a new DHCP data store	Use `dhcpmgr` to configure the DHCP server to use a different data store, and convert the DHCP data to the format used by the new data store.
Move DHCP data to another server	Use `dhcpmgr` to export data from one Solaris DHCP server and import data onto another Solaris DHCP server.

Exit Status The following exit values are returned:

`0`	Successful completion.
`non-zero`	An error occurred.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	network/dhcp/dhcpmgr
Interface Stability	Obsolete

See Also dhcpconfig(1M), dhtadm(1M), pntadm(1M), in.dhcpd(1M), dhcpsvc.conf(4), dhcp_network(4), dhcptab(4), attributes(5), dhcp(5), dhcp_modules(5)

Oracle Solaris DHCP Service Developer's Guide

System Administration Guide: IP Services

Name dhtadm – DHCP configuration table management utility

Synopsis

```
dhtadm -C [-r resource] [-p path] [-u uninterpreted] [-g]

dhtadm -A -s symbol_name -d definition [-r resource]
     [-p path] [-u uninterpreted] [-g]

dhtadm -A -m macro_name -d definition [-r resource]
     [-p path] [-u uninterpreted] [-g]

dhtadm -M -s symbol_name -d definition [-r resource]
     [-p path] [-u uninterpreted] [-g]

dhtadm -M -s symbol_name -n new_name [-r resource]
     [-p path] [-u uninterpreted] [-g]

dhtadm -M -m macro_name -n new_name [-r resource] [-p path]
     [-u uninterpreted] [-g]

dhtadm -M -m macro_name -d definition [-r resource]
     [-p path] [-u uninterpreted] [-g]

dhtadm -M -m macro_name -e symbol=value [-r resource]
     [-p path] [-u uninterpreted] [-g]

dhtadm -D -s symbol_name [-r resource] [-p path]
     [-u uninterpreted] [-g]

dhtadm -D -m macro_name [-r resource] [-p path]
     [-u uninterpreted] [-g]

dhtadm -P [-r resource] [-p path] [-u uninterpreted] [-g]

dhtadm -R [-r resource] [-p path] [-u uninterpreted] [-g]

dhtadm -B [-v] [batchfile] [-g]
```

Description `dhtadm` manages the Dynamic Host Configuration Protocol (DHCP) service configuration table, `dhcptab`. You can use it to add, delete, or modify DHCP configuration macros or options or view the table. For a description of the table format, see `dhcptab(4)`.)

The `dhtadm` command can be run by root, or by other users assigned to the DHCP Management profile. See `rbac(5)` and `user_attr(4)`.

After you make changes with `dhtadm`, you should issue a `SIGHUP` to the DHCP server, causing it to read the `dhcptab` and pick up the changes. Do this using the `-g` option.

The `dhtadm` utility is obsolete and is subject to removal in a future release of Oracle Solaris.

Options One of the following function flags must be specified with the `dhtadm` command: `-A`, `-B`, `-C`, `-D`, `-M`, `-P` or `-R`.

The following options are supported:

`-A`
Add a symbol or macro definition to the `dhcptab` table.

The following sub-options are required:

`-d` *definition*
Specify a macro or symbol definition.

definition must be enclosed in single quotation marks. For macros, use the form `-d ':`*symbol*`=`*value*`:`*symbol*`=`*value*`:'`. Enclose a *value* that contains colons in double quotation marks. For symbols, the definition is a series of fields that define a symbol's characteristics. The fields are separated by commas. Use the form `-d '`*context*`,`*code*`,`*type*`,`*granularity*`,`*maximum*`'`. See `dhcptab(4)` for information about these fields.

`-m` *macro_name*
Specify the name of the macro to be added.

The `-d` option must be used with the `-m` option. The `-s` option cannot be used with the `-m` option.

`-s` *symbol_name*
Specify the name of the symbol to be added.

The `-d` option must be used with the `-s` option. The `-m` option cannot be used with the `-s` option.

`-B`
Batch process `dhtadm` commands. `dhtadm` reads from the specified file or from standard input a series of `dhtadm` commands and execute them within the same process. Processing many `dhtadm` commands using this method is much faster than running an executable batchfile itself. Batch mode is recommended for using `dhtadm` in scripts.

The following sub-option is optional:

`-v`
Display commands to standard output as they are processed.

`-C`
Create the DHCP service configuration table, `dhcptab`.

`-D`
Delete a symbol or macro definition.

The following sub-options are required:

`-m` *macro_name*
Delete the specified macro.

`-s` *symbol_name*
Delete the specified symbol.

`-g`
Signal the DHCP daemon to reload the `dhcptab` after successful completion of the operation.

`-M`
Modify an existing symbol or macro definition.

The following sub-options are required:

`-d` *definition*
Specify a macro or symbol definition to modify.

The definition must be enclosed in single quotation marks. For macros, use the form `-d ':`*symbol*`=`*value*`:`*symbol*`=`*value*`:'`. Enclose a *value* that contains colons in double quotation marks. For symbols, the definition is a series of fields that define a symbol's characteristics. The fields are separated by commas. Use the form `-d '`*context*`,`*code*`,`*type*`,`*granularity*`,`*maximum*`'`. See `dhcptab(4)` for information about these fields.

`-e`
This sub-option uses the *symbol*`=`*value* argument. Use it to edit a *symbol*/*value* pair within a macro. To add a symbol which does not have an associate value, enter:

symbol`=_NULL_VALUE_`

To delete a symbol definition from a macro, enter:

symbol`=`

`-m`
This sub-option uses the *macro_name* argument. The `-n`, `-d`, or `-e` sub-options are legal companions for this sub-option..

`-n`
This sub-option uses the *new_name* argument and modifies the name of the object specified by the `-m` or `-s` sub-option. It is not limited to macros. . Use it to specify a new macro name or symbol name.

`-s`
This sub-option uses the *symbol_name* argument. Use it to specify a symbol. The `-d` sub-option is a legal companion.

`-p` *path*
Override the `dhcpsvc.conf(4)` configuration value for `PATH=` with *path*. See `dhcpsvc.conf(4)` for more details regarding *path*. See `dhcp_modules(5)` for information regarding data storage modules for the DHCP service.

`-P`
Print (display) the `dhcptab` table.

`-r` *data_store_resource*
Override the `dhcpsvc.conf`(4) configuration value for `RESOURCE=` with the *data_store_resource* specified. See `dhcpsvc.conf(4)` for more details on resource type. See `dhcp_modules(5)` for information regarding data storage modules for the DHCP service.

`-R`
Remove the `dhcptab` table.

`-u` *uninterpreted*
Data which is ignored by `dhtadm`, but passed to currently configured public module, to be interpreted by the data store. The private layer provides for module-specific configuration information through the use of the `RESOURCE_CONFIG` keyword. Uninterpreted data is stored within `RESOURCE_CONFIG` keyword of `dhcpsvc.conf(4)`. See `dhcp_modules(5)` for information regarding data storage modules for the DHCP service.

Examples

EXAMPLE 1 Creating the DHCP Service Configuration Table

The following command creates the DHCP service configuration table, `dhcptab`:

```
# dhtadm -C
```

EXAMPLE 2 Adding a Symbol Definition

The following command adds a `Vendor` option symbol definition for a new symbol called `MySym` to the `dhcptab` table in the `SUNWfiles` resource in the `/var/mydhcp` directory:

```
# dhtadm -A -s MySym
   -d 'Vendor=SUNW.PCW.LAN,20,IP,1,0'
   -r SUNWfiles -p /var/mydhcp
```

EXAMPLE 3 Adding a Macro Definition

The following command adds the `aruba` macro definition to the `dhcptab` table. Note that symbol/value pairs are bracketed with colons (`:`).

```
# dhtadm -A -m aruba \
     -d ':Timeserv=10.0.0.10 10.0.0.11:DNSserv=10.0.0.1:'
```

EXAMPLE 4 Modifying a Macro Definition

The following command modifies the `Locale` macro definition, setting the value of the `UTCOffst` symbol to 18000 seconds. Note that any macro definition which includes the definition of the `Locale` macro inherits this change.

```
# dhtadm -M -m Locale -e 'UTCOffst=18000'
```

EXAMPLE 5 Deleting a Symbol

The following command deletes the `Timeserv` symbol from the `aruba` macro. Any macro definition which includes the definition of the `aruba` macro inherits this change.

```
# dhtadm -M -m aruba -e 'Timeserv='
```

EXAMPLE 6 Adding a Symbol to a Macro

The following command adds the `Hostname` symbol to the `aruba` macro. Note that the `Hostname` symbol takes no value, and thus requires the special value `_NULL_VALUE_`. Note also that any macro definition which includes the definition of the `aruba` macro inherits this change.

```
# dhtadm -M -m aruba -e 'Hostname=_NULL_VALUE_'
```

EXAMPLE 7 Renaming a Macro

The following command renames the `Locale` macro to `MyLocale`. Note that any `Include` statements in macro definitions which include the `Locale` macro also need to be changed.

```
# dhtadm -M -m Locale -n MyLocale
```

EXAMPLE 8 Deleting a Symbol Definition

The following command deletes the `MySym` symbol definition. Note that any macro definitions which use `MySym` needs to be modified.

```
# dhtadm -D -s MySym
```

EXAMPLE 9 Printing a dhcptab

The following command prints to standard output the contents of the `dhcptab` that is located in the data store and path indicated in the `dhcpsvc.conf` file:.

```
# dhtadm -P
```

EXAMPLE 10 Executing dhtadm in Batch Mode

The following command runs a series of `dhtadm` commands contained in a batch file and signals the daemon to reload the `dhcptab` once the commands have been executed: :

```
# dhtadm -B addmacros -g
```

Exit Status

`0`
Successful completion.

`1`
Object already exists.

`2`
Object does not exist.

`3`
Non-critical error.

`4`
Critical error.

Files `/etc/inet/dhcpsvc.conf`
contains service configuration parameters for the DHCP service

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	service/network/dhcp
Interface Stability	Obsolete

See Also `dhcpconfig(1M)`, `dhcpmgr(1M)`, `in.dhcpd(1M)`, `dhcpsvc.conf(4)`, `dhcp_network(4)`, `dhcptab(4)`, `hosts(4)`, `user_attr(4)`, `attributes(5)`, `dhcp(5)`, `dhcp_modules(5)rbac(5)`

Oracle Solaris DHCP Service Developer's Guide

System Administration Guide: IP Services

Alexander, S., and R. Droms, *DHCP Options and BOOTP Vendor Extensions*, RFC 1533, Lachman Technology, Inc., Bucknell University, October 1993.

Droms, R., *Interoperation Between DHCP and BOOTP*, RFC 1534, Bucknell University, October 1993.

Droms, R., *Dynamic Host Configuration Protocol*, RFC 1541, Bucknell University, October 1993.

Wimer, W., *Clarifications and Extensions for the Bootstrap Protocol*, RFC 1542, Carnegie Mellon University, October 1993.

Name dig – DNS lookup utility

Synopsis

```
dig [@server] [-b address] [-c class] [-f filename]
     [-k filename] [-m] [-p port#] [-q name] [-t type] [-x addr]
     [-y [hmac:]name:key] [-4] [-6] [name] [type] [class] [queryopt]...

dig [-h]

dig [global-queryopt...] [query...]
```

Description The `dig` utility (domain information groper) is a flexible tool for interrogating DNS name servers. It performs DNS lookups and displays the answers that are returned from the name server(s) that were queried. Most DNS administrators use `dig` to troubleshoot DNS problems because of its flexibility, ease of use and clarity of output. Other lookup tools tend to have less functionality than `dig`.

Although `dig` is normally used with command-line arguments, it also has a batch mode of operation for reading lookup requests from a file. A brief summary of its command-line arguments and options is printed when the `-h` option is specified. Unlike earlier versions, the BIND 9 implementation of `dig` allows multiple lookups to be issued from the command line.

Unless it is told to query a specific name server, `dig` tries each of the servers listed in `/etc/resolv.conf`.

When no command line arguments or options are given, `dig` performs an NS query for "." (the root).

It is possible to set per-user defaults for `dig` with `${HOME}/.digrc`. This file is read and any options in it are applied before the command line arguments.

The `IN` and `CH` class names overlap with the `IN` and `CH` top level domains names. Either use the `-t` and `-c` options to specify the type and class, or use "`IN.`" and "`CH.`" when looking up these top level domains.

Simple Usage The following is a typical invocation of `dig`:

```
dig @server name type
```

where:

server
: The name or IP address of the name server to query. This can be an IPv4 address in dotted-decimal notation or an IPv6 address in colon-delimited notation. When the supplied *server* argument is a hostname, `dig` resolves that name before querying that name server. If no *server* argument is provided, `dig` consults `/etc/resolv.conf` and queries the name servers listed there. The reply from the name server that responds is displayed.

name
: The name of the resource record that is to be looked up.

type
Indicates what type of query is required (ANY, A, MX, SIG, among others.) *type* can be any valid query type. If no *type* argument is supplied, `dig` performs a lookup for an A record.

Options The following options are supported:

`-4`
Use only IPv4 transport. By default both IPv4 and IPv6 transports can be used. Options `-4` and `-6` are mutually exclusive.

`-6`
Use only IPv6 transport. By default both IPv4 and IPv6 transports can be used. Options `-4` and `-6` are mutually exclusive.

`-b` *address*
Set the source IP address of the query to *address*. This must be a valid address on one of the host's network interfaces or `0.0.0.0` or `::`. An optional port may be specified by appending: `#`<*port*>

`-c` *class*
Override the default query class (IN for internet). The *class* argument is any valid class, such as HS for Hesiod records or CH for CHAOSNET records.

`-f` *filename*
Operate in batch mode by reading a list of lookup requests to process from the file *filename*. The file contains a number of queries, one per line. Each entry in the file should be organized in the same way they would be presented as queries to `dig` using the command-line interface.

`-h`
Print a brief summary of command-line arguments and options.

`-k` *filename*
Specify a transaction signature (TSIG) key file to sign the DNS queries sent by `dig` and their responses using TSIGs.

`-m`
Enable memory usage debugging.

`-p` *port#*
Query a non-standard port number. The *port#* argument is the port number that `dig` sends its queries instead of the standard DNS port number 53. This option tests a name server that has been configured to listen for queries on a non-standard port number.

`-q` *name*
Sets the query name to *name*. This can be useful in that the query name can be easily distinguished from other arguments.

`-t` *type*
Set the query type to *type*, which can be any valid query type supported in BIND9. The default query type "A", unless the `-x` option is supplied to indicate a reverse lookup. A zone transfer can be requested by specifying a type of AXFR. When an incremental zone transfer (IXFR) is required, *type* is set to `ixfr=`*N*. The incremental zone transfer will contain the changes made to the zone since the serial number in the zone's SOA record was *N*.

`-x` *addr*
Simplify reverse lookups (mapping addresses to names). The *addr* argument is an IPv4 address in dotted-decimal notation, or a colon-delimited IPv6 address. When this option is used, there is no need to provide the *name*, *class* and *type* arguments. The `dig` utility automatically performs a lookup for a name like `11.12.13.10.in-addr.arpa` and sets the query type and class to PTR and IN, respectively. By default, IPv6 addresses are looked up using nibble format under the IP6.ARPA domain. To use the older RFC1886 method using the IP6.INT domain, specify the `-i` option. Bit string labels (RFC 2874) are now experimental and are not attempted.

`-y` [*hmac*:]*name*:*key*
Specify a transaction signature (TSIG) key on the command line. This is done to sign the DNS queries sent by `dig`, as well as their responses. You can also specify the TSIG key itself on the command line using the `-y` option. The optional *hmac* is the type of TSIG; the default is `HMAC-MD5`. The *name* argument is the name of the TSIG key and the *key* argument is the actual key. The key is a base-64 encoded string, typically generated by `dnssec-keygen(1M)`.

Caution should be taken when using the `-y` option on multi-user systems, since the key can be visible in the output from `ps(1)` or in the shell's history file. When using TSIG authentication with `dig`, the name server that is queried needs to know the key and algorithm that is being used. In BIND, this is done by providing appropriate `key` and `server` statements in `named.conf`.

Query Options The `dig` utility provides a number of query options which affect the way in which lookups are made and the results displayed. Some of these set or reset flag bits in the query header, some determine which sections of the answer get printed, and others determine the timeout and retry strategies.

Each query option is identified by a keyword preceded by a plus sign (+). Some keywords set or reset an option. These may be preceded by the string no to negate the meaning of that keyword. Other keywords assign values to options like the timeout interval. They have the form `+keyword=`*value*. The query options are:

`+[no]tcp`
Use [do not use] TCP when querying name servers. The default behaviour is to use UDP unless an AXFR or IXFR query is requested, in which case a TCP connection is used.

`+[no]vc`
Use [do not use] TCP when querying name servers. This alternate syntax to `+[no]tcp` is provided for backwards compatibility. The "vc" stands for "virtual circuit".

`+[no]ignore`
Ignore truncation in UDP responses instead of retrying with TCP. By default, TCP retries are performed.

`+domain=`*somename*
Set the search list to contain the single domain *somename*, as if specified in a `domain` directive in `/etc/resolv.conf`, and enable search list processing as if the `+search` option were given.

`+[no]search`
Use [do not use] the search list defined by the `searchlist` or `domain` directive in `resolv.conf` (if any). The search list is not used by default.

`+[no]showsearch`
Perform [do not perform] a search showing intermediate results.

`+[no]defname`
Deprecated, treated as a synonym for `+[no]search`.

`+[no]aaonly`
Sets the aa flag in the query.

`+[no]aaflag`
A synonym for `+[no]aaonly`.

`+[no]adflag`
Set [do not set] the AD (authentic data) bit in the query. This requests that the server return, regardless of whether all of the answer and authority sections have all been validated as secure according to the security policy of the server. A setting of `AD=1` indicates that all records have been validated as secure and the answer is not from an `OPT-OUT` range. `AD=0` indicates that some part of the answer is insecure or not validated.

`+[no]cdflag`
Set [do not set] the CD (checking disabled) bit in the query. This requests the server to not perform DNSSEC validation of responses.

`+[no]cl`
Display [do not display] the CLASS when printing the record.

`+[no]ttlid`
Display [do not display] the TTL when printing the record.

`+[no]recurse`
Toggle the setting of the RD (recursion desired) bit in the query. This bit is set by default, which means `dig` normally sends recursive queries. Recursion is automatically disabled when the `+nssearch` or `+trace` query options are used.

`+[no]nssearch`
When this option is set, `dig` attempts to find the authoritative name servers for the zone containing the name being looked up and display the SOA record that each name server has for the zone.

`+[no]trace`
Toggle tracing of the delegation path from the root name servers for the name being looked up. Tracing is disabled by default. When tracing is enabled, `dig` makes iterative queries to resolve the name being looked up. It will follow referrals from the root servers, showing the answer from each server that was used to resolve the lookup.

`+[no]cmd`
Toggle the printing of the initial comment in the output identifying the version of `dig` and the query options that have been applied. This comment is printed by default.

`+[no]short`
Provide a terse answer. The default is to print the answer in a verbose form.

`+[no]identify`
Show [or do not show] the IP address and port number that supplied the answer when the *+short* option is enabled. If short form answers are requested, the default is not to show the source address and port number of the server that provided the answer.

`+[no]comments`
Toggle the display of comment lines in the output. The default is to print comments.

`+[no]stats`
Toggle the printing of statistics: when the query was made, the size of the reply and so on. The default behaviour is to print the query statistics.

`+[no]qr`
Print [do not print] the query as it is sent. By default, the query is not printed.

`+[no]question`
Print [do not print] the question section of a query when an answer is returned. The default is to print the question section as a comment.

`+[no]answer`
Display [do not display] the answer section of a reply. The default is to display it.

`+[no]authority`
Display [do not display] the authority section of a reply. The default is to display it.

`+[no]additional`
Display [do not display] the additional section of a reply. The default is to display it.

`+[no]all`
Set or clear all display flags.

`+time=`*T*
Sets the timeout for a query to *T* seconds. The default time out is 5 seconds. An attempt to set *T* to less than 1 will result in a query timeout of 1 second being applied.

`+tries=`*T*
Sets the maximum number of UDP attempts to *T*. The default number is 3 (1 initial attempt followed by 2 retries). If T is less than or equal to zero, the number of retries is silently rounded up to 1.

`+retry=`*T*
Sets the number of UDP retries to *T*. The default is 2.

`+ndots=`*D*
Set the number of dots that have to appear in *name* to *D* for it to be considered absolute. The default value is that defined using the `ndots` statement in `/etc/resolv.conf`, or 1 if no `ndots` statement is present. Names with fewer dots are interpreted as relative names and will be searched for in the domains listed in the `search` or `domain` directive in `/etc/resolv.conf`.

`+bufsize=`*B*
Set the UDP message buffer size advertised using EDNS0 to *B* bytes. The maximum and minimum sizes of this buffer are 65535 and 0 respectively. Values outside this range are rounded up or down appropriately.

`+edns=#`
Specify the EDNS version with which to query. Valid values are 0 to 255. Setting the EDNS version causes a EDNS query to be sent. `+noedns` clears the remembered EDNS version.

`+[no]multiline`
Print records like the SOA records in a verbose multi-line format with human-readable comments. The default is to print each record on a single line, to facilitate machine parsing of the `dig` output.

`+[no]fail`
Do not try the next server if you receive a `SERVFAIL`. The default is to not try the next server which is the reverse of normal stub resolver behavior.

`+[no]besteffort`
Attempt to display the contents of messages which are malformed. The default is to not display malformed answers.

`+[no]dnssec`
Request DNSSEC records be sent by setting the DNSSEC OK bit (DO) in the OPT record in the additional section of the query.

`+[no]sigchase`
Chase DNSSEC signature chains. Requires `dig` be compiled with `-DDIG_SIGCHASE`.

`+trusted-key=####`
Specifies a file containing trusted keys to be used with `+sigchase`. Each `DNSKEY` record must be on its own line.

If not specified dig will look for `/etc/trusted-key.key` then `trusted-key.key` in the current directory.

Requires `dig` be compiled with `-DDIG_SIGCHASE`.

`+[no]topdown`
When chasing DNSSEC signature chains, perform a top-down validation. Requires `dig` be compiled with `-DDIG_SIGCHASE`.

`+[no]nsid`
Include an EDNS name server ID request when sending a query.

Multiple Queries The BIND 9 implementation of `dig` supports specifying multiple queries on the command line (in addition to supporting the `-f` batch file option). Each of those queries can be supplied with its own set of flags, options and query options.

In this case, each *query* argument represent an individual query in the command-line syntax described above. Each consists of any of the standard options and flags, the name to be looked up, an optional query type, and class and any query options that should be applied to that query.

A global set of query options, which should be applied to all queries, can also be supplied. These global query options must precede the first tuple of name, class, type, options, flags, and query options supplied on the command line. Any global query options (except the `+[no]cmd` option) can be overridden by a query-specific set of query options. For example:

```
dig +qr www.isc.org any -x 127.0.0.1 isc.org ns +noqr
```

...shows how `dig` could be used from the command line to make three lookups: an ANY query for `www.isc.org`, a reverse lookup of 127.0.0.1 and a query for the NS records of `isc.org`. A global query option of `+qr` is applied, so that `dig` shows the initial query it made for each lookup. The final query has a local query option of `+noqr` which means that `dig` will not print the initial query when it looks up the NS records for `isc.org`.

Files `/etc/resolv.conf`
Resolver configuration file

`${HOME}/.digrc`
User-defined configuration file

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	network/dns/bind

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Volatile

See Also dnssec-keygen(1M), host(1M), named(1M), nslookup(1M), attributes(5)

RFC 1035

See the BIND 9 *Administrator's Reference Manual*. As of the date of publication of this man page, this document is available at https://www.isc.org/software/bind/documentation.

Bugs There are probably too many query options.

Notes nslookup(1M) and dig now report "Not Implemented" as NOTIMP rather than NOTIMPL. This will have impact on scripts that are looking for NOTIMPL.

Name directoryserver – front end for the Directory Server (DS)

Synopsis

```
/usr/sbin/directoryserver
     { setup [-f configuration_file] | uninstall}

/usr/sbin/directoryserver
     {start-admin | stop-admin | restart-admin | startconsole}

/usr/sbin/directoryserver [{-s | -server} server-instance ]
     {start | stop | restart}

/usr/sbin/directoryserver { -s |-server } server-instance
     { monitor | saveconfig | restoreconfig | db2index-task |
       ldif2db-task | ldif2db | ldif2ldap | vlvindex | db2ldif |
       db2ldif-task | db2bak | db2bak-task | bak2db | bak2db-task |
       suffix2instance | account-status | account-activate |
       account-inactivate }
     {...}

/usr/sbin/directoryserver nativetoascii | admin_ip | ldif |
     pwdhash | idsktune | mmldif | keyupg
     {...}

/usr/sbin/directoryserver { magt | sagt } {...}

/usr/sbin/directoryserver help [subcommand]
```

Description The `directoryserver` command is a comprehensive, front end to the utility programs provided by the Solaris Directory Server (DS).

Options for the `directoryserver` command itself must appear before the subcommand. Arguments for a subcommand must appear after the subcommand. Subcommands have specific arguments. See `SUBCOMMANDS`.

Subcommands The following subcommands are supported:

account-inactivate *args*

Inactivates and locks an entry or group of entries.

The `account-inactivate` subcommand supports the following arguments:

[`-D` *rootdn*] Directory Server *userDN* with root permissions, such as Directory Manager.

[`-h` *host*] Host name of Directory Server. The default value is the full hostname of the machine where Directory Server is installed.

`-I` *DN* Entry DN or role DN to activate.

`-j` *file*	Password associated with the user DN. This option allows the password to be stored in clear text in the named file for scripting. This is considered insecure. Use with extreme caution.
[`-p` *port*]	Directory Server port. The default value is the LDAP port of Directory Server specified at installation time.
`-w` *password*	Password associated with the user DN. Supplying the password on the command line is visible using the `/bin/ps` command. This is considered insecure. Use with extreme caution. The value `-` can be used in place the password. The program prompts the user for a password to be entered from the terminal.

account-activate *args*

Activates an entry or group of entries.

The `account-activate` subcommand supports the following arguments

`-D` *rootdn*	Directory Server userDN with root permissions, such as Directory Manager.
`-h` *host*	Host name of Directory Server. The default value is the full hostname of the machine where Directory Server is installed.
`-I` DN	Entry DN or role DN to activate.
`-j` *file*	Password associated with the user DN. This option allows the password to be stored in clear text in the named file for scripting. This is considered insecure. Use with extreme caution.

`-p` *port*	Directory Server port. The default value is the LDAP port of Directory Server specified at installation time.
`-w` *password*	Password associated with the user DN. Supplying the password on the command line is visible using the `/bin/ps` command. This is considered insecure. Use with extreme caution. The value `-` can be used in place the password. The program prompts the user for a password to be entered from the terminal.

account-status *args*

Provides account status information to establish whether an entry or group of entries is inactivated or not.

The `account-status` subcommand supports the following arguments:

`-D` *rootdn*

`-h` *host*	Host name of Directory Server. The default value is the full hostname of the machine where Directory Server is installed.
`-I` *DN*	Entry DN or role DN whose status is required.
`-j` *file*	Password associated with the user DN. This option allows the password to be stored in clear text in the named file for scripting. This is considered insecure. Use with extreme caution.
`-p` *port*	Directory Server port. The default value is the LDAP port of Directory Server specified at installation time.
`-w` *password*	Password associated with the rootDN. Supplying the password on the command line is visible using the

`/bin/ps` command. This is considered insecure. Use with extreme caution.

The value `-` can be used in place of the password. The program prompts the user for a password to be entered from the terminal.

admin_ip *args*

Change the IP address of the administrative server in the configuration.

The `admin_ip` subcommand supports the following arguments:

dir_mgr_DN	Directory Manager's DN.
dir_mgr_password	Directory Manager's password.
old_ip	Old IP.
new_ip	New IP.
port_#	Port number.

bak2db *backup_directory*

Restore the database from the most recent archived backup.

Specify *backup_directory* as the backup directory.

bak2db-task *args*

Restore the data to the database.

The bak2db-task subcommand supports the following arguments:

[`-a` *directory*]	Directory where the backup files are stored. By default it is under `/var/ds5/slapd-serverID/bak`
`-D` *rootDN*	User DN with root permissions, such as Directory Manager. The default is the DN of the directory manager which is read from the `nsslapd-root` attribute under `cn=config`.

	`-j` *file*	Password associated with the user DN. This option allows the password to be stored in clear text in the named file for scripting. This is considered insecure. Use with extreme caution.
	[`-t` *database_type*]	Database type. The only possible database type is `ldbm`.
	[`-v`]	Verbose mode.
	`-w` *password*	Password associated with the user DN. Supplying the password on the command line is visible using the `/bin/ps` command. This is considered insecure. Use with extreme caution. The value `-` can be used in place the password. The program prompts the user for a password to be entered from the terminal.
db2bak-task *args*	Back up the contents of the database. It creates an entry in the directory that launches this dynamic task. An entry is generated based upon the values provided for each option. The `db2bak-task` subcommand supports the following arguments:	
	[`-a` *directory*]	Directory where the backup files are stored. By default it is under `/var/ds5/slapd-serverID/bak`. The backup file is named according to the year-month-day-hour format (*YYYY_MM_DD_hhmmss*).
	`-D` *rootDN*	User DN with root permissions, such as Directory

		Manager. The default is the DN of the directory manager which is read from the `nsslapd-root` attribute under `cn=config`.
	`-j` *file*	Password associated with the user DN. This option allows the password to be stored in clear text in the named file for scripting. This is considered insecure. Use with extreme caution.
	`-t` *database_type*	Database type. The only possible database type is `ldbm`.
	[`-v`]	Verbose mode.
	`-w` *password*	Password associated with the user DN. Supplying the password on the command line is visible using the `/bin/ps` command. This is considered insecure. Use with extreme caution. The value `-` can be used in place the password. The program prompts the user for a password to be entered from the terminal.
db2bak [*backup_directory*]		Create a backup of the current database contents. The server must be stopped to run this subcommand. The default is `/var/ds5/slapd-serverID/bak`. The backup file is named according to the year-month-day-hour format (*YYYY_MM_DD_hhmmss*).
db2index-text *args*		Create and generate the new set of indexes to be maintained following the modification of indexing entries in the `cn=config` configuration file. The `db2index-text` subcommand supports the following arguments:

`-D` *rootdn*	User DN with root permissions, such as Directory Manager.
`-j` *file*	Password associated with the user DN. This option allows the password to be stored in clear text in the named file for scripting. This is considered insecure. Use with extreme caution.
`-n` *backend_instance*	Instance to be indexed.
[`-t` *attributeName*]	Name of the attribute to be indexed. If omitted, all indexes defined for that instance are generated.
[`-v`]	Verbose mode.
`-w` *password*	Password associated with the user DN. Supplying the password on the command line is visible using the `/bin/ps` command. This is considered insecure. Use with extreme caution. The value `-` can be used in place the password. The program prompts the user for a password to be entered from the terminal.

db2ldif-task *args*

Exports the contents of the database to LDIF. It creates an entry in the directory that launches this dynamic task. The entry is generated based upon the values you provide for each option. To run this subcommand the server must be running and either `-n` *backend_instance* or `-s` *include* suffix is required.

The `db2ldif-task` subcommand supports the following arguments:

[`-a` *outputfile*]	File name of the output LDIF file.

`-C`	Only the main db file is used.
`-D` *rootDN*	User DN with root permissions, such as Directory Manager.
`-j` *file*	Password associated with the user DN. This option allows the password to be stored in clear text in the named file for scripting. This is considered insecure.Use with extreme caution.
[`-M`]	Output LDIF is stored in multiple files.
[`-m`]	Minimal base 64 encoding.
{`-n` *backend_instance*}*	Instance to be exported.
[`-N`]	Minimal base 64 encoding.
[`-o`]	Output LDIF to be stored in one file by default with each instance stored in *instance_file* name.
[`-r`]	Export replica.
[`-s`]*includesuffix*}*	Suffix(es) to be included or to specify the subtrees to be included if `-n` has been used.
[`-u`]	Request that the unique ID is not exported.
[`-U`]	Request that the output LDIF is not folded.
`-w` *password*	Password associated with the user DN. Supplying the password on the command line is visible

		using the `/bin/ps` command. This is considered insecure. Use with extreme caution. The value `-` can be used in place the password. The program prompts the user for a password to be entered from the terminal.
	{`-x` *excludesuffix*}*	Suffixes to be excluded.
	[`-1`]	Delete, for reasons of backward compatibility the first line of the LDIF file that gives the version of the LDIF standard.
db2ldif *args*	Export the contents of the database to LDIF. You must specify either the `-n` or the `-s` option or both. The `db2ldif` subcommand supports the following options:	
	[`-a` *outputfile*]	File name of the output LDIF file.
	[`-C`]	Only use the main db file.
	[`-m`]	Minimal base64 encoding.
	[`-M`]	Use of several files for storing the output LDIF with each instance stored in *instance_file name* (where file name is the file name specified for `-a` option).
	{`-n` *baclemd_instance*}*	Instance to be exported.
	[`-N`]	Specify that the entry IDs are not to be included in the LDIF output. The entry IDs are necessary

	only if the `db2ldif` output is to be used as input to `db2index-text`.
[`-r`]	Export replica.
{`-s` *includesuffix*}*	Suffixes to be included or to specify the subtrees to be included if `-n` has been used.
[{`-x` *excludesuffix*}]*	Suffixes to be excluded.
[`-u`]	Request that the unique id is not exported.
[`-U`]	Request that the output LDIF is not folded.
[`-1`]	Delete, for reasons of backward compatibility, the first line of the LDIF file which gives the version of the LDIF standard.

help [*subcommand*]

Display `directoryserver` usage message or subcommand specific usage message.

idsktune *args*

Provide an easy and reliable way of checking the patch levels and kernel parameter settings for your system. You must install the Directory Server before you can run `idsktune`. It gathers information about the operating system, kernel, and TCP stack to make tuning recommendations.

The `idsktune` subcommand supports the following arguments:

[`-c`]	Client-specific tuning: the output only includes tuning recommendations for running a directory client application.
[`-D`]	Debug mode: the output includes the commands it runs internally, preceded by `DEBUG` heading.
[`-i` *installdir*]	The install directory.

[`-q`]	Quiet mode. Output only includes tuning recommendations. OS version statements are omitted.
[`-v`]	Version. Gives the build date identifying the version of the toll.

keyupg *args*

Upgrade the key from Lite to normal (only one way).

The `keyupg` subcommand supports the following arguments:

`-k`*key*	The key to be upgraded.
`-f` *key_file_path*	The key file path.

ldif2db-task args

Import data to the directory. It create an entry in the directory that launches this dynamic task. The entry is generated based upon the values you provide for each option. The server must be running when you run this subcommand.

The `ldif2sb-task` subcommand supports the following arguments:

[`-c`]	Request that only the core db is created without attribute indexes.
`-D` rootDN	User DN with root permissions, such as Directory Manager.
[`-g` *string*]	Generation of a unique ID. Enter `none` for no unique ID to be generated and `deterministic` for the generated unique ID to be name-based. Generates a time based unique ID by default. If you use the `deterministic` generation to have a name-based unique ID, you can also

	specify the namespace you want the server to use as follows:
`-g deterministic` *namespace_id*	
	where namespace_id is a string of characters in the following format
`00-xxxxxxxx-xxxxxxxx-xxxxxxxx-xxxxxxxx`	
	Use this option if you want to import the same LDIF file into two different directory servers, and you want the contents of both directories to have the same set of unique IDs. If unique IDs already exist in the LDIF file you are importing, then the existing IDs are imported to the server regardless of the options you have specified.
[`-G` *namespace_id*]	Generate a namespace ID as a name-based unique ID. This is the same as specifying `-g deterministic`.
{`-i` *filename*}*	File name of the input LDIF files. When you import multiple files, they are imported in the order in which you specify them on the command line.
`-j` *file*	Password associated with the user DN. This option allows the password to be stored in clear text in the named file for scripting. This is considered insecure. Use with extreme caution.
`-n` backend_instance	Instance to be imported.

	[-O]	Request that only the core db is created without attribute indexes.
	{-s *includesuffix* }*	Suffixes to be included. This argument can also be used to specify the subtrees to be included with -n.
	-w *password*	Password associated with the user DN. Supplying the password on the command line is visible using the /bin/ps command. This is considered insecure. Use with extreme caution. The value - can be used in place the password. The program prompts the user for a password to be entered from the terminal.
	[{-x *excludesuffix* }*]	
	[-v]	Verbose mode.
ldif *args*	Format LDIF files, and create base 64 encoded attribute values. With Base 64 Encoding you can represent binary data, such as a JPEG image, in LDIF by using base 64 encoding. You identify base 64 encoded data by using the :: symbol. The ldifsubcommand takes any input and formats it with the correct line continuation and appropriate attribute information. The subcommand also senses whether the input requires base 64 encoding. The ldif subcommand supports the following arguments	
	[-b]	Interpret the entire input as a single binary value. If -b is not present, each line is considered to be a separate input value.
	[*attrtype*]	If -b is specified, the output is attrtype:: <base 64 encoded

`value`.

ldif2db *args*

Import the data to the directory. To run this subcommand the server must be stopped. Note that ldif2db supports LDIF version 1 specifications. You can load an attribute using the URL specifier notation, for example:
`jpegphoto:file:///tmp/myphoto.jpg`

[`-c`] Merge chunk size.

[`-g` *string*] Generation of a unique ID. Type none for no unique ID to be generated and deterministic for the generated unique ID to be name-based. By default a time based unique ID is generated.

If you use the deterministic generation to have a name-based unique ID, you can also specify the namespace you want the server to use as follows:

`-g deterministic` *namespace_id*

where *namespace_id* is a string of characters in the following format:

`00-xxxxxxxx-xxxxxxxx-xxxxxxxx-xxxxxxxx`

Use this option if you want to import the same LDIF file into two different directory servers, and you want the contents of both directories to have the same set of unique IDs. If unique IDs already exist in the LDIF file you are importing, then the existing IDs are imported to the server regardless of the options you have specified.

	[`-G` *naemspace_id*]	Generate a namespace ID as a name-based unique ID. This is the same as specifying the `-g deterministic` option.
	{`-` *filename*}*	File name of the input LDIF file(s). When you import multiple files, they are imported in the order in which you specify them on the command line.
	`-n` *backend_instance*	Instance to be imported.
	[`-O`]	Request that only the core db is created without attribute indexes.
	{`-s` *includesuffix*}*	Suffixes to be included or to specify the subtrees to be included if `-n` has been used.
	[{`-x` *excludesuffix*}*]	Suffixes to be excluded

ldif2ldap *rootDN password filename*

Perform an import operation over LDAP to the Directory Server. To run this subcommand the server must be running.

The `ldif2ldap` subcommand supports the following arguments:

rootdn	User DN with root permissions, such as Directory Manager.
password	Password associated with the user DN.
filename	File name of the file to be imported. When you import multiple files, they are imported in the order in which you specify them on the command line.

magt CONFIG INIT

Start SNMP master agent. The Config and INIT files are in `/usr/iplanet/ds5/plugins/snmp/magt`. For more information, see the *iPlanet Directory Server 5.1 Administrator's Guide*.

The `magt` subcommand supports the following options:

CONFIG	The CONFIG file defines the community and the manager that master agent works with. Specify the manager value as a valid system name or an IP address.
INIT	The INIT file is a nonvolatile file that contains information from the MIB-II system group, including system location and contact information. If INIT doesn't already exist, starting the master agent for the first time creates it. An invalid manager name in the CONFIG file causes the master agent start-up to fail.

monitor

Retrieves performance monitoring information using the `ldapsearch` command-line utility.

mmldif *args*

Combine multiple LDIF files into a single authoritative set of entries. Typically each LDIF file is from a master server cooperating in a multi master replication agreement.[e.g. masters that refuse to sync up for whatever reason]. Optionally, it can generate LDIF change files that could be applied to original to bring it up to date with authoritative. At least two input files must be specified.

The `mmldif` subcommand supports the following arguments:

[`-c` *inputfile* ...]	Write a change file (.delta) for each input file. Specify *inputfile* as the input LDIF files.
[`-D`]	Print debugging information.
[`-o` *out.ldif*]	Write authoritative data to this file.

nativetoascii *args*

Convert one language encoding to another. For example, convert a native language to UTF-8 format.

The `nativetoascii` subcommand supports the following options:

`-d` *Encodings Directory*	Path to the

		directory which contains the `conv` directory
	[`-i` *input_filename* `-o` *output_filename*]	The input file name and output file name.
	`-l`	List supported encodings
	`-r`	Replace existing files.
	`-s` *suffix*	Suffix to be mapped to the backend.
	`-s` *SourceEncoding*	Source Encoding of input stream.
	`-t` *TargetEncoding*	Target Encoding of output stream.
	`-v`	Verbose output.
pwdhash *args*	Print the encrypted form of a password using one of the server's encryption algorithms. If a user cannot log in, you can use this script to compare the user's password to the password stored in the directory. The `pwdhash` subcommand supports the following arguments:	

	`-c` *comparepwd* | `-s` *scheme*	The available schemes are `SSHA`, `SHA`, `CRYPT` and `CLEARE`. It generates the encrypted passwords according to scheme's algorithm. The `-c` specifies the encrypted password to be compared with. The result of comparison is either `OK` or `doesn't match`.
	`-D` *instance-dir*	The instance directory.
	[`-H`]	The passwords are hex-encoded.
	password ...	The clear passwords to generate encrypted form from or to be compared with.
restart	Restarts the directory server. When the `-s` option is not specified, restarts all instances of servers. When the `-s` option is specified, restarts the server specified by `-s`. restart-admin Restarts the administration server.	
restoreconfig	Restores the most recently saved Administration Server configuration information to the NetscapeRoot partition under `/var/ds5/slapd-serverID/confbak`.	
sagt `-c` CONFIG	Start proxy SNMP agent. For more information, see the *iPlanet Directory Server 5.1 Administrator's Guide*. The `sagt` subcommand supports the following options:	

`-c` *configfile*	The CONFIG file includes the port that the SNMP daemon listens to. It also needs to include the MIB trees and traps that the proxy SNMP agent forwards. Edit the CONFIG file located in `/usr/iplanet/ds5/plugins/snmp/sagt`.
saveconfig	Saves the administration server configuration information to the `/var/ds5/slapd-serverID/confbak` directory.
setup [`-f` *configuration_file*]	Configures an instance of the directory server or administration server. Creates a basic configuration for the directory server and the administrative server that is used to manage the directory. The `setup` subcommand has two modes of operation. You can invoke it with a curses-based interaction to gather input. Alternatively, you can provide input in a configuration file using the `-f` option. The setup subcommand supports the following option: `-f` *configuration_file* Specifies the configuration file for silent installation.
start	Starts the directory server. When the `-s` option is not specified, starts servers of all instances. When the `-s` option is specified, starts the server instance specified by `-s`.
start-admin	Starts the directory server. When the `-s` option is not specified, restarts all instances of servers. When the `-s` option is specified, restarts the server specified by `-s`.
startconsole	Starts the directory console..
stop	Stops the directory server. When the `-s` option is not specified, restarts all instances of servers. When the `-s` option is specified, restarts the server specified by `-s`.
stop-admin	Stop the administration server.

suffix2instance {`-s` *suffix*}
: Map a suffix to a backend name.

  Specify `-s` *suffix* as the suffix to be mapped to the backend.

uninstall
: Uninstalls the directory server and the administration server.

  This subcommand stops servers of all instances and removes all the changes created by setup.

vlvindex *args*
: Create virtual list view (VLV) indexes, known in the Directory Server Console as Browsing Indexes. The server must be stopped beforehand.

  The `vlvindex` subcommand supports the following arguments:

  `-d` *debug_level*
  : Specify the debug level to use during index creation. Debug levels are defined in `nsslapd-errorlog-level (error Log Level)`. See the *iPlanet Directory Server 5.1 Configuration, Command, and File Reference*.

  `-n` *backend_instance*
  : Name of the database containing the entries to index.

  `-s` *suffix*
  : Name of the suffix containing the entries to index.

  `-T` *VLVTag*
  : Name of the database containing the entries to index.

Options Options for the `directoryserver` command itself must appear before the subcommand argument.

The following options are supported:

`-s` *server-instance*
`-server` *server-instance*
: The server instance name. Specify the directory server instance to process the command against. For some of the listed subcommands the server instance is optional and for other sub

commands it is a required option.

Examples

EXAMPLE 1 Starting All Instances of the Directory Servers

The following command starts all the instances of the directory servers:

```
example% directoryserver start
```

EXAMPLE 2 Starting the Instances of myhost of the Directory Server

The following command starts the instances `myhost` of the directory server.

```
example% directoryserver -s myhost start
```

EXAMPLE 3 Running the Monitor Tool and Outputting the Current Status

The following command runs the monitor tool and output the current status of the `ephesus` directory instance.

```
example% directoryserver -s ephesus monitor
```

EXAMPLE 4 Running the idsktune Tool and Outputting Performance Tuning Information

The following command runs the idsktune tool and outputs performance tuning information:

```
example% directoryserver idsktune
```

Exit Status

The following exit values are returned:

`0` Successful completion.

`non-zero` An error occurred.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	IPLTdsr, IPLTdsu

See Also

iPlanet Directory Server 5.1 Administrator's Guide

iPlanet Directory Server 5.1 Configuration, Command, and File Reference

Name disks – creates /dev entries for hard disks attached to the system

Synopsis `/usr/sbin/disks [-C] [-r` *rootdir*`]`

Description `devfsadm(1M)` is now the preferred command for `/dev` and should be used instead of `disks`.

`disks` creates symbolic links in the `/dev/dsk` and `/dev/rdsk` directories pointing to the actual disk device special files under the `/devices` directory tree. It performs the following steps:

1. `disks` searches the kernel device tree to see what hard disks are attached to the system. It notes the `/devices` pathnames for the slices on the drive and determines the physical component of the corresponding `/dev/dsk` or `/dev/rdsk` name.
2. The `/dev/dsk` and `/dev/rdsk` directories are checked for disk entries – that is, symbolic links with names of the form c*N*[t*N*]d*N*s*N*, or c*N*[t*N*]dNp*N*, where *N* represents a decimal number. c*N* is the logical controller number, an arbitrary number assigned by this program to designate a particular disk controller. The first controller found on the first occasion this program is run on a system, is assigned number `0`. t*N* is the bus-address number of a subsidiary controller attached to a peripheral bus such as SCSI or IPI (the `target` number for SCSI, and the `facility` number for IPI controllers). d*N* is the number of the disk attached to the controller. s*N* is the *slice* number on the disk. p*N* is the FDISK partition number used by `fdisk(1M)`. (x86 Only)
3. If only some of the disk entries are found in `/dev/dsk` for a disk that has been found under the `/devices` directory tree, disks creates the missing symbolic links. If none of the entries for a particular disk are found in `/dev/dsk`, `disks` checks to see if any entries exist for other disks attached to the same controller, and if so, creates new entries using the same controller number as used for other disks on the same controller. If no other `/dev/dsk` entries are found for slices of disks belonging to the same physical controller as the current disk, `disks` assigns the lowest-unused controller number and creates entries for the disk slices using this newly-assigned controller number.

`disks` is run automatically each time a reconfiguration-boot is performed or when `add_drv(1M)` is executed. When invoking `disks` manually, first run `drvconfig(1M)` to ensure `/devices` is consistent with the current device configuration.

Notice to Driver Writers `disks` considers all devices with a node type of `DDI_NT_BLOCK`, `DDI_NT_BLOCK_CHAN`, `DDI_NT_CD`, `DDI_NT_BLOCK_WWN` or `DDI_NT_CD_CHAN` to be disk devices. `disks` requires the minor name of disk devices obey the following format conventions.

The minor name for block interfaces consists of a single lowercase ASCII character, `a` through `u`, representing the slices and the primary partitions. The minor name for logical drive block interfaces consists of the strings `p5` through `p36`. The minor name for character (raw) interfaces consists of a single lowercase ASCII character, `a` through `a`, followed by the string `,raw`, representing the slices and the primary partitions. The minor name for logical drive character (raw) interfaces consists of the string `p5` through `p36` followed by `,raw`.

`disks` performs the following translations:

- a through p to s0 through s15
- q through u to p0 through p4
- p5 through p36 to p5 through p36

SPARC drivers should only use the first eight slices: a through h, while x86 drivers can use a through u, with q through u corresponding to fdisk(1M) primary partitions. q represents the entire disk, while r, s, t, and u represent up to four additional primary partitions. For logical drives, p5 to p36 correspond to the 32 logical drives that are supported. The device nodes for logical drives change dynamically as and when they are created or deleted.

To prevent disks from attempting to automatically generate links for a device, drivers must specify a private node type and refrain from using a node type: DDI_NT_BLOCK, DDI_NT_BLOCK_CHAN, DDI_NT_CD, or DDI_NT_CD_CHAN when calling ddi_create_minor_node(9F).

Options The following options are supported:

`-C` Causes disks to remove any invalid links after adding any new entries to /dev/dsk and /dev/rdsk. Invalid links are links which refer to non-existent disk nodes that have been removed, powered off, or are otherwise inaccessible.

`-r` *rootdir* Causes disks to presume that the /dev/dsk, /dev/rdsk and /devices directory trees are found under *rootdir*, not directly under /.

Errors If disks finds entries of a particular logical controller linked to different physical controllers, it prints an error message and exits without making any changes to the /dev directory, since it cannot determine which of the two alternative logical-to-physical mappings is correct. The links should be manually corrected or removed before another reconfiguration-boot is performed.

Examples **EXAMPLE 1** Creating Block and Character Minor Devices

The following example demonstrates creating the block and character minor devices from within the xkdisk driver's attach(9E) function.

```
#include    <sys/dkio.h>
/*
 * Create the minor number by combining the instance number
 * with the slice number.
 */
#define MINOR_NUM(i, s)   ((i) << 4 | (s))

int
xkdiskattach(dev_info_t *dip, ddi_attach_cmd_t cmd)
{
    int instance, slice;
    char name[8];
```

EXAMPLE 1 Creating Block and Character Minor Devices *(Continued)*

```
        /* other stuff in attach... */

    instance = ddi_get_instance(dip);
    for (slice = 0; slice < V_NUMPAR; slice++) {
        /*
         * create block device interface
         */
        sprintf(name, "%c", slice + 'a');
        ddi_create_minor_node(dip, name, S_IFBLK,
            MINOR_NUM(instance, slice), DDI_NT_BLOCK_CHAN, 0);

        /*
         * create the raw (character) device interface
         */
        sprintf(name,"%c,raw", slice + 'a');
        ddi_create_minor_node(dip, name, S_IFCHR,
            MINOR_NUM(instance, slice), DDI_NT_BLOCK_CHAN, 0);
    }
}
```

Installing the `xkdisk` disk driver on a Sun Fire 4800, with the driver controlling a SCSI disk (target 3 attached to an isp(7D) SCSI HBA) and performing a reconfiguration-boot (causing disks to be run) creates the following special files in `/devices`.

```
# ls -l /devices/ssm@0,0/pci@18,700000/pci@1/SUNW,isptwo@4/
brw-r-----   1 root sys   32, 16 Aug 29 00:02 xkdisk@3,0:a
crw-r-----   1 root sys   32, 16 Aug 29 00:02 xkdisk@3,0:a,raw
brw-r-----   1 root sys   32, 17 Aug 29 00:02 xkdisk@3,0:b
crw-r-----   1 root sys   32, 17 Aug 29 00:02 xkdisk@3,0:b,raw
brw-r-----   1 root sys   32, 18 Aug 29 00:02 xkdisk@3,0:c
crw-r-----   1 root sys   32, 18 Aug 29 00:02 xkdisk@3,0:c,raw
brw-r-----   1 root sys   32, 19 Aug 29 00:02 xkdisk@3,0:d
crw-r-----   1 root sys   32, 19 Aug 29 00:02 xkdisk@3,0:d,raw
brw-r-----   1 root sys   32, 20 Aug 29 00:02 xkdisk@3,0:e
crw-r-----   1 root sys   32, 20 Aug 29 00:02 xkdisk@3,0:e,raw
brw-r-----   1 root sys   32, 21 Aug 29 00:02 xkdisk@3,0:f
crw-r-----   1 root sys   32, 21 Aug 29 00:02 xkdisk@3,0:f,raw
brw-r-----   1 root sys   32, 22 Aug 29 00:02 xkdisk@3,0:g
crw-r-----   1 root sys   32, 22 Aug 29 00:02 xkdisk@3,0:g,raw
brw-r-----   1 root sys   32, 23 Aug 29 00:02 xkdisk@3,0:h
crw-r-----   1 root sys   32, 23 Aug 29 00:02 xkdisk@3,0:h,raw
```

/dev/dsk will contain the disk entries to the block device nodes in `/devices`

```
# ls -l /dev/dsk
/dev/dsk/c0t3d0s0 -> ../../devices/[...]/xkdisk@3,0:a
```

EXAMPLE 1 Creating Block and Character Minor Devices *(Continued)*

```
/dev/dsk/c0t3d0s1 -> ../../devices/[...]/xkdisk@3,0:b
/dev/dsk/c0t3d0s2 -> ../../devices/[...]/xkdisk@3,0:c
/dev/dsk/c0t3d0s3 -> ../../devices/[...]/xkdisk@3,0:d
/dev/dsk/c0t3d0s4 -> ../../devices/[...]/xkdisk@3,0:e
/dev/dsk/c0t3d0s5 -> ../../devices/[...]/xkdisk@3,0:f
/dev/dsk/c0t3d0s6 -> ../../devices/[...]/xkdisk@3,0:g
/dev/dsk/c0t3d0s7 -> ../../devices/[...]/xkdisk@3,0:h
```

and /dev/rdsk will contain the disk entries for the character device nodes in `/devices`

```
# ls -l /dev/rdsk
/dev/rdsk/c0t3d0s0 -> ../../devices/[...]/xkdisk@3,0:a,raw
/dev/rdsk/c0t3d0s1 -> ../../devices/[...]/xkdisk@3,0:b,raw
/dev/rdsk/c0t3d0s2 -> ../../devices/[...]/xkdisk@3,0:c,raw
/dev/rdsk/c0t3d0s3 -> ../../devices/[...]/xkdisk@3,0:d,raw
/dev/rdsk/c0t3d0s4 -> ../../devices/[...]/xkdisk@3,0:e,raw
/dev/rdsk/c0t3d0s5 -> ../../devices/[...]/xkdisk@3,0:f,raw
/dev/rdsk/c0t3d0s6 -> ../../devices/[...]/xkdisk@3,0:g,raw
/dev/rdsk/c0t3d0s7 -> ../../devices/[...]/xkdisk@3,0:h,raw
```

Files

`/dev/dsk/*`	Disk entries (block device interface)
`/dev/rdsk/*`	Disk entries (character device interface)
`/devices/*`	Device special files (minor device nodes)

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also `add_drv(1M)`, `devfsadm(1M)`, `fdisk(1M)`, `attributes(5)`, `isp(7D)`, `devfs(7FS)`, `dkio(7I)`, `attach(9E)`, `ddi_create_minor_node(9F)`

Writing Device Drivers

Bugs `disks` silently ignores malformed minor device names.

Name diskscan – perform surface analysis

Synopsis

```
diskscan [-W] [-n] [-y] raw_device
```

Description `diskscan` is used by the system administrator to perform surface analysis on a portion of a hard disk. The disk portion may be a raw partition or slice; it is identified using its raw device name. By default, the specified portion of the disk is read (non-destructive) and errors reported on standard error. In addition, a progress report is printed on standard out. The list of bad blocks should be saved in a file and later fed into `addbadsec(1M)`, which will remap them.

Options The following options are supported:

`-n` Causes `diskscan` to suppress linefeeds when printing progress information on standard out.

`-W` Causes `diskscan` to perform write and read surface analysis. This type of surface analysis is destructive and should be invoked with caution.

`-y` Causes `diskscan` to suppress the warning regarding destruction of existing data that is issued when `-W` is used.

Operands The following operands are supported:

raw_device The address of the disk drive (see `FILES`).

Files The raw device should be `/dev/rdsk/c?[t?]d?[ps]?`. See `disks(1M)` for an explanation of SCSI and IDE device naming conventions.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	x86
Availability	system/core-os

See Also `addbadsec(1M)`, `disks(1M)`, `fdisk(1M)`, `fmthard(1M)`, `format(1M)`, `attributes(5)`

Notes The `format(1M)` utility is available to format, label, analyze, and repair SCSI disks. This utility is included with the `diskscan`, `addbadsec(1M)`, `fdisk(1M)`, and `fmthard(1M)` commands available for x86. To format an IDE disk, use the DOS `format` utility; however, to label, analyze, or repair IDE disks on x86 systems, use the Solaris `format(1M)` utility.

Name dispadmin – process scheduler administration

Synopsis

```
dispadmin -l

dispadmin -c class {-g [-r res] | -s file}

dispadmin -d [class]
```

Description The `dispadmin` command displays or changes process scheduler parameters while the system is running.

`dispadmin` does limited checking on the values supplied in *file* to verify that they are within their required bounds. The checking, however, does not attempt to analyze the effect that the new values have on the performance of the system. Inappropriate values can have a negative effect on system performance. (See *Oracle Solaris Administration: Common Tasks*.)

Options The following options are supported:

`-c` *class*
Specifies the class whose parameters are to be displayed or changed. Valid *class* values are: `RT` for the real-time class, `TS` for the time-sharing class, `IA` for the inter-active class, `FSS` for the fair-share class, and FX for the fixed-priority class. The time-sharing and inter-active classes share the same scheduler, so changes to the scheduling parameters of one will change those of the other.

`-d` [*class*]
Sets or displays the name of the default scheduling class to be used on reboot when starting `svc:/system/scheduler:default`. If class name is not specified, the name and description of the current default scheduling class is displayed. If class name is specified and is a valid scheduling class name, then it is saved in `dispadmin`'s private configuration file `/etc/dispadmin.conf`. Only super-users can set the default scheduling class.

`-g`
Gets the parameters for the specified class and writes them to the standard output. Parameters for the real-time class are described in `rt_dptbl(4)`. Parameters for the time-sharing and inter-active classes are described in `ts_dptbl(4)`. Parameters for the fair-share class are described in `FSS(7)`. Parameters for the fixed-priority class are described in `fx_dptbl(4)`.

The `-g` and `-s` options are mutually exclusive: you may not retrieve the table at the same time you are overwriting it.

`-l`
Lists the scheduler classes currently configured in the system.

`-r` *res*
When using the `-g` option you may also use the `-r` option to specify a resolution to be used for outputting the time quantum values. If no resolution is specified, time quantum values are in milliseconds. If *res* is specified it must be a positive integer between 1 and 1000000000 inclusive, and the resolution used is the reciprocal of *res* in seconds. For

example, a *res* value of 10 yields time quantum values expressed in tenths of a second; a *res* value of 1000000 yields time quantum values expressed in microseconds. If the time quantum cannot be expressed as an integer in the specified resolution, it is rounded up to the next integral multiple of the specified resolution.

`-s` *file*
Sets scheduler parameters for the specified class using the values in *file*. These values overwrite the current values in memory—they become the parameters that control scheduling of processes in the specified class. The values in *file* must be in the format output by the `-g` option. Moreover, the values must describe a table that is the same size (has same number of priority levels) as the table being overwritten. Super-user privileges are required in order to use the `-s` option.

Specify time quantum values for scheduling classes in system clock ticks, and not in constant-time units. Time quantum values are based on the value of the kernel's `hz` variable. If kernel variable `hires_tick` is set to `1` to get higher resolution clock behavior, the actual time quanta will be reduced by the order of `10`.

The `-g` and `-s` options are mutually exclusive: you may not retrieve the table at the same time you are overwriting it.

Examples

EXAMPLE 1 Retrieving the Current Scheduler Parameters for the real-time class

The following command retrieves the current scheduler parameters for the real-time class from kernel memory and writes them to the standard output. Time quantum values are in microseconds.

```
dispadmin -c RT -g -r 1000000
```

EXAMPLE 2 Overwriting the Current Scheduler Parameters for the Real-time Class

The following command overwrites the current scheduler parameters for the real-time class with the values specified in `rt.config`.

```
dispadmin -c RT -s rt.config
```

EXAMPLE 3 Retrieving the Current Scheduler Parameters for the Time-sharing Class

The following command retrieves the current scheduler parameters for the time-sharing class from kernel memory and writes them to the standard output. Time quantum values are in nanoseconds.

```
dispadmin -c TS -g -r 1000000000
```

EXAMPLE 4 Overwriting the Current Scheduler Parameters for the Time-sharing Class

The following command overwrites the current scheduler parameters for the time-sharing class with the values specified in `ts.config`.

```
dispadmin -c TS -s ts.config
```

Files `/etc/dispadmin.conf`
Possible location for argument to `-s` option.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also `priocntl(1)`, `svcs(1)`, `svcadm(1M)`, `priocntl(2)`, `fx_dptbl(4)`, `rt_dptbl(4)`, `ts_dptbl(4)`, `attributes(5)`, `smf(5)`, `FSS(7)`

Oracle Solaris Administration: Common Tasks Programming Interfaces Guide

Diagnostics `dispadmin` prints an appropriate diagnostic message if it fails to overwrite the current scheduler parameters due to lack of required permissions or a problem with the specified input file.

Notes The default scheduling class setting facility is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/system/scheduler:default
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. Note that disabling the service while it is running will not change anything. The service's status can be queried using the `svcs(1)` command.

Name distro_const – Utility for creating Oracle Solaris images and media

Synopsis

```
/usr/bin/distro_const -h

/usr/bin/distro_const build [-v] [-r checkpoint_name]
    [-p checkpoint_name] [-l] manifest
```

Description The `distro_const` command enables users to create an image by using a specified manifest file as the blueprint for the image.

You can create any of the following images:

- A text installer image that can be used to install the Oracle Solaris operating system on either x86 systems or SPARC systems.
- An ISO image that is comparable to a Live DVD image containing the Oracle Solaris operating system.
- A SPARC AI ISO image that can be used for network installations of the Oracle Solaris OS on SPARC clients, or an x86 AI ISO image that can be used for network installations of the Oracle Solaris OS on x86 clients.
- A custom ISO image.

The `distro_const build` command with no options creates a full image in one step.

Options enable you to pause and resume the image creation process at various checkpoints, thus enabling you to review status of the image and to check for bugs at each stage. Checkpointing saves time during builds by allowing you to bypass lengthy steps that have already been done at least once.

Note – You must assume the root role or have root privileges to run the `distro_const` command.

When using the distribution constructor, you can create only SPARC images on a SPARC system, and you can create only x86 images on an x86 system. Also, the operating system release version on your system must be the same release version as the image that you are building.

Options

`-h`
`--help`
Display a usage message.

Sub-commands The `distro_const` command has the subcommand and options listed below. Also see the "Examples" section.

`build` [`-v`] [`-r` *resume_checkpoint*] [`-p` *pause_checkpoint*] [`-l`] *manifest*
With no options, create a full image, using the specified manifest file as the blueprint for that image.

`-v`
`--verbose`
Show verbose output.

`-l`
`--list`
List all valid checkpoints at which you can choose to pause or resume building an image. This option queries the *manifest* manifest file for valid checkpoints. Use the names provided by this option as valid values for the other checkpointing options.

`-p` *pause_checkpoint*
`--pause` *pause_checkpoint*
Build an image, pausing at the specified checkpoint name. Use the `-l` option to find valid checkpoint names. You can combine the `-p` and `-r` options.

`-r` *resume_checkpoint*
`--resume` *resume_checkpoint*
Resume building the image from the specified checkpoint name. The specified checkpoint name must be either the checkpoint at which the previous build stopped executing, or an earlier checkpoint. A later checkpoint is not valid. Use the `-l` option to determine which checkpoints are resumable. You can combine the `-r` and `-p` options.

Examples

EXAMPLE 1 Create an Image Using Checkpoint Options

1. Check which checkpoints are available.

```
# distro_const build -l /usr/share/distro_const/dc_text_x86.xml
Checkpoint           Resumable Description
-----------          --------- ----------------------
transfer-ips-install X         Transfer pkg contents from IPS
set-ips-attributes   X         Set post-install IPS attributes
pre-pkg-img-mod      X         Pre-package image modification
ba-init              X         Boot archive initialization
ba-config            X         Boot archive configuration
ba-arch              X         Boot archive archival
boot-setup           X         Set up GRUB menu
pkg-img-mod          X         Pkg image area modification
create-iso                     ISO media creation
create-usb                     USB media creation
```

2. Start building the image and pause at the `ba-init` checkpoint.

```
# distro_const build -p ba-init /usr/share/distro_const/dc_text_x86.xml
```

3. Restart the build from the `ba-init` checkpoint. Finish creating the image.

```
# distro_const build -r ba-init /usr/share/distro_const/dc_text_x86.xml
```

EXAMPLE 2 Create an Image in One Step

To run a complete build of an image without pausing, use the `distro_const` command with no options.

```
# distro_const build /usr/share/distro_const/dc_text_x86.xml
```

Attributes

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	`install/distribution-constructor`
Interface Stability	Uncommitted

See Also

`dc_manifest`(4)

Creating a Custom Oracle Solaris 11.1 Installation Image

Name dladm – administer data links

Synopsis

```
dladm
```

```
dladm show-link [-PZ] [-s [-i interval]] [[-p] -o field[,...]]
     [-z zone[,...]] [link]
dladm rename-link [-R root-dir] link new-link
```

```
dladm delete-phys phys-link
dladm show-phys [-PZ] [-Lm] [[-p] -o field[,...]] [-H]
     [-z zone[,...]] [-D [dcb-feature]] [phys-link]
```

```
dladm create-aggr [-t] [-R root-dir] [-m mode] [-P policy] [-L lacpmode]
     [-T time] [-u address] -l ether-link1 [-l ether-link2...] aggr-link
dladm modify-aggr [-t] [-R root-dir] [-m mode] [-P policy] [-L lacpmode]
     [-T time] [-u address] aggr-link
dladm delete-aggr [-t] [-R root-dir] aggr-link
dladm add-aggr [-t] [-R root-dir] -l ether-link1 [-l ether-link2...]
     aggr-link
dladm remove-aggr [-t] [-R root-dir] -l ether-link1 [-l ether-link2...]
     aggr-link
dladm show-aggr [-PLxZ] [-s [-i interval]] [[-p] -o field[,...]]
     [-z zone[,...]] [aggr-link]
```

```
dladm create-bridge [-P protect] [-R root-dir] [-p priority]
     [-m max-age] [-h hello-time] [-d forward-delay] [-f force-protocol]
     [-l link...] bridge-name
dladm modify-bridge [-P protect] [-R root-dir] [-p priority]
     [-m max-age] [-h hello-time] [-d forward-delay] [-f force-protocol]
     bridge-name
dladm delete-bridge [-R root-dir] bridge-name
dladm add-bridge [-R root-dir] -l link [-l link...]bridge-name
dladm remove-bridge [-R root-dir] -l link [-l link...] bridge-name
dladm show-bridge [-flt] [-s [-i interval]] [[-p] -o field,...]
     [bridge-name]
```

```
dladm create-vlan [-ft] [-R root-dir] -l ether-link -v vid [vlan-link]
dladm modify-vlan [-t] [-R root-dir] [-l ether-link] [-v vid [-f]]
     {vlan-link,[vlan-link,...] | -L ether-link}
dladm delete-vlan [-t] [-R root-dir] vlan-link
dladm show-vlan [-PZ] [[-p] -o field[,...]] [-z zone[,...]] [vlan-link]
```

```
dladm scan-wifi [[-p] -o field[,...]] [wifi-link]
dladm connect-wifi [-e essid] [-i bssid] [-k key,...]
     [-s none | wep | wpa ] [-a open | shared] [-b bss | ibss] [-c]
     [-m a | b | g | n ] [-T time] [wifi-link]
dladm disconnect-wifi [-a] [wifi-link]
dladm show-wifi [-Z] [[-p] -o field[,...]] [-z zone[,...]] [wifi-link]
```

```
dladm show-ether [-xZ] [[-p] -o field[,...]] [-z zone[,...]]
     [-P protocol] [ether-link]
```

```
dladm set-linkprop [-t] [-R root-dir] -p prop=value[,...] link
dladm reset-linkprop [-t] [-R root-dir] [-p prop[,...]] link
dladm show-linkprop [-PZ] [[-c] -o field[,...]] [-p prop[,...]]
     [-z zone[,...]] [link]

dladm create-secobj [-t] [-R root-dir] [-f file] -c class secobj
dladm delete-secobj [-t] [-R root-dir] secobj[,...]
dladm show-secobj [-P] [[-p] -o field[,...]] [secobj,...]

dladm create-vnic [-t] -l link [-R root-dir] [-m value | auto |
     {factory [-n slot-identifier]} | {vrrp -A {inet | inet6} -V vrid}
     | {random [-r prefix]}] [-v vlan-id] [-p prop=value[,...]] vnic-link
dladm modify-vnic [-t] [-R root-dir] [-l link]  [-m value | auto |
     {factory [-n slot-identifier]} | {vrrp -A {inet | inet6} -V vrid}
     | {random [-r prefix]}] [-v vlan-id]
     {vnic-link,[vnic-link,...] | -L link}
dladm delete-vnic [-t] [-R root-dir] vnic-link
dladm show-vnic [-pPZ] [-s [-i interval]] [-o field[,...]]
     [-l link] [-z zone[,...]] [vnic-link]

dladm create-etherstub [-t] [-R root-dir] etherstub
dladm delete-etherstub [-t] [-R root-dir] etherstub
dladm show-etherstub [-Z] [-z zone[,...]] [etherstub]

dladm create-iptun [-t] [-R root-dir] -T type [-a {local|remote}=addr,...]
     iptun-link
dladm modify-iptun [-t] [-R root-dir] -a {local|remote}=addr,...
     iptun-link
dladm delete-iptun [-t] [-R root-dir] iptun-link
dladm show-iptun [-PZ] [[-p] -o field[,...]] [-z zone[,...]] [iptun-link]

dladm create-part [-t] [-f] -l ib-link [-R root-dir] -P pkey
     [-p prop=value[,...]] part-link
dladm delete-part [-t] [-R root-dir] part-link
dladm show-part [-pP] [-o field[,...]] [-l ib-link]  [part-link]

dladm show-ib [-pP] [-o field[,...]] [ib-link]

dladm show-usage [-a] -f filename [-p plotfile -F format] [-s time]
     [-e time] [link]

dladm help [subcommand-name]
```

Description The `dladm` command is used to administer data-links. A data-link is represented in the system as a STREAMS DLPI (v2) interface which can be plumbed under protocol stacks such as TCP/IP. Each data-link relies on either a single network device or an aggregation of devices to send packets to or receive packets from a network.

Each `dladm` subcommand operates on one of the following objects:

`link`
A datalink, identified by a name. In general, the name can use any alphanumeric characters (or the underscore, `_`, or the period, `.`), but must start with an alphabetic character and end with a number. A datalink name can be at most 31 characters, and the ending number must be between 0 and 4294967294 (inclusive). The ending number must not begin with a zero. Datalink names between 3 and 8 characters are recommended.

Some subcommands operate only on certain types or classes of datalinks. For those cases, the following object names are used:

`aggr-link`
An aggregation datalink (or a key; see NOTES).

`ether-link`
A physical Ethernet datalink.

`iptun-link`
An IP tunnel link.

`part-link`
An InfiniBand (IB) partition data link.

`phys-link`
A physical datalink.

`vlan-link`
A VLAN datalink.

`vnic-link`
A virtual network interface created on a link or an `etherstub`. It is a pseudo device that can be treated as if it were an network interface card on a machine.

`wifi-link`
A WiFi datalink.

`bridge`
A bridge instance, identified by an administratively-chosen name. The name may use any alphanumeric characters or the underscore, `_`, but must start and end with an alphabetic character. A bridge name can be at most 31 characters. The name `default` is reserved, as are all names starting with `SUNW`.

Note that appending a zero (`0`) to a bridge name produces a valid link name, used for observability.

Also note that the bridge-related subcommands, described with `dladm` subcommands below, require installation of the `pkg://solaris/network/bridging` package.

`dev`
A network device, identified by concatenation of a driver name and an instance number.

`etherstub`
An Ethernet stub can be used instead of a physical NIC to create VNICs. VNICs created on an `etherstub` will appear to be connected through a virtual switch, allowing complete virtual networks to be built without physical hardware.

`part`
An IB partition link created on a IB physical link.

`secobj`
A secure object, identified by an administratively-chosen name. The name can use any alphanumeric characters, as well as underscore (_), period (.), and hyphen (-). A secure object name can be at most 32 characters.

`dladm` is implemented as a set of subcommands with corresponding options. Options are described in the context of each subcommand. Many of the subcommands have the following as a common option:

`-R` *root-dir*, `--root-dir=`*root-dir*
Specifies an alternate root directory where the operation-such as creation, deletion, or renaming-should apply.

`dladm` also supports a command form with no arguments. When invoked this way, `dladm` displays basic configuration information for all datalinks on a system. See `EXAMPLES`.

SUBCOMMANDS The following subcommands are supported:

`dladm show-link` [`-PZ`] [`-s` [`-i` *interval*]] [[`-p`] `-o` *field*[,...]] [`-z` *zone*[,...]] [*link*]
Show link configuration information either for all datalinks or for the specified link. By default, the system is configured with one datalink for each known network device. The option to print link statistics is moved to `dlstat(1M)`.

`-o` *field*[,...], `--output=`*field*[,...]
A case-insensitive, comma-separated list of output fields to display. When not modified by the `-s` option (described below), the field name must be one of the fields listed below, or the special value `all` to display all fields. By default (without `-o`), `show-link` displays all fields.

`LINK`
The name of the datalink.

`ZONE`
The current zone of the datalink.

`CLASS`
The class of the datalink. `dladm` distinguishes between the following classes:

`aggr`
Link Aggregation either as Datalink Multipathing (`dlmp`) or IEEE 802.3ad `trunk`. The `show-aggr` subcommand displays more details for this class of datalink.

`bridge`
A bridge instance, identified by an administratively-chosen name.

`etherstub`
Instance of an etherstub. An Ethernet stub can be used instead of a physical NIC to create VNICs. VNICs created on an `etherstub` will appear to be connected through a virtual switch, allowing complete virtual networks to be built without physical hardware.

`iptun`
An instance of an IP tunnel link.

`part`
An IP-over-IB interface. The `show-part` subcommand displays more detail for this class of datalink.

`phys`
A physical datalink. The `show-phys` subcommand displays more detail for this class of datalink.

`vlan`
A VLAN datalink. The `show-vlan` subcommand displays more detail for this class of datalink.

`vnic`
A virtual network interface. The `show-vnic` subcommand displays more detail for this class of datalink.

`MTU`
The maximum transmission unit size for the datalink being displayed.

`STATE`
The link state of the datalink. The state can be `up`, `down`, or `unknown`.

`BRIDGE`
The name of the bridge to which this link is assigned, if any.

`OVER`
The physical datalink(s) over which the datalink is operating. This applies to `aggr`, `bridge`, and `vlan` and `part` partition classes of datalinks. A VLAN or IB partition is created over a single physical datalink, a bridge has multiple attached links, and an aggregation is comprised of one or more physical datalinks.

When the `-o` option is used in conjunction with the `-s` option, used to display link statistics, the field name must be one of the fields listed below, or the special value `all` to display all fields

`LINK`
The name of the datalink.

`IPACKETS`
Number of packets received on this link.

`RBYTES`
Number of bytes received on this link.

`IERRORS`
Number of input errors.

`OPACKETS`
Number of packets sent on this link.

`OBYTES`
Number of bytes sent on this link.

`OERRORS`
Number of output errors.

`-p`, `--parseable`
Display using a stable machine-parseable format. The `-o` option is required with `-p`. See "Parseable Output Format", below.

`-P`, `--persistent`
Display the persistent link configuration.

`-s`, `--statistics`
Display link statistics. This option is made obsolete by `dlstat(1M)`.

`-i` *interval*, `--interval=`*interval*
Used with the `-s` option to specify an interval, in seconds, at which statistics should be displayed. This option is made obsolete by `dlstat(1M)`.

`-Z`
Display `ZONE` column in the output.

`-z` *zone*[,...]
Display links from the specified zones. By default, `dladm` displays links in all the zones when it is run from the global zone. The links in other zones are displayed with the corresponding zonename as its prefix, followed by the slash (`/`) separator. For example, `zone1/net0`

When run from a non-global zone, this subcommand displays only links from that zone. A non-global zone cannot see links in other zones.

`dladm rename-link` [`-R` *root-dir*] *link new-link*
Rename *link* to *new-link*. This is used to give a link a meaningful name, or to associate existing link configuration such as link properties of a removed device with a new device. See the `EXAMPLES` section for specific examples of how this subcommand is used.

`-R` *root-dir*, `--root-dir=`*root-dir*
See "Options," above.

`dladm delete-phys` *phys-link*
This command is used to delete the persistent configuration of a link associated with physical hardware which has been removed from the system. See the `EXAMPLES` section.

`dladm show-phys` [`-PZ`] [`-Lm`] [[`-p`] `-o` *field*[,...]] [`-H`] [`-z` *zone*[,...]] [`-D` [*dcb-feature*]] [*phys-link*]
Show the physical device and attributes of all physical links, or of the named physical link. Without `-P`, only physical links that are available on the running system are displayed.

`-D` [*dcb-feature*]
Show DCB (Data Center Bridging)-related configuration information on the phys-link. Supported *dcb-feature*s include `ets` (Enhanced Transmission Selection, IEEE 802.1Qaz) and `pfc` (Priority-based Flow Control, IEEE 802.1Qbb). Output from `-D ets` displays the following elements for ETS:

`LINK`
A physical device corresponding to a NIC driver.

`COS`
802.1p priority value.

`ETSBW`
The configured ETS BW as a percentage for the CoS (802.1p priority) value.

`ETSBW_EFFECT` (%age)
The effective ETS BW as a percentage for the CoS (802.1p priority) value.

`CLIENTS`
MAC clients that are using the CoS value.

Output from `-D pfc` displays the `LINK`, `COS`, and `CLIENTS` fields, just the same as the `-D ets` output. In addition, `-D pfc` displays the following elements specifically for PFC:

`PFC`
If the configured PFC is enabled for the CoS (802.1p priority) value.

`PFC_EFFECT`
If the effective PFC is enabled for the CoS (802.1p priority) value.

`-H`
Show hardware resource usage, as returned by the NIC driver. Output from `-H` displays the following elements:

`LINK`
A physical device corresponding to a NIC driver.

`RINGTYPE`
The type of the ring, either RX or TX.

`RINGS`
The ring index. A ring is an hardware resource, which typically maps to a DMA channel, that can be programmed for specific use. For example, an RX ring can be programmed to receive only packets belonging to a specific MAC address.

`CLIENTS`
MAC clients that are using the rings.

`-L`
Display location information for the physical devices/links. Output is in location order—that is, onboard devices before expansion slots—and location information (for example, PCIexp Slot 2, MB) is supplied where available. Output from `-L` supports the following elements:

`LINK`
A physical device corresponding to a NIC driver.

`DEVICE`
The name of the physical device under this link.

`LOC`
Physical location description string (where available).

`-m`
Display the list of factory MAC addresses, their slot identifiers, and their availability.

`-o` *field*, `--output=`*field*
A case-insensitive, comma-separated list of output fields to display. The field name must be one of the fields listed below, or the special value `all`, to display all fields. For each link, the following fields can be displayed:

`LINK`
The name of the datalink.

`MEDIA`
The media type provided by the physical datalink.

`STATE`
The state of the link. This can be `up`, `down`, or `unknown`.

`SPEED`
The current speed of the link, in megabits per second.

`DUPLEX`
For Ethernet links, the full/half duplex status of the link is displayed if the link state is `up`. The duplex is displayed as `unknown` in all other cases.

`DEVICE`
The name of the physical device under this link.

`-p`, `--parseable`
Display using a stable machine-parseable format. The `-o` option is required with `-p`. See "Parseable Output Format", below.

`-P`, `--persistent`
This option displays persistent configuration for all links, including those that have been removed from the system. The output provides a `FLAGS` column in which the `r` flag indicates that the physical device associated with a physical link has been removed. For such links, `delete-phys` can be used to purge the link's configuration from the system.

`-Z`
Display `ZONE` column in the output.

`-z` *zone*[,...]
See description of `-z` option under `dladm show-link`, above.

By default, Solaris assigns link names with the prefix of `net`. Before installing Solaris, you can change this default by modifying the value of the `linkname-policy/phys-prefix` SMF property of the service `svc:/network/datalink-management:default`. Specify a new value for this property in the System Configuration manifests used the Automated Install (AI) program. See *Oracle Solaris Administration: Network Interfaces and Network Virtualization* for details.

`dladm create-aggr` [`-t`] [`-R` *root-dir*] [`-m` *mode*] [`-P` *policy*] [`-L` *lcapmode*] [`-T` *time*] [`-u` *address*] `-l` *ether-link1* [`-l` *ether-link2*...] *aggr-link*
Combine a set of links into a single link aggregation named *aggr-link*. The aggregation could be HA-only or IEEE 802.3ad compliant. The use of an integer *key* to generate a link name for the aggregation is also supported for backward compatibility. Many of the `*-aggr` subcommands below also support the use of a *key* to refer to a given aggregation, but use of the aggregation link name is preferred. See the `NOTES` section for more information on keys.

`dladm` supports a number of port selection policies for an aggregation of ports. (See the description of the `-P` option, below.) If you do not specify a policy, `create-aggr` uses the default, the L4 policy, described under the `-P` option.

`-l` *ether-link*, `--link=`*ether-link*
Each Ethernet link (or port) in the aggregation is specified using an `-l` option followed by the name of the link to be included in the aggregation. Multiple links are included in the aggregation by specifying multiple `-l` options. For backward compatibility with previous versions of Solaris, the `dladm` command also supports the using the `-d` option (or `--dev`) with a device name to specify links by their underlying device name. The other `*-aggr` subcommands that take `-l`options also accept `-d`.

`-t`, `--temporary`
Specifies that the aggregation is temporary. Temporary aggregations last until the next reboot.

`-R` *root-dir*, `--root-dir=`*root-dir*
See "Options," above.

`-m` *mode*
Mode must be set to one of the following:

`trunk`
IEEE 802.3ad compliant link aggregation. If unspecified, *mode* is `trunk`.

`dlmp`
Datalink Multipathing mode. A layer 2 high availability technology that can provide failover among multiple switches, and does not require switch configuration. A `dlmp` link aggregation can also aggregate ports connected to same switch. However, it cannot be used in back-to-back setup.

An `dlmp` link aggregation is limited in its load-spreading ability: MAC clients configured on plumbed `dlmp` `aggr` are distributed across all `aggr` ports but an individual MAC client cannot spread load across multiple ports.

This mode is not IEEE 802.3ad compliant. Setting `policy`, `lacpmode`, `time` or MAC address is invalid in this mode.

`-P` *policy*, `--policy=`*policy*
Specifies the port selection policy to use for load spreading of outbound traffic. The policy specifies which *dev* object is used to send packets. A policy is a list of one or more layers specifiers separated by commas. A layer specifier is one of the following:

`L2`
Select outbound device according to source and destination MAC addresses of the packet.

`L3`
Select outbound device according to source and destination IP addresses of the packet.

`L4`
Select outbound device according to the upper layer protocol information contained in the packet. For TCP and UDP, this includes source and destination ports. For IPsec, this includes the SPI (Security Parameters Index).

For example, to use upper layer protocol information, the following policy can be used:

```
-P L4
```

Note that policy L4 is the default.

To use the source and destination MAC addresses as well as the source and destination IP addresses, the following policy can be used:

```
-P L2,L3
```

`-L` *lacpmode*, `--lacp-mode=`*mode*
Specifies whether LACP should be used and, if used, the mode in which it should operate. Supported values are `off`, `active` or `passive`.

`-T` *time*, `--lacp-timer=`*time*
Specifies the LACP timer value. The supported values are `short` or `long`.

`-u` *address*, `--unicast=`*address*
Specifies a fixed unicast hardware address to be used for the aggregation. If this option is not specified, then an address is automatically chosen from the set of addresses of the component devices.

`dladm modify-aggr` [`-t`] [`-R` *root-dir*] [`-m` *mode*] [`-P` *policy*] [`-L` *lacpmode*] [`-T` *time*] [`-u` *address*] *aggr-link*
Modify the parameters of the specified aggregation.

`-t`, `--temporary`
Specifies that the modification is temporary. Temporary aggregations last until the next reboot.

`-R` *root-dir*, `--root-dir=`*root-dir*
See "Options," above.

`-m` *mode*
See description of `-m` *mode* option under `create-aggr` subcommand, above.

`-P` *policy*, `--policy=`*policy*
Specifies the port selection policy to use for load spreading of outbound traffic. See `dladm create-aggr` for a description of valid policy values.

`-L` *lacpmode*, `--lacp-mode=`*mode*
Specifies whether LACP should be used and, if used, the mode in which it should operate. Supported values are `off`, `active`, or `passive`.

`-T` *time*, `--lacp-timer=`*time*
Specifies the LACP timer value. The supported values are `short` or `long`.

`-u` *address*, `--unicast=`*address*
Specifies a fixed unicast hardware address to be used for the aggregation. If this option is not specified, then an address is automatically chosen from the set of addresses of the component devices.

(Note that modification of the fixed unicast hardware address will override any previously defined `mac-address` link property defined for the aggregation. See "General Link Properties".)

`dladm delete-aggr` [`-t`] [`-R` *root-dir*] *aggr-link*
Deletes the specified aggregation.

`-t`, `--temporary`
Specifies that the deletion is temporary. Temporary deletions last until the next reboot.

`-R` *root-dir*, `--root-dir=`*root-dir*
See "Options," above.

`dladm add-aggr` [`-t`] [`-R` *root-dir*] `-l` *ether-link1* [`--link=`*ether-link2*...] *aggr-link*
Adds links to the specified aggregation.

`-l` *ether-link*, `--link=`*ether-link*
Specifies an Ethernet link to add to the aggregation. Multiple links can be added by supplying multiple `-l` options.

`-t`, `--temporary`
Specifies that the additions are temporary. Temporary additions last until the next reboot.

`-R` *root-dir*, `--root-dir=`*root-dir*
See "Options," above.

`dladm remove-aggr` [`-t`] [`-R` *root-dir*] `-l` *ether-link1* [`--l=`*ether-link2*...] *aggr-link*
Removes links from the specified aggregation.

`-l` *ether-link*, `--link=`*ether-link*
Specifies an Ethernet link to remove from the aggregation. Multiple links can be added by supplying multiple `-l` options.

`-t`, `--temporary`
Specifies that the removals are temporary. Temporary removal last until the next reboot.

`-R` *root-dir*, `--root-dir=`*root-dir*
See "Options," above.

`dladm show-aggr` [`-PLxZ`] [`-s` [`-i` *interval*]] [[`-p`] `-o` *field*[,...]] [`-z` *zone*[,...]] [*aggr-link*]
Show aggregation configuration (the default) or LACP information either for all aggregations or for the specified aggregation.

By default (with no options), the following fields can be displayed:

`LINK`
The name of the aggregation link.

`MODE`
The aggregation mode, either `trunk` or `dlmp`.

`POLICY`
The LACP policy of the aggregation. See the `create-aggr -P` option for a description of the possible values.

`ADDRPOLICY`
Either `auto`, if the aggregation is configured to automatically configure its unicast MAC address (the default if the `-u` option was not used to create or modify the aggregation), or `fixed`, if `-u` was used to set a fixed MAC address.

`LACPACTIVITY`
The LACP mode of the aggregation. Possible values are `off`, `active`, or `passive`, as set by the `-l` option to `create-aggr` or `modify-aggr`.

LACPTIMER
The LACP timer value of the aggregation as set by the -T option of create-aggr or modify-aggr.

The following field is not part of the default output, but can be queried using -o.

FLAGS
A set of state flags associated with the aggregation. The only possible flag is f, which is displayed if the administrator forced the creation the aggregation using the -f option to create-aggr. Other flags might be defined in the future.

The show-aggr command accepts the following options:

-L, --lacp
Displays detailed LACP information for the aggregation link and each underlying port. Most of the state information displayed by this option is defined by IEEE 802.3. With this option, the following fields can be displayed:

LINK
The name of the aggregation link.

PORT
The name of one of the underlying aggregation ports.

AGGREGATABLE
Whether the port can be added to the aggregation.

SYNC
If yes, the system considers the port to be synchronized and part of the aggregation.

COLL
If yes, collection of incoming frames is enabled on the associated port.

DIST
If yes, distribution of outgoing frames is enabled on the associated port.

DEFAULTED
If yes, the port is using defaulted partner information (that is, has not received LACP data from the LACP partner).

EXPIRED
If yes, the receive state of the port is in the EXPIRED state.

-x, --extended
Display additional aggregation information including detailed information on each underlying port. With -x, the following fields can be displayed:

LINK
The name of the aggregation link.

PORT
The name of one of the underlying aggregation ports.

`SPEED`
The speed of the link or port in megabits per second.

`DUPLEX`
The full/half duplex status of the link or port is displayed if the link state is `up`. The duplex status is displayed as `unknown` in all other cases.

`STATE`
The link state. This can be `up`, `down`, or `unknown`.

`ADDRESS`
The MAC address of the link or port.

`PORTSTATE`
This indicates whether the individual aggregation port is in the `standby` or `attached` state.

`-o` *field*[,...], `--output=`*field*[,...]
A case-insensitive, comma-separated list of output fields to display. The field name must be one of the fields listed above, or the special value `all`, to display all fields. The fields applicable to the `-o` option are limited to those listed under each output mode. For example, if using `-L`, only the fields listed under `-L`, above, can be used with `-o`.

`-p`, `--parseable`
Display using a stable machine-parseable format. The `-o` option is required with `-p`. See "Parseable Output Format", below.

`-P`, `--persistent`
Display the persistent aggregation configuration rather than the state of the running system.

`-s`, `--statistics`
Displays aggregation statistics. This option is made obsolete by `dlstat(1M)`.

`-i` *interval*, `--interval=`*interval*
Used with the `-s` option to specify an interval, in seconds, at which statistics should be displayed. This option is made obsolete by `dlstat(1M)`.

`-Z`
Display `ZONE` column in the output.

`-z` *zone*[,...]
See description of `-z` option under `dladm show-link`, above.

`dladm create-bridge` [`-P` *protect*] [`-R` *root-dir*] [`-p` *priority*] [`-m` *max-age*] [`-h` *hello-time*] [`-d` *forward-delay*] [`-f` *force-protocol*] [`-l` *link*...] *bridge-name*
Create an 802.1D bridge instance and optionally assign one or more network links to the new bridge. By default, no bridge instances are present on the system.

In order to bridge between links, you must create at least one bridge instance. Each bridge instance is separate, and there is no forwarding connection between bridges.

Note that the bridge-related subcommands, `create-bridge` among them, require installation of the `pkg://solaris/network/bridging` package.

`-P` *protect*, `--protect=`*protect*
Specifies a protection method. The defined protection methods are `stp` for the Spanning Tree Protocol and trill for `TRILL`, which is used on RBridges. The default value is `stp`.

`-R` *root-dir*, `--root-dir=`*root-dir*
See "Options," above.

`-p` *priority*, `--priority=`*priority*
Specifies the Bridge Priority. This sets the IEEE STP priority value for determining the root bridge node in the network. The default value is `32768`. Valid values are `0` (highest priority) to `61440` (lowest priority), in increments of 4096.

If a value not evenly divisible by 4096 is used, the system silently rounds downward to the next lower value that is divisible by 4096.

`-m` *max-age*, `--max-age=`*max-age*
Specifies the maximum age for configuration information in seconds. This sets the STP Bridge Max Age parameter. This value is used for all nodes in the network if this node is the root bridge. Bridge link information older than this time is discarded. It defaults to 20 seconds. Valid values are from 6 to 40 seconds. See the `-d` *forward-delay* parameter for additional constraints.

`-h` *hello-time*, `--hello-time=`*hello-time*
Specifies the STP Bridge Hello Time parameter. When this node is the root node, it sends Configuration BPDUs at this interval throughout the network. The default value is 2 seconds. Valid values are from 1 to 10 seconds. See the `-d` *forward-delay* parameter for additional constraints.

`-d` *forward-delay*, `--forward-delay=`*forward-delay*
Specifies the STP Bridge Forward Delay parameter. When this node is the root node, then all bridges in the network use this timer to sequence the link states when a port is enabled. The default value is 15 seconds. Valid values are from 4 to 30 seconds.

Bridges must obey the following two constraints:

`2 * (`*forward-delay*` - 1.0) >=` *max-age*

max-age` >= 2 * (`*hello-time*` + 1.0)`

Any parameter setting that would violate those constraints is treated as an error and causes the command to fail with a diagnostic message. The message provides valid alternatives to the supplied values.

`-f` *force-protocol*, `--force-protocol=`*force-protocol*
Specifies the MSTP forced maximum supported protocol. The default value is 3. Valid values are non-negative integers. The current implementation does not support RSTP

or MSTP, so this currently has no effect. However, to prevent MSTP from being used in the future, the parameter may be set to `0` for STP only or `2` for STP and RSTP.

`-l` *link*, `--link=`*link*
Specifies one or more links to add to the newly-created bridge. This is similar to creating the bridge and then adding one or more links, as with the `add-bridge` subcommand. However, if any of the links cannot be added, the entire command fails, and the new bridge itself is not created. To add multiple links on the same command line, repeat this option for each link. You are permitted to create bridges without links. For more information about link assignments, see the `add-bridge` subcommand.

Bridge creation and link assignment require the `PRIV_SYS_DL_CONFIG` privilege. Bridge creation might fail if the optional bridging feature is not installed on the system.

`dladm modify-bridge` [`-P` *protect*] [`-R` *root-dir*] [`-p` *priority*] [`-m` *max-age*] [`-h` *hello-time*] [`-d` *forward-delay*] [`-f` *force-protocol*] [`-l` *link*...] *bridge-name*
Modify the operational parameters of an existing bridge. The options are the same as for the `create-bridge` subcommand, except that the `-l` option is not permitted. To add links to an existing bridge, use the `add-bridge` subcommand.

Bridge parameter modification requires the `PRIV_SYS_DL_CONFIG` privilege.

`dladm delete-bridge` [`-R` *root-dir*] *bridge-name*
Delete a bridge instance. The bridge being deleted must not have any attached links. Use the `remove-bridge` subcommand to deactivate links before deleting a bridge.

Bridge deletion requires the `PRIV_SYS_DL_CONFIG` privilege.

The `-R` (`--root-dir`) option is the same as for the `create-bridge` subcommand.

`dladm add-bridge` [`-R` *root-dir*] `-l` *link* [`-l` *link*...] *bridge-name*
Add one or more links to an existing bridge. If multiple links are specified, and adding any one of them results in an error, the command fails and no changes are made to the system.

Link addition to a bridge requires the `PRIV_SYS_DL_CONFIG` privilege.

A link may be a member of at most one bridge. An error occurs when you attempt to add a link that already belongs to another bridge. To move a link from one bridge instance to another, remove it from the current bridge before adding it to a new one.

The links assigned to a bridge must not also be VLANs, VNICs, or tunnels. Only physical Ethernet datalinks, aggregation datalinks, and Ethernet stubs are permitted to be assigned to a bridge.

Links assigned to a bridge must all have the same MTU. This is checked when the link is assigned. The link is added to the bridge in a deactivated form if it is not the first link on the bridge and it has a differing MTU.

Note that systems using bridging should not set the `eeprom(1M)` `local-mac-address?` variable to false.

The options are the same as for the `create-bridge` subcommand.

`dladm remove-bridge` [`-R` *root-dir*] `-l` *link* [`-l` *link*...] *bridge-name*
Remove one or more links from a bridge instance. If multiple links are specified, and removing any one of them would result in an error, the command fails and none are removed.

Link removal from a bridge requires the `PRIV_SYS_DL_CONFIG` privilege.

The options are the same as for the `create-bridge` subcommand.

`dladm show-bridge` [`-flt`] [`-s` [`-i` *interval*]] [[`-p`] `-o` *field*,...] [*bridge-name*]
Show the running status and configuration of bridges, their attached links, learned forwarding entries, and `TRILL` nickname databases. When showing overall bridge status and configuration, the bridge name can be omitted to show all bridges. The other forms require a specified bridge.

The show-bridge subcommand accepts the following options:

`-i` *interval*, `--interval=`*interval*
Used with the `-s` option to specify an interval, in seconds, at which statistics should be displayed. If this option is not specified, statistics will be displayed only once.

`-s`, `--statistics`
Display statistics for the specified bridges or for a given bridge's attached links. This option cannot be used with the `-f` and `-t` options.

`-p`, `--parseable`
Display using a stable machine-parsable format. See "Parsable Output Format," below.

`-o` *field*[,...], `--output=`*field*[,...]
A case-insensitive, comma-separated list of output fields to display. The field names are described below. The special value all displays all fields. Each set of fields has its own default set to display when `-o` is not specified.

By default, the `show-bridge` subcommand shows bridge configuration. The following fields can be shown:

`BRIDGE`
The name of the bridge.

`ADDRESS`
The Bridge Unique Identifier value (MAC address).

`PRIORITY`
Configured priority value; set by `-p` with `create-bridge` and `modify-bridge`.

`BMAXAGE`
Configured bridge maximum age; set by `-m` with `create-bridge` and `modify-bridge`.

`BHELLOTIME`
Configured bridge hello time; set by `-h` with `create-bridge` and `modify-bridge`.

`BFWDDELAY`
Configured forwarding delay; set by `-d` with `create-bridge` and `modify-bridge`.

`FORCEPROTO`
Configured forced maximum protocol; set by `-f` with `create-bridge` and `modify-bridge`.

`TCTIME`
Time, in seconds, since last topology change.

`TCCOUNT`
Count of the number of topology changes.

`TCHANGE`
This indicates that a topology change was detected.

`DESROOT`
Bridge Identifier of the root node.

`ROOTCOST`
Cost of the path to the root node.

`ROOTPORT`
Port number used to reach the root node.

`MAXAGE`
Maximum age value from the root node.

`HELLOTIME`
Hello time value from the root node.

`FWDDELAY`
Forward delay value from the root node.

`HOLDTIME`
Minimum BPDU interval.

By default, when the `-o` option is not specified, only the `BRIDGE`, `ADDRESS`, `PRIORITY`, and `DESROOT` fields are shown.

When the `-s` option is specified, the `show-bridge` subcommand shows bridge statistics. The following fields can be shown:

`BRIDGE`
Bridge name.

`DROPS`
Number of packets dropped due to resource problems.

FORWARDS
Number of packets forwarded from one link to another.

MBCAST
Number of multicast and broadcast packets handled by the bridge.

RECV
Number of packets received on all attached links.

SENT
Number of packets sent on all attached links.

UNKNOWN
Number of packets handled that have an unknown destination. Such packets are sent to all links.

By default, when the `-o` option is not specified, only the `BRIDGE`, `DROPS`, and `FORWARDS` fields are shown.

The `show-bridge` subcommand also accepts the following options:

`-l`, `--link`
Displays link-related status and statistics information for all links attached to a single bridge instance. By using this option and without the `-s` option, the following fields can be displayed for each link:

`LINK`
The link name.

`INDEX`
Port (link) index number on the bridge.

`STATE`
State of the link. The state can be `disabled`, `discarding`, `learning`, `forwarding`, `non-stp`, or `bad-mtu`.

`UPTIME`
Number of seconds since the last reset or initialization.

`OPERCOST`
Actual cost in use (1-65535).

`OPERP2P`
This indicates whether point-to-point (`P2P`) mode been detected.

`OPEREDGE`
This indicates whether edge mode has been detected.

`DESROOT`
The Root Bridge Identifier that has been seen on this port.

`DESCOST`
Path cost to the network root node through the designated port.

`DESBRIDGE`
Bridge Identifier for this port.

`DESPORT`
The ID and priority of the port used to transmit configuration messages for this port.

`TCACK`
This indicates whether Topology Change Acknowledge has been seen.

When the `-l` option is specified without the `-o` option, only the `LINK`, `STATE`, `UPTIME`, and `DESROOT` fields are shown.

When the `-l` option is specified, the `-s` option can be used to display the following fields for each link:

`LINK`
Link name.

`CFGBPDU`
Number of configuration BPDUs received.

`TCNBPDU`
Number of topology change BPDUs received.

`RSTPBPDU`
Number of Rapid Spanning Tree BPDUs received.

`TXBPDU`
Number of BPDUs transmitted.

`DROPS`
Number of packets dropped due to resource problems.

`RECV`
Number of packets received by the bridge.

`XMIT`
Number of packets sent by the bridge.

When the `-o` option is not specified, only the `LINK`, `DROPS`, `RECV`, and `XMIT` fields are shown.

`-f`, `--forwarding`
Displays forwarding entries for a single bridge instance. With this option, the following fields can be shown for each forwarding entry:

`DEST`
Destination MAC address.

AGE
Age of entry in seconds and milliseconds. Omitted for local entries.

FLAGS
The L (local) flag is shown if the MAC address belongs to an attached link or to a VNIC on one of the attached links.

OUTPUT
For local entries, this is the name of the attached link that has the MAC address. Otherwise, for bridges that use Spanning Tree Protocol, this is the output interface name. For RBridges, this is the output TRILL nickname.

When the -o option is not specified, the DEST, AGE, FLAGS, and OUTPUT fields are shown.

-t, --trill
Displays TRILL nickname entries for a single bridge instance. With this option, the following fields can be shown for each TRILL nickname entry:

NICK
TRILL nickname for this RBridge, which is a number from 1 to 65535.

FLAGS
The L flag is shown if the nickname identifies the local system.

LINK
Link name for output when sending messages to this RBridge.

NEXTHOP
MAC address of the next hop RBridge that is used to reach the RBridge with this nickname.

When the -o option is not specified, the NICK, FLAGS, LINK, and NEXTHOP fields are shown.

dladm create-vlan [-ft] [-R *root-dir*] -l *ether-link* -v *vid* [*vlan-link*]
Create a tagged VLAN link with an ID of *vid* over Ethernet link *ether-link*. The name of the VLAN link can be specified as *vlan-link*. If the name is not specified, a name will be automatically generated (assuming that *ether-link* is *namePPA*) as:

<*name*><1000 * *vlan-tag* + *PPA*>

For example, if *ether-link* is bge1 and *vid* is 2, the name generated is bge2001.

-f, -—force
Force the creation of the VLAN link. Some devices do not allow frame sizes large enough to include a VLAN header. When creating a VLAN link over such a device, the -f option is needed, and the MTU of the IP interfaces on the resulting VLAN must be set to 1496 instead of 1500.

-l *ether-link*
Specifies Ethernet link over which VLAN is created.

`-t`, `--temporary`
Specifies that the VLAN link is temporary. Temporary VLAN links last until the next reboot.

`-R` *root-dir*, `--root-dir=`*root-dir*
See "Options," above.

`dladm modify-vlan` [`-t`] [`-R` *root-dir*] [`-l` *ether-link*] [`-v` *vid* [`-f`]] {*vlan-link*,[*vlan-link*,...] | `-L` *source-ether-link*}
Modifies the underlying link and/or the VLAN-ID of the specified VLAN link(s). The VLAN link(s) can be specified as a comma-delimited list or as `-L` *source-ether-link* to indicate "all VLANs on *source-ether-link*".

`-t`, `--temporary`
Specifies that the VLAN modification is temporary.

`-R` *root-dir*, `--root-dir=`*root-dir*
See "Options," above.

`-l` *ether-link*
Specifies the Ethernet link to which to move the VLAN(s). The Ethernet link must be different from the current one the VLAN(s) is or are using.

`-v` *vid* [`-f`]
Specifies the VLAN-ID to be used. This option can be used only if a single VLAN link is specified. The purpose of the `-f` option is the same as in `create-vlan`, above.

`dladm delete-vlan` [`-t`] [`-R` *root-dir*] *vlan-link*
Delete the VLAN link specified.

The `delete-vlan`subcommand accepts the following options:

`-t`, `--temporary`
Specifies that the deletion is temporary. Temporary deletions last until the next reboot.

`-R` *root-dir*, `--root-dir=`*root-dir*
See "Options," above.

`dladm show-vlan` [`-PZ`] [[`-p`] `-o` *field*[,...]] [`-z` *zone*[,...]] [*vlan-link*]
Display VLAN configuration for all VLAN links or for the specified VLAN link.

The `show-vlan` subcommand accepts the following options:

`-o` *field*[,...], `--output=`*field*[,...]
A case-insensitive, comma-separated list of output fields to display. The field name must be one of the fields listed below, or the special value `all`, to display all fields. For each VLAN link, the following fields can be displayed:

`LINK`
The name of the VLAN link.

`VID`
The ID associated with the VLAN.

`OVER`
The name of the physical link over which this VLAN is configured.

`FLAGS`
A set of flags associated with the VLAN link. Possible flags are:

`f`
The VLAN was created using the `-f` option to `create-vlan`.

`i`
The VLAN was implicitly created when the DLPI link was opened. These VLAN links are automatically deleted on last close of the DLPI link (for example, when the IP interface associated with the VLAN link is unplumbed).

Additional flags might be defined in the future.

`-p`, `--parseable`
Display using a stable machine-parseable format. The `-o` option is required with `-p`. See "Parseable Output Format", below.

`-P`, `--persistent`
Display the persistent VLAN configuration rather than the state of the running system.

`-Z`
Display `ZONE` column in the output.

`-z` *zone*[,...]
See description of `-z` option under `dladm show-link`, above.

`dladm scan-wifi` [[`-p`] `-o` *field*[,...]] [*wifi-link*]
Scans for WiFi networks, either on all WiFi links, or just on the specified *wifi-link*.

By default, currently all fields but `BSSTYPE` are displayed.

`-o` *field*[,...], `--output=`*field*[,...]
A case-insensitive, comma-separated list of output fields to display. The field name must be one of the fields listed below, or the special value `all` to display all fields. For each WiFi network found, the following fields can be displayed:

`LINK`
The name of the link the WiFi network is on.

`ESSID`
The ESSID (name) of the WiFi network.

`BSSID`
Either the hardware address of the WiFi network's Access Point (for BSS networks), or the WiFi network's randomly generated unique token (for IBSS networks).

SEC
Either none for a WiFi network that uses no security, wep for a WiFi network that requires WEP (Wired Equivalent Privacy), or wpa for a WiFi network that requires WPA (Wi-Fi Protected Access).

MODE
The supported connection modes: one or more of a, b, g, or n.

STRENGTH
The strength of the signal: one of excellent, very good, good, weak, or very weak.

SPEED
The maximum speed of the WiFi network, in megabits per second.

BSSTYPE
Either bss for BSS (infrastructure) networks, or ibss for IBSS (ad-hoc) networks.

`-p`, `--parseable`
Display using a stable machine-parseable format. The `-o` option is required with `-p`. See "Parseable Output Format", below.

`dladm connect-wifi` [`-e` *essid*] [`-i` *bssid*] [`-k` *key*,...] [`-s none` | `wep` | `wpa`] [`-a open`|`shared`] [`-b bss`|`ibss`] [`-c`] [`-m a`|`b`|`g`|`n`] [`-T` *time*] [*wifi-link*]
Connects to a WiFi network. This consists of four steps: *discovery*, *filtration*, *prioritization*, and *association*. However, to enable connections to non-broadcast WiFi networks and to improve performance, if a BSSID or ESSID is specified using the `-e` or `-i` options, then the first three steps are skipped and `connect-wifi` immediately attempts to associate with a BSSID or ESSID that matches the rest of the provided parameters. If this association fails, but there is a possibility that other networks matching the specified criteria exist, then the traditional discovery process begins as specified below.

The discovery step finds all available WiFi networks on the specified WiFi link, which must not yet be connected. For administrative convenience, if there is only one WiFi link on the system, *wifi-link* can be omitted.

Once discovery is complete, the list of networks is filtered according to the value of the following options:

`-e` *essid,* `--essid=`*essid*
Networks that do not have the same *essid* are filtered out.

`-b bss`|`ibss`, `--bsstype=bss`|`ibss`
Networks that do not have the same `bsstype` are filtered out.

`-m a`|`b`|`g`, `--mode=a`|`b`|`g`|`n`
Networks not appropriate for the specified 802.11 mode are filtered out.

`-k` *key,...*, `--key=`*key, ...*
Use the specified `secobj` named by the key to connect to the network. Networks not appropriate for the specified keys are filtered out.

`-s` `none|wep|wpa`, `--sec=none|wep|wpa`
Networks not appropriate for the specified security mode are filtered out.

Next, the remaining networks are prioritized, first by signal strength, and then by maximum speed. Finally, an attempt is made to associate with each network in the list, in order, until one succeeds or no networks remain.

In addition to the options described above, the following options also control the behavior of `connect-wifi`:

`-a` `open|shared`, `--auth=open|shared`
Connect using the specified authentication mode. By default, `open` and `shared` are tried in order.

`-c`, `--create-ibss`
Used with `-b ibss` to create a new ad-hoc network if one matching the specified ESSID cannot be found. If no ESSID is specified, then `-c -b ibss` always triggers the creation of a new ad-hoc network.

`-T` *time*, `--timeout=`*time*
Specifies the number of seconds to wait for association to succeed. If *time* is `forever`, then the associate will wait indefinitely. The current default is ten seconds, but this might change in the future. Timeouts shorter than the default might not succeed reliably.

`-k` *key,...*, `--key=`*key,...*
In addition to the filtering previously described, the specified keys will be used to secure the association. The security mode to use will be based on the key class; if a security mode was explicitly specified, it must be compatible with the key class. All keys must be of the same class.

For security modes that support multiple key slots, the slot to place the key will be specified by a colon followed by an index. Therefore, `-k mykey:3` places `mykey` in slot 3. By default, slot 1 is assumed. For security modes that support multiple keys, a comma-separated list can be specified, with the first key being the active key.

`dladm disconnect-wifi` [`-a`] [*wifi-link*]
Disconnect from one or more WiFi networks. If *wifi-link* specifies a connected WiFi link, then it is disconnected. For administrative convenience, if only one WiFi link is connected, *wifi-link* can be omitted.

`-a`, `--all-links`
Disconnects from all connected links. This is primarily intended for use by scripts.

`dladm show-wifi` [`-Z`] [[`-p`] `-o` *field*,...] [`-z` *zone*[,...]] [*wifi-link*]
Shows WiFi configuration information either for all WiFi links or for the specified link *wifi-link*.

`-o` *field,...*, `--output=`*field*

A case-insensitive, comma-separated list of output fields to display. The field name must be one of the fields listed below, or the special value `all`, to display all fields. For each WiFi link, the following fields can be displayed:

`LINK`
The name of the link being displayed.

`STATUS`
Either `connected` if the link is connected, or `disconnected` if it is not connected. If the link is disconnected, all remaining fields have the value `--`.

`ESSID`
The ESSID (name) of the connected WiFi network.

`BSSID`
Either the hardware address of the WiFi network's Access Point (for BSS networks), or the WiFi network's randomly generated unique token (for IBSS networks).

`SEC`
Either `none` for a WiFi network that uses no security, `wep` for a WiFi network that requires WEP, or `wpa` for a WiFi network that requires WPA.

`MODE`
The supported connection modes: one or more of `a`, `b`, `g`, or `n`.

`STRENGTH`
The connection strength: one of `excellent`, `very good`, `good`, `weak`, or `very weak`.

`SPEED`
The connection speed, in megabits per second.

`AUTH`
Either `open` or `shared` (see `connect-wifi`).

`BSSTYPE`
Either `bss` for BSS (infrastructure) networks, or `ibss` for IBSS (ad-hoc) networks.

By default, currently all fields but `AUTH`, `BSSID`, `BSSTYPE` are displayed.

`-p`, `--parseable`

Displays using a stable machine-parseable format. The `-o` option is required with `-p`. See "Parseable Output Format", below.

`-Z`

Display `ZONE` column in the output.

`-z` *zone*[,...]

See description of `-z` option under `dladm show-link`, above.

`dladm show-ether` [`-xZ`] [[`-p`] `-o` *field*,...] [`-z` *zone*[,...]] [`-P` *protocol*] [*ether-link*]

Shows state information either for all physical Ethernet links or for a specified physical Ethernet link.

The `show-ether` subcommand accepts the following options:

`-o` *field*,..., `--output=`*field*

A case-insensitive, comma-separated list of output fields to display. The field name must be one of the fields listed below, or the special value `all` to display all fields. For each link, the following fields can be displayed:

`LINK`

The name of the link being displayed.

`PTYPE`

Parameter type, where `current` indicates the negotiated state of the link, `capable` indicates capabilities supported by the device, `adv` indicates the advertised capabilities, and `peeradv` indicates the capabilities advertised by the link-partner.

`STATE`

The state of the link.

`AUTO`

A `yes`/`no` value indicating whether auto-negotiation is advertised.

`SPEED-DUPLEX`

Combinations of speed and duplex values available. The units of speed are encoded with a trailing suffix of `G` (Gigabits/s) or `M` (Mb/s). Duplex values are encoded as `f` (full-duplex) or `h` (half-duplex).

`PAUSE`

Flow control information. Can be `no`, indicating no flow control is available; `tx`, indicating that the end-point can transmit pause frames, but ignores any received pause frames; `rx`, indicating that the end-point receives and acts upon received pause frames; or `bi`, indicating bi-directional flow-control.

`REM_FAULT`

Fault detection information. Valid values are `none` or `fault`.

By default, all fields except `REM_FAULT` are displayed for the "current" `PTYPE`.

`-p`, `--parseable`

Displays using a stable machine-parseable format. The `-o` option is required with `-p`. See "Parseable Output Format", below.

`-P` *protocol*

Displays information about supported Ethernet protocols. Supported protocols include `vdp`, the VSI Discovery and Configuration protocol, and `ecp`, Edge Control Protocol.

VDP information is specific to a VNIC. Thus, if the link argument is a `phys-link`, VDP information for all of the VNIC over the `phys-link` is displayed.

ECP information is specific to a `phys-link`.

For VDP, following information is displayed:

`VSI`
The name of the Virtual Station Interface (VSI) or VNIC.

`LINK`
The name of the physical link over which this VNIC is configured.

`VSI-STATE`
The state of the VDP protocol state machine for the VNIC. Supported states include `ASSOC`, `DEASSOC`, or `TIMEDOUT`.

`VSIID`
The identifier for the VSI or VNIC. This identifier is used by the bridge to associate properties with VNICs. Supported format for the VSIID is the MAC address. Thus, the VSIID for a VNIC is its MAC address.

`VSI-TYPEID`
This is VSI Type ID and Version associated with a VNIC and is of the form VSI Type ID/Version. The VSI Type identifies the properties associated with the VNIC.

`CMD-PENDING`
The VDP command that is currently in progress. Supported commands are: `ASSOC`, `DEASSOC`. The `ASSOC` command requests the bridge to associate properties with a VSI (identified by the VSIID), whereas the `DEASSOC` requests the bridge to disassociate the properties from a given VSIID.

`FILTER-INFO`
The information used by the switch to filter packets for a given VNIC. Supported format for Filter Info includes the MAC/VLAN ID combination. Thus, the `FilterInfo` for a VNIC is its MAC address and VLAN ID, if any.

`KEEPALIVE-INTERVAL`
The inteval (in seconds) for Keep Alive messages to be transmitted for existing associations. The default is 11.6 secs.

`RESP-TIMEOUT`
The time (in seconds) to wait for a response from the bridge before timing out a request.

For ECP, following information is displayed:

`LINK`
The name of the physical link for the ECP instance.

`MAC-RETRIES`
The maximum number of transmission retries without receiving an acknowledgement from the peer.

`TIMEOUT`
The interval of time (in milliseconds) to wait for an acknowledgment from the peer.

`-x`, `--extended`
Extended output is displayed for `PTYPE` values of `current`, `capable`, `adv` and `peeradv`.

`-Z`
Display `ZONE` column in the output.

`-z` *zone*[,...]
See description of `-z` option under `dladm show-link`, above.

`dladm set-linkprop` [`-t`] [`-R` *root-dir*] `-p` *prop=value*[,...] *link*
Sets the values of one or more properties on the link specified. The list of properties and their possible values depend on the link type, the network device driver, and networking hardware. These properties can be retrieved using `show-linkprop`.

`-t`, `--temporary`
Specifies that the changes are temporary. Temporary changes last until the next reboot.

`-R` *root-dir*, `--root-dir=`*root-dir*
See "Options," above.

`-p` *prop=value*[,...], `--prop` *prop=value*[,...]
A comma-separated list of properties to set to the specified values.

Note that when the persistent value is set, the temporary value changes to the same value.

`dladm reset-linkprop` [`-t`] [`-R` *root-dir*] [`-p` *prop*,...] *link*
Resets one or more properties to their values on the link specified. Properties are reset to the values they had at startup. If no properties are specified, all properties are reset. See `show-linkprop` for a description of properties.

`-t`, `--temporary`
Specifies that the resets are temporary. Values are reset to default values. Temporary resets last until the next reboot.

`-R` *root-dir*, `--root-dir=`*root-dir*
See "Options," above.

`-p` *prop, ...*, `-–prop=`*prop, ...*
A comma-separated list of properties to reset.

Note that when the persistent value is reset, the temporary value changes to the same value.

`dladm show-linkprop` [`-PZ`] [[`-c`] `-o` *field*[,...]][`-p` *prop*[,...]] [`-z` *zone*[,...]] [*link*]

Show the current or persistent values of one or more properties, either for all datalinks or for the specified link. By default, current values are shown. If no properties are specified, all available link properties are displayed. For each property, the following fields are displayed:

`-o` *field*[,...], `--output=`*field*

A case-insensitive, comma-separated list of output fields to display. The field name must be one of the fields listed below, or the special value `all` to display all fields. For each link, the following fields can be displayed:

`LINK`
The name of the datalink.

`PROPERTY`
The name of the property.

`PERM`
The read/write permissions of the property. The value shown is one of `ro` or `rw`.

`VALUE`
The current (or persistent) property value. If the value is not set, it is shown as `--`. If it is unknown, the value is shown as `?`. Persistent values that are not set or have been reset will be shown as `--` and will use the system `DEFAULT` value (if any).

`DEFAULT`
The default value of the property. If the property has no default value, `--` is shown.

`POSSIBLE`
A comma-separated list of the values the property can have. If the values span a numeric range, *min* - *max* might be shown as shorthand. If the possible values are unknown or unbounded, `--` is shown.

The list of properties depends on the link type and network device driver, and the available values for a given property further depends on the underlying network hardware and its state. General link properties are documented in the "General Link Properties" section. However, link properties that begin with "_" (underbar) are specific to a given link or its underlying network device and subject to change or removal. See the appropriate network device driver man page for details.

`-c`, `--parseable`

Display using a stable machine-parseable format. The `-o` option is required with this option. See "Parseable Output Format", below.

`-P`, `--persistent`

Display persistent link property information

`-p` *prop, ...*, `--prop=`*prop, ...*

A comma-separated list of properties to show. See the sections on link properties following subcommand descriptions.

`-Z`
Display `ZONE` column in the output.

`-z` *zone*[,...]
See description of `-z` option under `dladm show-link`, above.

`dladm create-secobj` [`-t`] [`-R` *root-dir*] [`-f` *file*] `-c` *class secobj*
Create a secure object named *secobj* in the specified *class* to be later used as a WEP or WPA key in connecting to an encrypted network. The value of the secure object can either be provided interactively or read from a file. The sequence of interactive prompts and the file format depends on the class of the secure object.

Currently, the classes `wep` and `wpa` are supported. The WEP (Wired Equivalent Privacy) key can be either 5 or 13 bytes long. It can be provided either as an ASCII or hexadecimal string -- thus, `12345` and `0x3132333435` are equivalent 5-byte keys (the `0x` prefix can be omitted). A file containing a WEP key must consist of a single line using either WEP key format. The WPA (Wi-Fi Protected Access) key must be provided as an ASCII string with a length between 8 and 63 bytes.

This subcommand is only usable by users or roles that belong to the "Network Link Security" RBAC profile.

`-c` *class*, `--class=`*class*
class can be `wep` or `wpa`. See preceding discussion.

`-t`, `--temporary`
Specifies that the creation is temporary. Temporary creation last until the next reboot.

`-R` *root-dir*, `--root-dir=`*root-dir*
See "Options," above.

`-f` *file*, `--file=`*file*
Specifies a file that should be used to obtain the secure object's value. The format of this file depends on the secure object class. See the `EXAMPLES` section for an example of using this option to set a WEP key.

`dladm delete-secobj` [`-t`] [`-R` *root-dir*] *secobj*[,...]
Delete one or more specified secure objects. This subcommand is only usable by users or roles that belong to the "Network Link Security" RBAC profile.

`-t`, `--temporary`
Specifies that the deletions are temporary. Temporary deletions last until the next reboot.

`-R` *root-dir*, `--root-dir=`*root-dir*
See "Options," above.

`dladm show-secobj` [`-P`] [[`-p`] `-o` *field*[,...]] [*secobj*,...]
Show current or persistent secure object information. If one or more secure objects are specified, then information for each is displayed. Otherwise, all current or persistent secure objects are displayed.

By default, current secure objects are displayed, which are all secure objects that have either been persistently created and not temporarily deleted, or temporarily created.

For security reasons, it is not possible to show the value of a secure object.

`-o` *field*[,...] , `--output=`*field*[,...]
A case-insensitive, comma-separated list of output fields to display. The field name must be one of the fields listed below. For displayed secure object, the following fields can be shown:

`OBJECT`
The name of the secure object.

`CLASS`
The class of the secure object.

`-p`, `--parseable`
Display using a stable machine-parseable format. The `-o` option is required with `-p`. See "Parseable Output Format", below.

`-P`, `--persistent`
Display persistent secure object information

`dladm create-vnic` [`-t`] `-l` *link* [`-R` *root-dir*] [`-m` *value* | auto | {factory [`-n` *slot-identifier*]} | {vrrp `-A` {inet | inet6} `-V` *vrid*} | {random [`-r` *prefix*]}] [`-v` *vlan-id*] [`-p` *prop*=*value*[,...]] *vnic-link*
Create a VNIC with name *vnic-link* over the specified link.

`-t`, `-–temporary`
Specifies that the VNIC is temporary. Temporary VNICs last until the next reboot.

`-R` *root-dir*, `-–root-dir=`*root-dir*
See "Options," above.

`-l` *link*, `-–link=`*link*
link can be a physical link or an `etherstub`.

`-m` *value* | *keyword*, `-–mac-address=`*value* | *keyword*
Sets the VNIC's MAC address based on the specified value or keyword. If *value* is not a keyword, it is interpreted as a unicast MAC address, which must be valid for the underlying NIC. A user-specified MAC address must be drawn from the ranges specified by the Globally Unique and Locally Administered types of MAC addresses.

The following special keywords can be used:

factory [`-n` *slot-identifier*],
factory [`--slot=`*slot-identifier*]
Assign a factory MAC address to the VNIC. When a factory MAC address is requested, `-m` can be combined with the `-n` option to specify a MAC address slot to be used. If `-n` is not specified, the system will choose the next available factory MAC address. The `-m` option of the `show-phys` subcommand can be used to display the list of factory MAC addresses, their slot identifiers, and their availability.

`random` [`-r` *prefix*],
random [`--mac-prefix=`*prefix*]
Assign a random MAC address to the VNIC. A default prefix consisting of a valid IEEE OUI with the local bit set will be used. That prefix can be overridden with the `-r` option.

`vrrp -A` {`inet` | `inet6`} `-V` *vrid*
Assign a VRRP virtual MAC address to the VNIC base on the specified address family and *vrid*.

`auto`
Try and use a factory MAC address first. If none is available, assign a random MAC address. `auto` is the default action if the `-m` option is not specified.

`-v` *vlan-id*
Enable VLAN tagging for this VNIC. The VLAN tag will have id *vlan-id*.

`-p` *prop=value*,..., `--prop` *prop=value*,...
A comma-separated list of properties to set to the specified values.

`dladm modify-vnic` [`-t`] [`-R` *root-dir*] [`-l` *link*] [`-m` *value* | auto | {factory [`-n` *slot-identifier*]} | {vrrp `-A` {inet | inet6} `-V` *vrid*} | {random [`-r` *prefix*]}] [`-v` *vlan-id*] {*vnic-link*,[*vnic-link*,...] | `-L` *source-link*}

Modifies the underlying link and/or the MAC address/VLAN-ID of the specified VNIC link(s). The VNIC link(s) can be specified as a comma-delimited list or as `-L` *source-link* to indicate "all VNICs on source-link".

`-t`, `--temporary`
Specifies that the VNIC modification is temporary.

`-R` *root-dir*, `--root-dir=`*root-dir*
See "Options," above.

`-l` *link*, `-link=`*link*
Specifies the link to which to move the VNIC(s). *link* can be of any link type supported by `create-vnic`. *link* must be different from the link the VNIC(s) are currently using. If the VNIC(s) are using a factory MAC address and `-m` is not specified, a new MAC address will be allocated on the target link, using the `-m` *auto* scheme, and assigned to the VNIC(s).

`-m` *value* | *keyword*, `--mac-address=`*value* | *keyword*
See `create-vnic`, above, for supported options. If multiple VNICs are specified, only the `auto`, `random`, and `factory` (without `-n`) address assignment schemes will be

supported.

`dladm delete-vnic` [`-t`] [`-R` *root-dir*] *vnic-link*
Deletes the specified VNIC.

`-t`, `--temporary`
Specifies that the deletion is temporary. Temporary deletions last until the next reboot.

`-R` *root-dir*, `--root-dir=`*root-dir*
See "Options," above.

`dladm show-vnic` [`-pPZ`] [`-s` [`-i` *interval*]] [`-o` *field*[,...]] [`-l` *link*] [`-z` *zone*[,...]] [*vnic-link*]
Show VNIC configuration information for all VNICs, all VNICs on a link, or only the specified *vnic-link*.

`-o` *field*[,...] , `--output=`*field*[,...]
A case-insensitive, comma-separated list of output fields to display. The field name must be one of the fields listed below, or the special value `all` to display all fields. By default (without `-o`), `show-vnic` displays all fields.

`LINK`
The name of the VNIC.

`OVER`
The name of the physical link over which this VNIC is configured.

`SPEED`
The maximum speed of the VNIC, in megabits per second.

`MACADDRESS`
MAC address of the VNIC.

`MACADDRTYPE`
MAC address type of the VNIC. `dladm` distinguishes among the following MAC address types:

`random`
A random address assigned to the VNIC.

`factory`
A factory MAC address used by the VNIC.

`-p`, `--parseable`
Display using a stable machine-parseable format. The `-o` option is required with `-p`. See "Parseable Output Format", below.

`-P`, `--persistent`
Display the persistent VNIC configuration.

`-s`, `--statistics`
Displays VNIC statistics. This option is made obsolete by `dlstat(1M)`.

`-i` *interval*, `--interval=`*interval*
Used with the `-s` option to specify an interval, in seconds, at which statistics should be displayed. This option is made obsolete by `dlstat(1M)`.

`-l` *link*, `--link=`*link*
Display information for all VNICs on the named link.

`-Z`
Display `ZONE` column in the output.

`-z` *zone*[,...]
See description of `-z` option under `dladm show-link`, above.

`dladm create-part` [`-t`] [`-f`] [`-R` *root-dir*] `-l` *ib-link* [`-p` *prop=value*[,..]] `-P` *pkey part-link*
Create an IP-over-IB link with the name *part-link* over the specified link. This subcommand is supported only on InfiniBand physical links.

`-f`, `--force`
Forces the creation of the partition link even if *pkey* is absent on the port, the multicast group is absent, or the port is down.

`-l` *ib-link*, `--link=`*ib-link*
IP-over-IB physical link name.

`-P`, `--pkey=`*pkey*
Partition key to be used for creating the partition link. *pkey* specified is always treated as hexadecimal, whether it has the `0x` prefix or not.

`-p` *prop=value*[,..]
`--prop` *prop=value*[,..]
A comma-separated list of properties to set to the specified values. Supported properties are given "General Link Properties" section below.

`-R` *root-dir*, `--root-dir=`*root-dir*
See "Options," above.

`-t`, `--temporary`
Specifies that the partition link creation is temporary. Temporary partition links last until the next reboot.

`dladm delete-part` [`-R` *root-dir*] *part-link*
Delete the specified partition link.

`-R` *root-dir*, `--root-dir=`*root-dir*
See "Options," above.

`-t`, `--temporary`
Specifies that the partition link deletion is temporary. Temporary deletion last until the next reboot.

`dladm show-part` [`-pP`] [`-l` *ib-link*] [`-o` *field*[,...]] [*part-link*]
Displays IB partition link information for all partition links, for all partitions on *ib-link*, or for only the specified *part-link*.

`-l` *ib-link*, `--link=`*ib-link*
Display information for all the partitions on the named link.

`-o` *field*[,...]
A case-insensitive, comma-separated list of output fields to display. The field name must be one of the fields listed below, or the special value `all` to display all fields. By default (without `-o`), `show-part` displays all fields.

`LINK`
The name of the partition link.

`PKEY`
`Pkey` associated with the partition link.

`OVER`
The name of the physical link over which this partition link is created.

`STATE`
Current state of the partition link. Possible values are `up`, `down`, or `unknown`.

`FLAGS`
A set of state flags used for creating the partition link. Possible values are:

`f`
Partition was created forcibly (without checking whether creating a partition were possible).

`t`
Partition link is temporary, lasting only until the next reboot.

`-P`, `--persistent`
Display the persistent IB partition link configuration.

`-p`, `--parseable`
Display using a stable machine-parseable format. The `-o` option is required with `-p`. See "Parseable Output Format", below.

`dladm show-ib` [`-pP`] [`-o` *field*[,...]] [*ib-link*]
Display IB physical link information on all or the specified IB links.

`-o` *field*[,...]
A case-insensitive, comma-separated list of output fields to display. The field name must be one of the fields listed below, or the special value `all` to display all fields. By default (without `-o`), `show-ib` displays all fields.

`LINK`
The name of the physical link.

`HCAGUID`
Globally unique identifier of the HCA.

`PORTGUID`
Globally unique identifier of the port.

`PORT`
Port number.

`STATE`
Current state of the physical link. Possible values are `up`, `down`, or `unknown`.

`PKEYS`
`Pkeys` available on the port associated with the IP-over-IB link specified in the `LINK` field.

`-P`, `--persistent`
Display the persistent IB physical link configuration.

`-p`, `--parseable`
Display using a stable machine-parseable format. The `-o` option is required with `-p`. See "Parseable Output Format", below.

`dladm create-etherstub` [`-t`] [`-R` *root-dir*] *etherstub*
Create an etherstub with the specified name.

`-t`, `-–temporary`
Specifies that the etherstub is temporary. Temporary etherstubs do not persist across reboots.

`-R` *root-dir*, `-–root-dir=`*root-dir*
See "Options," above.

VNICs can be created on top of etherstubs instead of physical NICs. As with physical NICs, such a creation causes the stack to implicitly create a virtual switch between the VNICs created on top of the same etherstub.

`dladm delete-etherstub` [`-t`] [`-R` *root-dir*] *etherstub*
Delete the specified etherstub.

`-t`, `-–temporary`
Specifies that the deletion is temporary. Temporary deletions last until the next reboot.

`-R` *root-dir*, `-–root-dir=`*root-dir*
See "Options," above.

`dladm show-etherstub` [`-Z`] [`-z` *zone*[,...]] [*etherstub*]
Show all configured etherstubs by default, or the specified etherstub if *etherstub* is specified.

`-Z`
Display `ZONE` column in the output.

`-z` *zone*[,...]
See description of `-z` option under `dladm show-link`, above.

`dladm create-iptun` [`-t`] [`-R` *root-dir*] `-T` *type* [`-a` {`local`|`remote`}=*addr*,...] *iptun-link*
Create an IP tunnel link named *iptun-link*. Such links can additionally be protected with IPsec using `ipsecconf(1M)`.

An IP tunnel is conceptually comprised of two parts: a virtual link between two or more IP nodes, and an IP interface above this link that allows the system to transmit and receive IP packets encapsulated by the underlying link. This subcommand creates a virtual link. The `ipadm(1M)` command is used to configure IP interfaces above the link.

`-t`, `--temporary`
Specifies that the IP tunnel link is temporary. Temporary tunnels last until the next reboot.

`-R` *root-dir*, `--root-dir`=*root-dir*
See "Options," above.

`-T` *type*, `--tunnel-type`=*type*
Specifies the type of tunnel to be created. The type must be one of the following:

`ipv4`
A point-to-point, IP-over-IP tunnel between two IPv4 nodes. This type of tunnel requires IPv4 source and destination addresses to function. IPv4 and IPv6 interfaces can be plumbed above such a tunnel to create IPv4-over-IPv4 and IPv6-over-IPv4 tunneling configurations.

`ipv6`
A point-to-point, IP-over-IP tunnel between two IPv6 nodes as defined in IETF RFC 2473. This type of tunnel requires IPv6 source and destination addresses to function. IPv4 and IPv6 interfaces can be plumbed above such a tunnel to create IPv4-over-IPv6 and IPv6-over-IPv6 tunneling configurations.

`6to4`
A 6to4, point-to-multipoint tunnel as defined in IETF RFC 3056. This type of tunnel requires an IPv4 source address to function. An IPv6 interface is plumbed on such a tunnel link to configure a 6to4 router.

`-a` {`local`|`remote`}=*addr*,...
`--address` {`local`|`remote`}=*addr*,...
Literal IP addresses or hostnames corresponding to the local or remote tunnel addresses. Either local or remote can be specified individually, or both can be specified separated by a comma (for example, `-a local=`*laddr*`,remote=`*raddr*).

`dladm modify-iptun` [`-t`] [`-R` *root-dir*] `-a` {`local`|`remote`}=*addr*,... *iptun-link*
Modify the parameters of the specified IP tunnel.

`-t`, `--temporary`
Specifies that the modification is temporary. Temporary modifications last until the next reboot.

`-R` *root-dir*, `--root-dir=`*root-dir*
See "Options," above.

`-a` {`local`|`remote`}=*addr*,...
`--address` {`local`|`remote`}=*addr*,...
Specify new local or remote addresses for the tunnel link. See `create-iptun` for a description.

`dladm delete-iptun` [`-t`] [`-R` *root-dir*] *iptun-link*
Delete the specified IP tunnel link.

`-t`, `--temporary`
Specifies that the deletion is temporary. Temporary deletions last until the next reboot.

`-R` *root-dir*, `--root-dir=`*root-dir*
See "Options," above.

`dladm show-iptun` [`-PZ`] [[`-p`] `-o` *field*[,...]] [`-z` *zone*[,...]] [*iptun-link*]
Show IP tunnel link configuration for a single IP tunnel or all IP tunnels.

`-P`, `--persistent`
Display the persistent IP tunnel configuration.

`-p`, `--parseable`
Display using a stable machine-parseable format. The -o option is required with -p. See "Parseable Output Format", below.

`-o` *field*[,...], `--output=`*field*[,...]
A case-insensitive, comma-separated list of output fields to display. The field name must be one of the fields listed below, or the special value `all`, to display all fields. By default (without `-o`), `show-iptun` displays all fields.

`LINK`
The name of the IP tunnel link.

`TYPE`
Type of tunnel as specified by the `-T` option of `create-iptun`.

`FLAGS`
A set of flags associated with the IP tunnel link. Possible flags are:

`s`
The IP tunnel link is protected by IPsec policy. To display the IPsec policy associated with the tunnel link, enter:

`# ipsecconf -ln -i` *tunnel-link*

See `ipsecconf(1M)` for more details on how to configure IPsec policy.

`i`
The IP tunnel link was implicitly created with `ipadm(1M)`, and will be automatically deleted when it is no longer referenced (that is, when the last IP interface over the tunnel is removed). See `ipadm(1M)` for details on implicit tunnel creation.

`LOCAL`
The local tunnel address.

`REMOTE`
The remote tunnel address.

`-Z`
Display `ZONE` column in the output.

`-z` *zone*[,...]
See description of `-z` option under `dladm show-link`, above.

`dladm show-usage` [`-a`] `-f` *filename* [`-p` *plotfile* `-F` *format*] [`-s` *time*] [`-e` *time*] [*link*]
This subcommand is made obsolete by the `dlstat(1M) show-link -h` command.

`help` [*subcommand-name*]
Displays all the supported `dladm` subcommands or usage for a given subcommand. If you invoke help for a specific subcommand, the command syntax is displayed, along with an example. Using `dladm` help without any argument displays all of the subcommands.

Parseable Output Format

Many `dladm` subcommands have an option that displays output in a machine-parseable format. The output format is one or more lines of colon (`:`) delimited fields. The fields displayed are specific to the subcommand used and are listed under the entry for the `-o` option for a given subcommand. Output includes only those fields requested by means of the `-o` option, in the order requested.

When you request multiple fields, any literal colon characters are escaped by a backslash (`\`) before being output. Similarly, literal backslash characters will also be escaped (`\\`). This escape format is parseable by using shell `read(1)` functions with the environment variable `IFS=:` (see `EXAMPLES`, below). Note that escaping is not done when you request only a single field.

General Link Properties

The following general link properties are supported:

`autopush`
Specifies the set of STREAMS modules to push on the stream associated with a link when its DLPI device is opened. It is a space-delimited list of modules.

The optional special character sequence `[anchor]` indicates that a STREAMS anchor should be placed on the stream at the module previously specified in the list. It is an error to specify more than one anchor or to have an anchor first in the list.

The `autopush` property is preferred over the more general `autopush(1M)` command.

`cos`
The 802.1p priority associated with the link. This property, when set, indicates the 802.1p priority on outbound packets on the link. The values range from 0 to 7. When this property is set, all the packets outbound on the link will have a VLAN tag with the priority field set to the property value. When this property is set on a physical NIC, only traffic for the primary client on that physical NIC will have priority set and not any other datalinks on the NIC. This property is only valid on Ethernet data link. The default cos is 0 for VLAN data links or when the underlyng device registers DCB capabilities, otherwise the default is not to add a VLAN tag.

`cpus`
Bind the processing of packets for a given data link to a processor or a set of processors. The value can be a comma-separated list of one or more processor ids or a range of ids. If the list consists of more than one processor, the processing will spread out to all the processors. Connection to processor affinity and packet ordering for any individual connection will be maintained.

The processor or set of processors are not exclusively reserved for the link. Only the kernel threads and interrupts associated with processing of the link are bound to the processor or the set of processors specified. In case it is desired that processors be dedicated to the link, `psrset(1M)` can be used to create a processor set and then specifying the processors from the processor set to bind the link to.

If the link was already bound to processor or set of processors due to a previous operation, the binding will be removed and the new set of processors will be used instead.

The default is no CPU binding, which is to say that the processing of packets is not bound to any specific processor or processor set.

Specification of the `cpus` property is not allowed on links with a `pool` link property.

`cpus-effective`
This read-only property displays the list of CPUs used for packet processing on the named data link.

If the `cpus` property has been set, `cpus-effective` will be the same.

If the `pool` property has been set, the `cpus-effective` will be selected from the pool designated by the administrator.

If neither the `pool` nor `cpus` property is set, the system will select the appropriate value for `cpus-effective`.

`etsbw-lcl`
This indicates the ETS bandwidth configured on the TX side for a link. This property can be configured on a data link only if the underlying physical NIC registers DCB capability and supports ETS. The value is a percentage of the physical NIC's bandwidth and the sum

of values of this property over all links on a physical NIC cannot exceed 100. Aggregation of physical NIC that register DCB capabilities is not supported currently, hence this property cannot be set on aggregations.

`etsbw-lcl-advice`
This indicates the ETS bandwidth (as a percentage) recommended by the remote end for this link. The value is obtained by means of LLDP.

`etsbw-lcl-effective`
This indicates the ETS bandwidth (as a percentage) that is effective on the TX side for the link. This could be the `etsbw-lcl` or `etsbw-lcl-advice` depending on LLDP negotiations.

`etsbw-rmt-effective`
This indicates the ETS bandwidth (in percentage) that is effective on the remote end for this link. The value is obtained by means of LLDP.

`rxfanout`
Allows you to specify the number of receive-side fanout threads.

Traffic received on a receive ring can be fanned out across multiple threads and processed in parallel. This is particularly useful when the system has large number of CPUs. This property is a count for the number of receive-side fanout threads for a particular datalink. Note that this property lets an administrator specify the desired `rxfanout`. However, based on the number of available CPUs and hardware RX rings, the system might choose a different (smaller or even higher) value for fanout.

`rxfanout-effective`
The number of CPUs is the upper bound on the receive side fanout while the number of `rxrings` is the lower bound. Thus, the actual receive-side fanout count can have a value different from the one set by the user.

`learn_limit`
Limits the number of new or changed MAC sources to be learned over a bridge link. When the number exceeds this value, learning on that link is temporarily disabled. Only non-VLAN, non-VNIC type links have this property.

The default value is **`1000`**. Valid values are greater or equal to 0.

`learn_decay`
Specifies the decay rate for source changes limited by `learn_limit`. This number is subtracted from the counter for a bridge link every 5 seconds. Only non-VLAN, non-VNIC type links have this property.

The default value is **`200`**. Valid values are greater or equal to 0.

`lro`
Specifies the user's disposition of turning LRO on or off or using system default LRO value on a data link.

The default value is `off`. Valid values are `off`, `on`, or `auto`. `auto` is to apply the default LRO setting on the data link.

`lro-effective`
Read-only property that shows the actual LRO status of a data link. Even if the user has enabled LRO for a data link, the system might not turn it on if it determines it is unsafe to do so. For instance, if IP is forwarding traffic using a data link, then the system would deem it unsafe to turn on LRO for that data link.

Valid values are `off` or `on`.

`mac-address`
Sets the primary MAC address for the data link. When set, changes the primary MAC address used by all current and future MAC clients of the underlying data link.

`maxbw`
Sets the full duplex bandwidth for the link. The bandwidth is specified as an integer with one of the scale suffixes (`K`, `M`, or `G` for Kbps, Mbps, and Gbps). If no units are specified, the input value will be read as Mbps. The default is no bandwidth limit.

`pool`
Bind the processing of packets for a given data link to a pool of processors defined and administered by `poolcfg(1M)` and `pooladm(1M)`. The binding of processes is similar to what occurs with the `cpus` link property, except that the list of CPUs is not explicit and is instead maintained by the pools facility.

If pools are enabled, and no pool is specified for the link, `pool_default` will be used for packet processing.

For zones with `ip-type=exclusive`, if a pool is specified through a pool zone property or `dedicated-cpus` allocation, that pool will also be used for all data links associated with the zone.

Specification of the `pool` property is not allowed on links with a `cpus` link property.

`pool-effective`
If the pools facility has been enabled, this read-only property displays the pool that is being used for packet processing. If the administrator has not assigned a pool to a data link, the pool will be `pool_default`.

If the pools facility is disabled, there is no effective pool and the value will be empty.

`priority`
Sets the relative priority for the link. The value can be given as one of the tokens `high`, `medium`, or `low`. The default is `high`. This priority is not reflected in any protocol priority fields on the wire, but used for packet processing scheduling within the system.

`rxringsavail`
A read-only property that specifies the number of rings available on the receive side.

`rxrings`
Specifies the number of receive rings side for the MAC client. A value of `sw` means this MAC client should not be assigned any RX ring and will be software-based. A value of `hw` means this MAC client can get one RX ring, if available, or will be software-based. A non-zero value means reserve that many rings for this MAC client, if available, and fail if not. If this property is not specified, the MAC client can get one RX ring, if available, or will be software-based.

`rxhwclntavail`
A read-only property that specifies the number of additional RX hardware-based MAC clients that can be created.

`txringsavail`
A read-only property that specifies the number of rings available on the transmit side.

`txrings`
Specifies the number of transmit rings for the MAC client. A value of `sw` means this MAC client should not be assigned any TX ring. A value of `hw` means this MAC client can get one TX ring, if available, or will be software-based. A non-zero value means reserve that many rings for this MAC client, if available, and fail if not. If this property is not specified, the MAC client can get one TX ring, if available, or will be software-based.

`txhwclntavail`
A read-only property that specifics the number of additional TX hardware-based MAC clients that can be created.

`stp`
Enables or disables Spanning Tree Protocol on a bridge link. Setting this value to `0` disables Spanning Tree, and puts the link into forwarding mode with BPDU guarding enabled. This mode is appropriate for point-to-point links connected only to end nodes. Only non-VLAN, non-VNIC type links have this property. The default value is `1`, to enable STP.

`forward`
Enables or disables forwarding for a VLAN. Setting this value to `0` disables bridge forwarding for a VLAN link. Disabling bridge forwarding removes that VLAN from the "allowed set" for the bridge. The default value is `1`, to enable bridge forwarding for configured VLANs.

`default_tag`
Sets the default VLAN ID that is assumed for untagged packets sent to and received from this link. Only non-VLAN, non-VNIC type links have this property. Setting this value to `0` disables the bridge forwarding of untagged packets to and from the port. The default value is `VLAN ID 1`. Valid values values are from 0 to 4094. The default VLAN ID is also referred to as the Port VLAN Identifier (PVID).

You cannot create a tagged VLAN or VLAN-tagged VNIC link with a VLAN ID that matches the default VLAN value of the underlying link. All untagged packets on the link are already associated with the default VLAN (PVID). To successfully create a tagged

VLAN or VLAN-tagged VNIC link with VLAN ID equal to the default VLAN value, you must first change the `default_tag` property of the underlying link to a different VLAN value.

When `default_tag=0`, all untagged packets on the link are no longer associated with any VLAN. As a result, you can create a VLAN link with any VLAN ID from 1 to 4094. Note that any received packets that are erroneously tagged with the PVID at an end-point might be dropped. This situation occurs if all the end-points on a given link do not agree on the PVID. All end-points on a link must use the same PVID and must not tag traffic with the PVID.

`stp_priority`
Sets the STP and RSTP Port Priority value, which is used to determine the preferred root port on a bridge. Lower numerical values are higher priority. The default value is `128`. Valid values range from 0 to 255.

`stp_cost`
Sets the STP and RSTP cost for using the link. The default value is `auto`, which sets the cost based on link speed, using `100` for 10Mbps, `19` for 100Mbps, `4` for 1Gbps, and `2` for 10Gbps. Valid values range from 1 to 65535.

`stp_edge`
Enables or disables bridge edge port detection. If set to `0` (false), the system assumes that the port is connected to other bridges even if no bridge PDUs of any type are seen. The default value is `1`, which detects edge ports automatically.

`stp_p2p`
Sets bridge point-to-point operation mode. Possible values are `true`, `false`, and `auto`. When set to `auto`, point-to-point connections are automatically discovered. When set to `true`, the port mode is forced to use point-to-point. When set to `false`, the port mode is forced to use normal multipoint mode. The default value is `auto`.

`stp_mcheck`
Triggers the system to run the RSTP `Force BPDU Migration Check` procedure on this link. The procedure is triggered by setting the property value to `1`. The property is automatically reset back to `0`. This value cannot be set unless the following are true:

- The link is bridged
- The bridge is protected by Spanning Tree
- The bridge `force-protocol` value is at least 2 (RSTP)

The default value is 0.

`protection`
Enables one or more types of link protection. Valid values are:

`mac-nospoof`
MAC address anti-spoof. An outbound packet's source MAC address must match the link's configured MAC address. Non-matching packets will be dropped. If the link belongs to a zone, turning `mac-nospoof` on will prevent the zone's owner from modifying the link's MAC address.

`ip-nospoof`
IP address anti-spoof. This protection type works in conjunction with the link property `allowed-ips`.

`allowed-ips` is a list containing IP (IPv4 or IPv6) addresses. This list is empty by default. Addresses that are implicitly in this list are: the link local IPv6 address conforming to RFC 2464 (derived from the link's MAC address); IPv4/IPv6 addresses learned from DHCP replies; the unspecified (all-zeros) IPv4/IPv6 address.

An outbound IP packet can pass if its source address is in `allowed-ips`.

An outbound ARP packet can pass if its sender protocol address is in `allowed-ips`.

When a datalink has been protected by setting `allowed-ips` to a set of one or more IP addresses, any attempts to configure IP addresses that are not in this set will fail with an `EPERM` error being returned to the user. Moreover, the interface may not be used for forwarding IP packets, and attempts to set the `ipadm(1M)` forwarding property on the interface will encounter an `EPERM` error.

`dhcp-nospoof`
DHCP client ID (DUID for DHCPv6) and hardware address anti-spoof. This protection type works in conjunction with the link property `allowed-dhcp-cids`.

Items in the `allowed-dhcp-cids` list should be formatted in the same way as the `CLIENT_ID` field in the `/etc/default/dhcpagent` file. The only difference is that `.` (period) should be used in place of `,` (comma) when specifying DUIDs. See `dhcpagent(1M)` for details.

An outbound DHCP (v4/v6) packet can pass only if these conditions are satisfied:

- If `allowed-dhcp-cids` is not configured and the packet type is:
  - DHCPv4, the client ID field must match the configured MAC address.
  - DHCPv6, the DUID must be of type 1 or 3 and the link layer address part of the DUID must match the configured MAC address.
- If `allowed-dhcp-cids` is configured and the packet type is:
  - DHCPv4, the client ID field must match one of the IDs on this list or the configured MAC address.
  - DHCPv6, the DUID field must match one of the IDs on this list or, the DUID must be of type 1 or 3 and the link layer address part of the DUID matches the configured MAC address.

`restricted`
This protection restricts outgoing packet types to just IPv4, IPv6, and ARP.

`vsi-mgrid`
An IPv6 address.

When the VDP service is enabled on a VNIC, properties of the VNIC are exchanged with the bridge using a 3-byte VSI Type ID and 1-byte VSI Version. A VSI Manager maintains the mapping between the {VSI Type ID-VSI Version} and the set of properties. The {VSI Manager ID, VSI Type id, VSI Version} tuple identifies a specific set of properties.

On a VNIC, the `vsi-mgrid` can be explicitly assigned. If the vsi-mgrid is not explicitly assigned, the `vsi-mgrid` is set to the `vsi-mgrid` value of the underlying link.

On physical link, `vsi-mgrid` specifies the default `vsi-manageid` for all the VNICs over it. The default value of the `vsi-mgrid` on a physical link is 0.

The default VSI Manager ID on a physical link is associated with the Oracle VSI Manager (`oracle_v1`). The Oracle VSI Manager is defined as a 3-byte encoding using the following link properties:

```
Bits            Properties
------------------------------------------------
0-4             Link Bandwidth Limit
                00000-10100 :   0-100% of link speed
                                in increments of 5%
                rest        :   reserved

                5-7             Link Speed
                                000 - Unknown
                                001 - 10 Mbps
                                010 - 100 Mbps
                                011 - 1 Gbps
                                100 - 10 Gbps
                                101 - 40 Gbps
                                110 - 100 Gbps
                                111 - Reserved

                8-12            Reserved

                13-15           Traffic Class (0-7)

                16-17           Link MTU
                                00 - 1500 bytes
                                01 - 9000 bytes
                                10 - Custom
                                11 - Reserved
```

```
18-23           Reserved
```

`vsi-mgrid-effective`
A read-only property for VNICs. The effective VSI Manager ID on a virtual link.

`vsi-mgrid-enc`
The encoding associated with the physical link's `vsi-mgrid`. Supported values include `oracle_v1` and `none`. If this property is set to `none`, the `vsi-typeid` and `vsi-vers` are not automatically generated over this link for VNICs that do not have their `vsi-mgrid` explicitly set.

`vsi-mgrid-enc-effective`
A read-only property for VNICs. The effective VSI Manager ID encoding used for a virtual link.

`vsi-typeid`
A 3-byte value that is used to determine the properties associated with a VNIC. The `vsi-typeid` is used along with the `vsi-vers` and `vsi-mgrid` to obtain the actual properties associated with the VNIC. When the `vsi-mgrid` is not explicitly on the VNIC, the `vsi-typeid` is automatically generated using the properties of the VNIC and the above encoding (`oracle_v1`).

`vsi-typeid-effective`
A read-only property. The effective VSI Type ID on a link.

`vsi-vers`
A 1-byte value that is used to determine the properties associated with a VNIC. The `vsi-vers` is used along with the `vsi-typeid` and `vsi-mgrid` to obtain the actual properties associated with the VNIC. When the `vsi-mgrid` is not explicitly on the VNIC, the `vsi-vers` is set to 0.

`vsi-vers-effective`
A read-only property. The effective VSI Version on a link.

`zone`
Specifies the zone to which the link belongs. This property can be modified only temporarily through `dladm`, and thus the `-t` option must be specified. To modify the zone assignment such that it persists across reboots, please use `zonecfg(1M)`. Possible values consist of any exclusive-IP zone currently running on the system. By default, the zone binding is as per `zonecfg(1M)`.

Wifi Link Properties

The following WiFi link properties are supported. Note that the ability to set a given property to a given value depends on the driver and hardware.

`channel`
Specifies the channel to use. This property can be modified only by certain WiFi links when in IBSS mode. The default value and allowed range of values varies by regulatory domain.

`powermode`
Specifies the power management mode of the WiFi link. Possible values are `off` (disable power management), `max` (maximum power savings), and `fast` (performance-sensitive power management). Default is `off`.

`radio`
Specifies the radio mode of the WiFi link. Possible values are `on` or `off`. Default is `on`.

`speed`
Specifies a fixed speed for the WiFi link, in megabits per second. The set of possible values depends on the driver and hardware (but is shown by `show-linkprop`); common speeds include 1, 2, 11, and 54. By default, there is no fixed speed.

Ethernet Link Properties

The following MII Properties, as documented in `ieee802.3(5)`, are supported in read-only mode:

- `duplex`
- `state`
- `adv_autoneg_cap`
- `adv_10gfdx_cap`
- `adv_1000fdx_cap`
- `adv_1000hdx_cap`
- `adv_100fdx_cap`
- `adv_100hdx_cap`
- `adv_10fdx_cap`
- `adv_10hdx_cap`

Each `adv_` property (for example, `adv_10fdx_cap`) also has a read/write counterpart `en_` property (for example, `en_10fdx_cap`) controlling parameters used at auto-negotiation. In the absence of Power Management, the `adv*` speed/duplex parameters provide the values that are both negotiated and currently effective in hardware. However, with Power Management enabled, the speed/duplex capabilities currently exposed in hardware might be a subset of the set of bits that were used in initial link parameter negotiation. Thus the MII `adv_*` parameters are marked read-only, with an additional set of `en_*` parameters for configuring speed and duplex properties at initial negotiation.

Note that the `adv_autoneg_cap` does not have an `en_autoneg_cap` counterpart: the `adv_autoneg_cap` is a 0/1 switch that turns off/on autonegotiation itself, and therefore cannot be impacted by Power Management.

In addition, the following Ethernet properties are reported:

`flowctrl`
Establishes flow-control modes that will be advertised by the device. Valid input is one of:

`auto`
Flow control mode on the device is dynamically determined. To see the actual flow control mode set on the device, check the `flowctrl-effective` link property.

`no`
No flow control enabled.

`rx`
Receive, and act upon incoming pause frames.

`tx`
Transmit pause frames to the peer when congestion occurs, but ignore received pause frames.

`pfc`
Transmit pause frames including the priority value of the traffic that should be paused. Receive pause frames, and act upon the traffic whose priority values are specified in the frame.

`bi`
Bidirectional flow control.

Note that the actual settings for this value are constrained by the capabilities allowed by the device and the link partner.

`gvrp-timeout`
Specifies wait period between VID announcement broadcasts, in milliseconds.

`flowctrl-effective`
Actual flow-control mode configured on the device. When `flowctrl` property is set to `auto`, this indicates the flow control mode that is in effect. This is a read-only property.

`mtu`
The maximum client SDU (Send Data Unit) supported by the device. Valid range is 68-65536.

`ntcs`
The number of Traffic Classes supported on the device. A device supporting extensions for DCB (Data Center Bridging) can support multiple traffic classes. This property can be used to determine if the device supports DCB extensions. This is a read-only property.

`pfcmap`
This property is used to indicate the 802.1p priority values for which PFC (Priority-based flow control) is enabled. This is an 8-bit mask, in which an individual bit signifies whether PFC is enabled for the corresponding priority. For priorities that have PFC enabled, the device will transmit a pause frame for that priority in the event of congestion. This is relevant only if `ntcs` is greater than zero and `flowctrl-effective` is `pfc`.

`pfcmap-rmt-effective`
This property is used to indicate the PFC configuration of the remote peer, usually an adjacent switch.

`pfcmap-lcl-effective`
This property is used to indicate the effective PFC configuration on the system. The value can be `pfcmap` or `pfcmap-rmt-effective` depending on LLDP DCBx negotiations.

`speed`
(read-only) The operating speed of the device, in Mbps.

`tagmode`
This link property controls the conditions in which 802.1Q VLAN tags will be inserted in packets being transmitted on the link. Two mode values can be assigned to this property:

`normal` Insert a VLAN tag in outgoing packets under the following conditions:

- The packet belongs to a VLAN.
- The user requested priority tagging.

`vlanonly` Insert a VLAN tag only when the outgoing packet belongs to a VLAN. If a tag is being inserted in this mode and the user has also requested a non-zero priority, the priority is honored and included in the VLAN tag.

The default value is `vlanonly`.

`vlan-announce`
This property controls automatic VLAN ID anouncement. When enabled, it broadcasts the VIDs of any VNICs or VLANs configured on the device. It supports both physical links and aggregations. Possible values are:

`off`
No VID announcements will be sent.

`gvrp`
Announcements sent using GVRP protocol, as defined in 802.1D. See `gvrp-timeout` to configure broadcast frequency.

InfiniBand Link Properties

The following properties are supported only on IB partition object links.

`linkmode`
Sets the link transport service type on an IB partition datalink. The default value is `cm`. Valid values are:

`cm`
Connected Mode. This mode uses a default MTU of 65520 and supports a maximum MTU of 65535 bytes. If Connected Mode is not available for a remote node, Unreliable Datagram mode will automatically be used instead.

`ud`
Unreliable Datagram Mode. This mode uses a default MTU of 2044 and supports a maximum MTU of 4092 bytes.

IP Tunnel Link Properties

The following IP tunnel link properties are supported.

`hoplimit`
Specifies the IPv4 TTL or IPv6 hop limit for the encapsulating outer IP header of a tunnel link. This property exists for all tunnel types. The default value is 64.

`encaplimit`
Specifies the IPv6 encapsulation limit for an IPv6 tunnel as defined in RFC 2473. This value is the tunnel nesting limit for a given tunneled packet. The default value is 4. A value of 0 disables the encapsulation limit.

Examples

EXAMPLE 1 Display Datalink Configuration

The following command shows the effect of invoking `dladm` with no arguments.

```
# dladm
LINK                CLASS     MTU    STATE    OVER
net0                phys      1500   up       --
net1                phys      1500   up       --
net2                phys      1500   unknown  --
net3                phys      1500   up       --
vnic1               vnic      1500   up       net1
vlan1               vlan      1500   up       net1
aggr1               aggr      1500   up       net2 net3
stub1               etherstub 9000   unknown  --
```

EXAMPLE 2 Configuring an Aggregation

To configure a data-link over an aggregation of devices `bge0` (linkname `net0`) and `bge1` (linkname `net1`) with key 1, enter the following command:

```
# dladm create-aggr -l net0 -l net1 1
```

To configure an IEEE 802.3ad link aggregation of devices `e1000g1` (linkname `net0`) and `e1000g2` (linkname `net1`) with the name `aggr1`, enter following command:

```
# dladm create-aggr -l net0 -l net1 aggr1
```

To configure an Datalink Multipathing (`dlmp`) link aggregation of devices `ixgbe1` (linkame `net2`) and `ixgbe2` (linkname `net3`) with the name `aggr2` enter following command:

```
# dladm create-aggr -m dlmp -l net2  -l net3 aggr2
```

To list aggregations, enter following command:

```
# dladm show-aggr
LINK              MODE    POLICY   ADDRPOLICY           LACPACTIVITY  LACPTIMER
aggr1             trunk   L4       auto                 off           short
aggr2             dlmp    --       --                   --            --
```

EXAMPLE 3 Connecting to a WiFi Link

To connect to the most optimal available unsecured network on a system with a single WiFi link (as per the prioritization rules specified for `connect-wifi`), enter the following command:

```
# dladm connect-wifi
```

EXAMPLE 4 Creating a WiFi Key

To interactively create the WEP key `mykey`, enter the following command:

```
# dladm create-secobj -c wep mykey
```

Alternatively, to non-interactively create the WEP key `mykey` using the contents of a file:

```
# umask 077
 # cat >/tmp/mykey.$$ <<EOF
 12345
 EOF
 # dladm create-secobj -c wep -f /tmp/mykey.$$ mykey
 # rm /tmp/mykey.$$
```

EXAMPLE 5 Connecting to a Specified Encrypted WiFi Link

To use key `mykey` to connect to ESSID `wlan` on link `ath0`, enter the following command:

```
# dladm connect-wifi -k mykey -e wlan ath0
```

EXAMPLE 6 Changing a Link Property

To set `powermode` to the value `fast` on link `pcwl0`, enter the following command:

```
# dladm set-linkprop -p powermode=fast pcwl0
```

EXAMPLE 7 Connecting to a WPA-Protected WiFi Link

Create a WPA key `psk` and enter the following command:

```
# dladm create-secobj -c wpa psk
```

To then use key `psk` to connect to ESSID `wlan` on link `ath0`, enter the following command:

```
# dladm connect-wifi -k psk -e wlan ath0
```

EXAMPLE 8 Renaming a Link

To rename the `bge0` link to `mgmt0`, enter the following command:

```
# dladm rename-link bge0 mgmt0
```

EXAMPLE 9 Replacing a Network Card

Consider that the `bge0` device, whose link was named `mgmt0` as shown in the previous example, needs to be replaced with a `ce0` device because of a hardware failure. The `bge0` NIC is

EXAMPLE 9 Replacing a Network Card *(Continued)*

physically removed, and replaced with a new `ce0` NIC. To associate the newly added `ce0` device with the `mgmt0` configuration previously associated with `bge0`, enter the following command:

```
# dladm rename-link ce0 mgmt0
```

EXAMPLE 10 Removing a Network Card

Suppose that in the previous example, the intent is not to replace the `bge0` NIC with another NIC, but rather to remove and not replace the hardware. In that case, the `mgmt0` datalink configuration is not slated to be associated with a different physical device as shown in the previous example, but needs to be deleted. Enter the following command to delete the datalink configuration associated with the `mgmt0` datalink, whose physical hardware (`bge0` in this case) has been removed:

```
# dladm delete-phys mgmt0
```

EXAMPLE 11 Using Parseable Output to Capture a Single Field

The following assignment saves the MTU of link `net0` to a variable named `mtu`.

```
# mtu=‘dladm show-link -p -o mtu net0‘
```

EXAMPLE 12 Using Parseable Output to Iterate over Links

The following script displays the state of each link on the system.

```
# dladm show-link -p -o link,state | while IFS=: read link state; do
            print "Link $link is in state $state"
        done
```

EXAMPLE 13 Configuring VNICs

Create two VNICs with names `hello0` and `test1` over a single physical link `net0`:

```
# dladm create-vnic -l net0 hello0
# dladm create-vnic -l net0 test1
```

EXAMPLE 14 Configuring VNICs and Allocating Bandwidth and Priority

Create two VNICs with names `hello0` and `test1` over a single physical link `net0` and make `hello0` a high priority VNIC with a factory-assigned MAC address with a maximum bandwidth of 50 Mbps. Make `test1` a low priority VNIC with a random MAC address and a maximum bandwidth of 100Mbps.

```
# dladm create-vnic -l net0 -m factory -p maxbw=50,priority=high hello0
# dladm create-vnic -l net0 -m random -p maxbw=100M,priority=low test1
```

EXAMPLE 15 Configuring a VNIC with a Factory MAC Address

First, list the available factory MAC addresses and choose one of them:

```
# dladm show-phys -m net0
LINK            SLOT         ADDRESS              INUSE    CLIENT
net0            primary      0:e0:81:27:d4:47     yes      net0
net0            1            8:0:20:fe:4e:a5      no
net0            2            8:0:20:fe:4e:a6      no
net0            3            8:0:20:fe:4e:a7      no
```

Create a VNIC named `hello0` and use slot 1's address:

```
# dladm create-vnic -l net0 -m factory -n 1 hello0
# dladm show-phys -m net0
LINK            SLOT         ADDRESS              INUSE    CLIENT
net0            primary      0:e0:81:27:d4:47     yes      net0
net0            1            8:0:20:fe:4e:a5      yes      hello0
net0            2            8:0:20:fe:4e:a6      no
net0            3            8:0:20:fe:4e:a7      no
```

EXAMPLE 16 Creating a VNIC with User-Specified MAC Address, Binding it to Set of Processors

Create a VNIC with name `hello0`, with a user specified MAC address, and a processor binding `0, 2, 4-6`.

```
# dladm create-vnic -l net0 -m 8:0:20:fe:4e:b8 -p cpus=0,2,4-6 hello0
```

EXAMPLE 17 Creating a Virtual Network Without a Physical NIC

First, create an etherstub with name `stub1`:

```
# dladm create-etherstub stub1
```

Create two VNICs with names `hello0` and `test1` on the etherstub. This operation implicitly creates a virtual switch connecting `hello0` and `test1`.

```
# dladm create-vnic -l stub1 hello0
# dladm create-vnic -l stub1 test1
```

EXAMPLE 18 Displaying Bridge Information

The following commands use the `show-bridge` subcommand with no and various options.

```
# dladm show-bridge
BRIDGE       PROTECT ADDRESS           PRIORITY DESROOT
foo          stp     32768/8:0:20:bf:f 32768    8192/0:d0:0:76:14:38
bar          stp     32768/8:0:20:e5:8 32768    8192/0:d0:0:76:14:38

# dladm show-bridge -l foo
LINK         STATE        UPTIME   DESROOT
hme0         forwarding   117      8192/0:d0:0:76:14:38
qfe1         forwarding   117      8192/0:d0:0:76:14:38
```

EXAMPLE 18 Displaying Bridge Information *(Continued)*

```
# dladm show-bridge -s foo
BRIDGE       DROPS        FORWARDS
foo          0            302

# dladm show-bridge -ls foo
LINK         DROPS     RECV      XMIT
hme0         0         360832    31797
qfe1         0         322311    356852

# dladm show-bridge -f foo
DEST              AGE     FLAGS  OUTPUT
8:0:20:bc:a7:dc   10.860  --     hme0
8:0:20:bf:f9:69   --      L      hme0
8:0:20:c0:20:26   17.420  --     hme0
8:0:20:e5:86:11   --      L      qfe1
```

EXAMPLE 19 Creating an IPv4 Tunnel

The following sequence of commands creates and then displays a persistent IPv4 tunnel link named `mytunnel0` between 66.1.2.3 and 192.4.5.6:

```
# dladm create-iptun -T ipv4 -a local=66.1.2.3,remote=192.4.5.6 mytunnel0
# dladm show-iptun mytunnel0
LINK            TYPE  FLAGS  SOURCE              DESTINATION
mytunnel0       ipv4  --     66.1.2.3            192.4.5.6
```

A point-to-point IP interface can then be created over this tunnel link:

```
# ipadm create-ip mytunnel0
# ipadm create-addr -T static -a local=10.1.0.1,remote=10.1.0.2 \
mytunnel0/addr
# ipadm show-addr mytunnel0/addr
ADDROBJ           TYPE     STATE        ADDR
mytunnel0/addr    static   ok           10.1.0.1->10.1.0.2
```

EXAMPLE 20 Creating a 6to4 Tunnel

The following command creates a 6to4 tunnel link. The IPv4 address of the 6to4 router is 75.10.11.12.

```
# dladm create-iptun -T 6to4 -a local=75.10.11.12 sitetunnel0
# dladm show-iptun sitetunnel0
LINK            TYPE  FLAGS  SOURCE              DESTINATION
sitetunnel0     6to4  --     75.10.11.12         --
```

The following command creates an IPv6 interface on this tunnel:

EXAMPLE 20 Creating a 6to4 Tunnel *(Continued)*

```
# ipadm create-ip sitetunnel0
# ipadm show-addr sitetunnel0/_a
ADDROBJ           TYPE     STATE        ADDR
sitetunnel0/_a    static   ok           2002:4b0a:b0c::1/16
```

Note that the system automatically configures the IPv6 address on the 6to4 IP interface. See `ipadm(1M)` for a description of how IPv6 addresses are configured on 6to4 tunnel links.

EXAMPLE 21 Using Link Protection

To enable link protection:

```
# dladm set-linkprop \
-p protection=mac-nospoof,restricted,ip-nospoof,dhcp-nospoof vnic0
```

To disable link protection:

```
# dladm reset-linkprop -p protection vnic0
```

To modify the `allowed-ips` list:

```
# dladm set-linkprop -p allowed-ips=10.0.0.1,10.0.0.2 vnic0
```

To modify the `allowed-dhcp-cids` list:

```
# dladm set-linkprop -p allowed-dhcp-cids=hello vnic0
```

To display the resulting configuration:

```
# dladm show-linkprop -p protection,allowed-ips vnic0

LINK     PROPERTY         PERM   VALUE        DEFAULT   POSSIBLE
vnic0    protection       rw     mac-nospoof, --        mac-nospoof,
                                 restricted,            restricted,
                                 ip-nospoof,            ip-nospoof,
                                 dhcp-nospoof           dhcp-nospoof

vnic0    allowed-ips      rw     10.0.0.1,    --        --
                                 10.0.0.2

vnic0    allowed-dhcp-cids rw    hello        --        --
```

EXAMPLE 22 Creating an IB Partition

The following command creates a partition `ffff.ibp0` with partition key `0xffff` on the physical link `ibp0`.

```
# dladm create-part -P ffff -l ibp0 ffff.ibp0
```

EXAMPLE 23 Displaying IB Partition Information

The following command displays IB partition information.

```
# dladm show-part
LINK         PKEY OVER         STATE    FLAGS
ffff.ibp0    FFFF ibp0         up       ----
```

EXAMPLE 24 Displaying IB Data Links Information

The following command displays IB data links information.

```
# dladm show-ib
LINK         HCAGUID         PORTGUID        PORT STATE  PKEYS
net0         3BA000100CD7C   3BA000100CD7D   1    down   FFFF
net1         3BA000100CD7C   3BA000100CD7E   2    down   FFFF
net3         5AD0000033634   5AD0000033636   2    up     FFFF,8001
net2         5AD0000033634   5AD0000033635   1    up     FFFF,8001
```

EXAMPLE 25 Deleting a Partition

The following command deletes the partition `ffff.ibp0`.

```
# dladm delete-part ffff.ibp0
```

EXAMPLE 26 Using `show-link` to Display Partition Information

The following command uses the `show-link` subcommand to display partition information.

```
# dladm show-link
LINK        CLASS     MTU    STATE    OVER
e1000g0     phys      1500   up       --
e1000g1     phys      1500   unknown  --
net0        phys      65520  down     --
net3        phys      65520  up       --
net2        phys      65520  up       --
net1        phys      65520  down     --
pffff.ibp0  part      2044   down     ibp0
p8001.ibp2  part      65520  unknown  ibp2
```

EXAMPLE 27 Displaying Links in All Zones from the Global Zone

The `show-link` command shown below displays data links in all zones from the global zone. Links that are not in the global zone are displayed with the zonename prefix followed by the slash (`/`) separator.

In this example, `net0` is a VNIC created in the global zone, `zone1/net0` is an automatically created VNIC for `zone1`, and `zone2/net0` is an automatically created VNIC for `zone2`.

```
# dladm show-link
LINK                CLASS     MTU    STATE    OVER
e1000g0             phys      1500   up       --
```

EXAMPLE 27 Displaying Links in All Zones from the Global Zone *(Continued)*

```
e1000g1             phys      8170   unknown  --
e1000g2             phys      1500   unknown  --
e1000g3             phys      1500   unknown  --
net0                vnic      1500   up       e1000g0
zone1/net0          vnic      1500   up       e1000g0
zone2/net0          vnic      1500   up       e1000g0
```

EXAMPLE 28 Displaying Links in the Global Zone

The following `show-link` command displays data links in the global zone only.

```
# dladm show-link -z global
LINK                CLASS     MTU    STATE    OVER
e1000g0             phys      1500   up       --
e1000g1             phys      8170   unknown  --
e1000g2             phys      1500   unknown  --
e1000g3             phys      1500   unknown  --
net0                vnic      1500   up       e1000g0
```

EXAMPLE 29 Displaying Links for a Specified Zone

The following `show-link` command displays data links in a specific, non-global zone.

```
# dladm show-link -z zone1
LINK                CLASS     MTU    STATE    OVER
zone1/net0          vnic      1500   up       e1000g0
```

EXAMPLE 30 Displaying Links for a Specified Zone from the Global Zone

The following `show-link` command displays, from the global zone, data links in a specific, non-global zone.

```
# dladm show-link -z zone1
LINK                CLASS     MTU    STATE    OVER
zone1/net0          vnic      1500   up       e1000g0
```

EXAMPLE 31 Displaying Links in a Non-Global Zone

The following `show-link` shown below is invoked from `zone1` and displays only data links for that zone.

Note that, in `show-link` output, the `zone1/` prefix is not displayed. The prefix is not displayed because the command was invoked from within the zone.

```
# zlogin zone1
# dladm show-link -z zone1
LINK                CLASS     MTU    STATE    OVER
net0                vnic      1500   up       ?
```

EXAMPLE 32 Using -Z Option to Display the Current Zone

The command below presumes the following conditions:

- The link `net1` is currently assigned to `zoneA`. The entries `net1` and `zoneA/net1` represents the same link. The `ZONE` column for these two entries is the same and is the name of the zone to which the link is currently assigned.
- The link `net2` is not assigned to any non-global zone.
- The link `zoneB/net2` is an automatic VNIC created for `zoneB`.
- The link `zoneC/net2` is an automatic VNIC created for `zoneC`.
- The link `zoneD/net2` is an IP tunnel created inside `zoneD`. Unlike for `net1`, each entry for `net2` represents a different link. The `ZONE` column for these entries is different.

```
# dladm show-link -Z
LINK        ZONE      CLASS     MTU    STATE    OVER
e1000g0     global    phys      1500   up       --
e1000g1     global    phys      1500   up       --
net1        zoneA     vnic      1500   up       e1000g0
zoneA/net1  zoneA     vnic      1500   up       e1000g0
net2        global    vnic      1500   up       e1000g1
zoneB/net2  zoneB     vnic      1500   up       e1000g1
zoneC/net2  zoneC     vnic      1500   up       e1000g1
zoneD/net2  zoneD     iptun     65515  up       --
```

EXAMPLE 33 Displaying VDP Information

The following command displays VDP information for `vnic1`.

```
# dladm show-ether -P vdp vnic1
LINK    VSI     VSIID           VSI-TYPEID      VSI-STATE CMD-PENDING
ixgbe1  vnic1   2:8:20:3:2:b    0x58/0          ASSOC     DEASSOC
```

EXAMPLE 34 Displaying ECP Information

The following command displays ECP information for `ixgbe1`.

```
# dladm show-ether -P ecp  ixgbe1
LINK    SEQNO   ACKNO   LAST-ACK        MAX-RETRIES     TIMEOUTS
ixgbe1  65535   25660   0               3                164
```

EXAMPLE 35 Setting the VSI Manager ID, VSI Type, and VSI Version

The following commands set the VSI Manager ID, VSI Type, and VSI Version on `vnic1`.

```
# dladm set-linkprop -p vsi-mgrid=fe80::214:4fff:fec2:67c8 vnic1
# dladm set-linkprop -p vsi-typeid=0x64,vsi-vers=1 vnic1
```

EXAMPLE 36 Migrating a VLAN, Modifying its VLAN-ID

The following command sequence shows how you migrate a VLAN and modify its VLAN-ID.

EXAMPLE 36 Migrating a VLAN, Modifying its VLAN-ID *(Continued)*

```
# dladm show-vlan vlan0
LINK                VID      OVER               FLAGS
vlan0               100      net0               -----
# dladm modify-vlan -l net1 -v 200 vlan0
# dladm show-vlan vlan0
LINK                VID      OVER               FLAGS
vlan0               200      net1               -----
```

EXAMPLE 37 Migrating Multiple VNICs

The following command sequence shows how you migrate multiple VNICs.

```
# dladm show-vnic
LINK      OVER     SPEED  MACADDRESS        MACADDRTYPE    VID
vnic0     net0     1000   2:8:20:ec:c4:1d   random         0
vnic1     net0     1000   2:8:20:ec:c4:1e   random         0
# dladm modify-vnic -l net1 -L net0
# dladm show-vnic
LINK      OVER     SPEED  MACADDRESS        MACADDRTYPE    VID
vnic0     net1     1000   2:8:20:ec:c4:1d   random         0
vnic1     net1     1000   2:8:20:ec:c4:1e   random         0
```

EXAMPLE 38 Migrating a VNIC and Modifying its MAC Address

The following command sequence shows how you migrate a VNIC and modify its MAC address.

```
# dladm show-vnic vnic0
LINK      OVER     SPEED  MACADDRESS        MACADDRTYPE    VID
vnic0     net0     1000   2:8:20:ec:c4:1d   random         0
# dladm modify-vnic -l net1 -m 2:8:20:00:01:02 vnic0
# dladm show-vnic vnic0
LINK      OVER     SPEED  MACADDRESS        MACADDRTYPE    VID
vnic0     net1     1000   2:8:20:0:1:2      fixed          0
```

EXAMPLE 39 Configuring cos and ETS Bandwidth

The following example creates a VNIC with name `vnic1` over the physical link `net1` and assigns to it a `cos` value of 3.

```
# dladm create-vnic -p cos=3 -l net1 vnic1
```

All packets transmitted by `vnic1` will have a VLAN header with the priority field set to 3.

Additionally, if the underlying physical NIC has registered DCB capability, an ETS bandwidth can be assigned to `vnic1`. The following commands assume the LLDP package is not installed or enabled.

EXAMPLE 39 Configuring cos and ETS Bandwidth *(Continued)*

Check if the underlying NIC has registered DCB capability using the `ntcs` link property. If the value of `ntcs` is non-zero, the underlying NIC has registered DCB capability.

```
# dladm show-linkprop -p ntcs net1
```

The following command assigns an ETS bandwidth of 10% of the link's bandwidth to `vnic1`.

```
# dladm set-linkprop -p etsbw_lcl=10 vnic1
```

Note if the `maxbw` link property has also been set, then the traffic is limited by the `maxbw` value.

With the LLDP package (`service/network/lldp`) installed and enabled, the ETS bandiwdth configuration will follow the IEEE 802.1Qaz specification.

The LLDP ETS TLV willing property determines whether the local or the remote's configuration is effective.

The `etsbw-lcl-advice` link property indicates the value recommended by the remote, if available. The `etsbw-lcl-effective` link property will indicate the actual ETS bandwidth assigned to `vnic1`, as shown below.

```
# dladm show-linkprop -p etsbw-lcl-advice,etsbw-lcl-effective vnic1
```

EXAMPLE 40 Displaying Help

The following command illustrates the use of invoking the `help` subcommand without arguments.

```
# dladm help
The following subcommands are supported:
Bridge subcommands          : add-bridge, create-bridge,
                              delete-bridge, modify-bridge,
                              remove-bridge, show-bridge
Etherstub subcommands       : create-etherstub, delete-etherstub,
                              show-etherstub
IB subcommands              : create-part, delete-part,
                              show-ib, show-part
IP tunnel subcommands       : create-iptun, delete-iptun,
                              modify-iptun, show-iptun
Link Aggregation subcommands: add-aggr, create-aggr, delete-aggr,
                              modify-aggr, remove-aggr, show-aggr
Link subcommands            : rename-link, reset-linkprop,
                              set-linkprop, show-link, show-linkprop
Secure Object subcommands   : create-secobj, delete-secobj,
                              show-secobj
VLAN subcommands            : create-vlan, delete-vlan, show-vlan
VNIC subcommands            : create-vnic, delete-vnic, show-vnic
Wifi subcommands            : connect-wifi, disconnect-wifi,
```

EXAMPLE 40 Displaying Help *(Continued)*

```
                              scan-wifi, show-wifi
Miscellaneous subcommands   : delete-phys, show-ether, show-phys,
                              show-usage
For more info, run: dladm help subcommand
```

The following command illustrates the use of invoking the `help` subcommand with a specific subcommand.

```
# dladm help create-vnic
usage:
          create-vnic     [-t] -l link [-m value | auto |
          {factory [-n slot-id]} | {random [-r prefix]} |
          {vrrp -V vrid -A {inet | inet6}} [-v vid [-f]]
          [-p prop=value[,...]] vnic-link

        example:
          # dladm create-vnic -l net0 -m factory -n 2 -p mtu=1200 vnic1
```

Attributes

See `attributes(5)` for descriptions of the following attributes:

`/usr/sbin`

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os
Interface Stability	Committed

`/sbin`

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os
Interface Stability	Committed

Note that, for both `/usr/sbin` and `/sbin`, the `-s` and `-i` options to the `show-aggr`, `show-link` and `show-vnic` subcommands are Committed Obsolete.

Note that, for both `/usr/sbin` and `/sbin`, `show-linkprop`'s `*effective` properties have an interface stability of Volatile.

Note that the bridge-related subcommands, described with `dladm` subcommands above, require installation of the `pkg://solaris/network/bridging` package.

See Also acctadm(1M), autopush(1M), dhcpagent(1M), dlstat(1M), ifconfig(1M), ipadm(1M), ipsecconf(1M), lldpadm(1M), ndd(1M), pooladm(1M), poolcfg(1M), psrset(1M), vrrpadm(1M), wpad(1M), zonecfg(1M), attributes(5), ieee802.3(5), dlpi(7P)

Notes The preferred method of referring to an aggregation in the aggregation subcommands is by its link name. Referring to an aggregation by its integer *key* is supported for backward compatibility, but is not necessary. When creating an aggregation, if a *key* is specified instead of a link name, the aggregation's link name will be automatically generated by `dladm` as `aggr`*key*.

Name dlmgmtd – datalink management daemon

Synopsis

```
/usr/sbin/dlmgmtd

svc:/network/datalink-management:default
```

Description `dlmgmtd` is a system daemon that handles administrative events for network datalink interfaces. It is controlled through the service management facility (SMF) service instance:

```
svc:/network/datalink-management:default
```

The daemon should not be invoked directly. It does not constitute an administrative nor a programming interface. The administrative interface for managing datalinks is `dladm`(1M).

Options The daemon has no options.

Attributes See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os
Interface Stability	Private

See Also `dladm`(1M), `attributes`(5)

Notes By default, Solaris assigns link names with the prefix of `net`. Before installing Solaris, you can change this default by modifying the value of the `linkname-policy/phys-prefix` SMF property of the service `svc:/network/datalink-management:default`. Specify a new value for this property in the System Configuration manifests used the Automated Install (AI) program. See *Oracle Solaris Administration: Network Interfaces and Network Virtualization* for details.

Name dlstat – report data links statistics

Synopsis

```
dlstat [-r] [-t] [-Z] [-i interval] [-z zone[,...]] [link]

dlstat [-a | -A] [-Z] [-i interval] [-p] [ -o field[,...]]
     [-u R|K|M|G|T|P] [-z zone[,...]] [link]

dlstat show-phys [-r] [-t] [-Z] [-i interval] [-a]
     [-p] [-o field[,...]] [-u R|K|M|G|T|P] [-z zone[,...]] [link]

dlstat show-link [-r] [-t] [-Z] [-i interval] [-a]
     [-p] [-o field[,...]] [-u R|K|M|G|T|P] [-z zone[,...]] [link]

dlstat show-link -h [-Z] [-a] -f filename [-d] [-F format] [-s time]
          [-e time] [-z zone[,...]] [link]

dlstat show-aggr [-r] [-t] [-Z] [-i interval] [-p] [-o field[,...]]
     [-u R|K|M|G|T|P] [-z zone[,...]] [link]

dlstat show-ether -P protocol [-Z] [-i interval] [-p] [-o field[,...]]
     [-u R|K|M|G|T|P] [ether-link]

dlstat help [subcommand-name]
```

Description The `dlstat` command reports run time statistics about data links. `dladm(1M)` `show-phys` provides link-name information to `dlstat show-phys`. `dladm(1M)` `show-link` provides link-name information to `dlstat show-link`. `dladm(1M)` `show-aggr` provides link-aggregation information to `dlstat show-aggr`.

`dlstat` has the forms of commands shown in the `SYNOPSIS`, above. The first two forms do not have subcommands, while the remaining forms do. All forms are described under "Subcommands," below.

Options The `dlstat` command has the following options and operands that are common (unless explicitly marked otherwise) among a number of command forms shown under "Subcommands," below.

`-a`
Dump all total statistics fields.

`-i` *interval*
Specify an interval in seconds at which statistics are refreshed. The default interval is one second.

`-o` *field*[,...]
Display a case-insensitive, comma-separated list of output fields to display. The field name must be one of the fields listed below, or the special value `all` to display all supported fields.

List of supported RX fields:

- `link`
- `index`

- `rbytes`
- `ipkts`
- `intrs` (only applicable without a subcommand or with `show-link`)
- `polls` (only applicable without a subcommand or with `show-link`)
- `idrops` (applicable only with the `-r` option, either without a subcommand or with `show-link`)

List of TX side fields:

- `link`
- `index`
- `obytes`
- `opkts`
- `odrops` (applicable only with the `-t` option, either without a subcommand or with `show-link`)

For the `show-phys` subcommand, the `index` column identifies individual RX and TX hardware rings within a physical device. For the `show-link` subcommand, the `index` column identifies RX and TX hardware lanes within a data link. See *Oracle Solaris Administration: Network Interfaces and Network Virtualization* for an explanation of the difference between hardware rings and hardware lanes.

`-p`
Display output in a stable, machine-parseable format.

`-r`
Display receive-side statistics only. Includes bytes and packets received, hardware and software drops, and so forth. See "Examples" for complete listing.

`-r` and `-t` could be used together in one command to display both receive-side as well as transmit-side statistics simultaneously.

`-t`
Display transmit-side statistics only. Includes bytes and packets sent, drops, and so forth. See "Examples" for complete listing.

`-u R|K|M|G|T|P`
If used, allows choosing the unit in which to display all statistics, for example, `R`:raw count, `K`:Kilobits, `M`:Megabits, `T`:Terabits, `P`:Petabits. If not used, then different units, as appropriate, are used to display the statistics, using the format *xy.zU*, where *x*, *y*, and *z* are numbers and *U* is the appropriate unit.

`-Z`
Display `ZONE` column in the output.

`-z` *zone*[,...]
Display the statistics only for links in the specified zone. By default, `dlstat` displays the statistics for links in all the zones when it is run from the global zone.

When run from a non-global zone, `dlstat` displays statistics only for links in that zone. A non-global zone cannot see links in other zones.

link
If specified, display the statistics only for the named link, physical device (for `show-phys`), or aggregation (for `show-aggr`). Otherwise, display statistics for all links, devices, or aggregations.

Sub-commands `dlstat` supports the following command forms.

`dlstat` [`-r`] [`-t`] [`-Z`] [`-i` *interval*] [`-z` *zone*[,...]] [*link*]
Iteratively examine all links and report statistics. The output is sorted in descending order of link utilization. If no link is specified, the system displays statistics for all links. The traffic statistics are displayed per link and not per physical device. For example, for a VNIC configured on a physical link, traffic flowing through that VNIC is not reflected in the statistics for the underlying physical link. However, the link statistics will include traffic that matches user-defined flows configured on top of that link.

This command form has one option that is not described under "Options," above:

`-A`
Dump all statistics fields for this data-link. Output statistics of this command are inclusive of all the statistics reported by all other `dlstat` commands.

`dlstat` [`-a` | `-A`] [`-Z`] [`-i` *interval*] [`-p`] [`-o` *field*[,...]] [`-u` `R`|`K`|`M`|`G`|`T`|`P`] [`-z` *zone*[,...]] [*link*]
Allows specifying which statistics to display.

The options for this command form are described under "Options," above.

`dlstat show-phys` [`-r`] [`-t`] [`-Z`] [`-i` *interval*] [`-a`] [`-p`] [`-o` *field*[,...]] [`-u` `R`|`K`|`M`|`G`|`T`|`P`] [`-z` *zone*[,...]] [*link*]
Display statistics for a physical device.

The options for this subcommand are described under "Options," above.

`dlstat show-link` [`-r`] [`-t`] [`-Z`] [`-i` *interval*] [`-a`] [`-p`] [`-o` *field*[,...]] [`-u` `R`|`K`|`M`|`G`|`T`|`P`] [`-z` *zone*[,...]] [*link*]
Display statistics for a link.

`dlstat show-link -h` [`-Z`] [`-a`] `-f` *filename* [`-d`] [`-F` *format*] [`-s` *time*] [`-e` *time*] [`-z` *zone*[,...]] [*link*]
Show the network usage history from a stored extended accounting file. Use of this syntax requires that net accounting has been previously configured and enabled by using `acctadm(1M)`. The default output is the summary of network usage of the existing links for the entire period when extended accounting was enabled.

The *link* argument is as described under "Options," above.

`-a`
Display all historical network usage for the specified period when extended accounting is enabled. This includes usage information about links that have already been deleted.

`-f` *filename*
Specify the file from which extended accounting records of network usage history are read.

`-d`
Display the dates for which there is logging information. The date is in the format *mm*/*dd*/*yyyy*.

`-F` *format*
Specify the output format of the network usage history information. `gnuplot` is the only supported format.

`-s` *time*
`-e` *time*
Specify start and stop times for data display. Time is in the format *MM*/*DD*/*YYYY*,*hh*:*mm*:*ss*. *hh* uses 24-hour clock notation.

`dlstat show-aggr` [`-r`] [`-t`] [`-Z`] [`-i` *interval*] [`-p`] [`-o` *field*[,...]] [`-u` `R`|`K`|`M`|`G`|`T`|`P`] [`-z` *zone*[,...]] [*link*]
Display per-port statistics for an aggregation.

The options for this subcommand are described under "Options," above.

`dlstat show-ether -P` *protocol* [`-Z`] [`-I` *interval*] [`-p`] [`-o` *field*[,...]] [`-u` R|K|M|G|T|P] [*ether-link*]
Display statistics for a given Ethernet protocol on a link. Supported IEEE protocols include `vdp`, the VSI Discovery and Configuration Protocol and `ecp`, Edge Control Protocol.

VDP statistics can be obtained on VNICs or a physical link. The VDP statistics for a physical link is the cumulative statistics of all the VNICs over it.

ECP statistics can be obtained for a physical link.

Fields displayed for VDP include:

`LINK`
The name of the link.

`IPKTS`
The number of inbound VDP packets.

`OPKTS`
The number of outbound VDP packets.

`KeepAlives`
The number KEEP-ALIVE packets transmitted.

Fields displayed for ECP include:

`LINK`
The name of the link.

`IPKTS`
The number of inbound ECP packets.

`IERRORS`
The number of inbound ECP packets in error.

`OPKTS`
The number of outbound ECP packets.

`OERRORS`
The number of errors when transmitting an ECP packet.

`RETRANSMITS`
The number of packets retransmitted.

`TIMEOUTS`
The number of timeouts, that is, the number of packets not acknowledged by the peer.

`help` [*subcommand-name*]
Displays all the supported `dlstat` subcommands or usage for the given subcommand. If you invoke help for a specific subcommand, the command syntax is displayed, along with an example. Using `dlstat` help without any argument displays all of the subcommands.

Examples

EXAMPLE 1 Displaying Statistics

To display statistics for all the links, enter following command. Statistics are displayed as 3-digits followed by decimal and then 2 digits with the appropriate unit.

```
# dlstat
      LINK   IPKTS  RBYTES   OPKTS  OBYTES
   e1000g0 101.88K  32.86M  40.16K   4.37M
     nxge1   4.50M   6.78G   1.38M  90.90M
     vnic1       8     336       0       0
      net0  73.96K   6.81M       0       0
zone1/net0 144.47K  13.32M     247  16.29K
zone2/net0 132.89K  12.25M     236  15.82K
```

EXAMPLE 2 Displaying RX-side Statistics

The following command displays receive-side statistics every one second.

```
# dlstat -r -i 1
    LINK   IPKTS  RBYTES   INTRS   POLLS   IDROPS
e1000g0 101.91K  32.86M  87.56K  14.35K         0
  nxge1   9.61M  14.47G   5.79M   3.82M         0
  vnic1       8     336       0       0         0
e1000g0       0       0       0       0         0
  nxge1  82.13K 123.69M  50.00K  32.13K         0
```

EXAMPLE 2 Displaying RX-side Statistics *(Continued)*

```
  vnic1       0       0       0       0         0
      .       .       .       .       .         .
      .       .       .       .       .         .
```

EXAMPLE 3 Displaying Statistics per Physical Device

The following command displays statistics for a specific physical device.

```
# dlstat show-phys ixgbe0
   LINK   IPKTS  RBYTES   INTRS   POLLS
e1000g0 101.91K  32.86M  87.56K  14.35K
  nxge1   9.61M  14.47G   5.79M   3.82M
  vnic1       8     336       0       0
e1000g0       0       0       0       0
  nxge1  82.13K 123.69M  50.00K  32.13K
  vnic1       0       0       0       0
      .       .       .       .       .
      .       .       .       .       .
```

EXAMPLE 4 Displaying Statistics per Datalink

The following command displays statistics for a specific datalink.

```
# dlstat show-link ixgbe0
  LINK    IPKTS   RBYTES    OPKTS   OBYTES
ixgbe0    2.14M  257.48M    3.19M  210.88M
```

EXAMPLE 5 Displaying Statistics per Hardware Ring

The following commands displays statistics on a per receive-side hardware ring basis.

```
# dlstat show-phys -r nxge1
 LINK TYPE   INDEX   IPKTS  RBYTES
nxge1   rx       0       0       0
nxge1   rx       1       0       0
nxge1   rx       2   1.73M   2.61G
nxge1   rx       3       0       0
nxge1   rx       4   8.44M  12.71G
nxge1   rx       5   5.68M   8.56G
nxge1   rx       6   4.90M   7.38G
nxge1   rx       7       0       0
```

EXAMPLE 6 Displaying Statistics per Lane

The following commands displays statistics on a per receive-side lane basis. First, an interface with dedicated hardware lanes:

```
# dlstat show-link -r nxge1
 LINK TYPE     ID INDEX   IPKTS  RBYTES   INTRS   POLLS  IDROPS
nxge1   rx  local    --       0       0       0       0       0
```

EXAMPLE 6 Displaying Statistics per Lane *(Continued)*

```
nxge1   rx     hw     1       0       0       0       0       0
nxge1   rx     hw     2    1.73M    2.61G    1.33M 400.22K       0
nxge1   rx     hw     3       0       0       0       0       0
nxge1   rx     hw     4    8.44M   12.71G    4.35M    4.09M       0
nxge1   rx     hw     5    5.68M    8.56G    3.72M    1.97M       0
nxge1   rx     hw     6    4.90M    7.38G    3.11M    1.80M       0
nxge1   rx     hw     7       0       0       0       0       0
```

Then, an interface without dedicated hardware lanes, that is, a software lane only:

```
# dlstat show-link -r ixgbe0
  LINK TYPE     ID INDEX   IPKTS  RBYTES   INTRS   POLLS IDROPS
ixgbe0   rx  local    --       0       0       0       0      0
ixgbe0   rx     sw    -- 794.28K   1.19G 794.28K       0      0
```

EXAMPLE 7 Displaying Transmit-Side Statistics

The following command displays transmit-side statistics at five-second intervals.

```
# dlstat -t -i 5
  LINK   OPKTS  OBYTES  ODROPS
e1000g0  40.24K   4.37M      0
  nxge1   9.76M 644.14M      0
  vnic1       0       0      0
e1000g0       0       0      0
  nxge1  26.82K   1.77M      0
  vnic1       0       0      0
      .       .       .      .
      .       .       .      .
      .       .       .      .
```

EXAMPLE 8 Displaying Transmit-Side Ring Statistics

The following command displays transmit-side hardware ring statistics.

```
# dlstat show-phys -t nxge1
 LINK TYPE INDEX   OPKTS  OBYTES
nxge1   tx     0      44   3.96K
nxge1   tx     1       0       0
nxge1   tx     2   1.48M 121.68M
nxge1   tx     3   2.45M 201.11M
nxge1   tx     4   1.47M 120.82M
nxge1   tx     5       0       0
nxge1   tx     6   1.97M 161.57M
nxge1   tx     7   4.59M 376.21M
nxge1   tx     8   2.43M 199.24M
nxge1   tx     9       0       0
nxge1   tx     10  3.23M 264.69M
```

EXAMPLE 8 Displaying Transmit-Side Ring Statistics *(Continued)*

```
nxge1   tx     11  1.88M 153.96M
```

EXAMPLE 9 Displaying Transmit-Side Lane Statistics

The following command displays transmit-side lane statistics.

```
# dlstat show-link -t nxge1
 LINK TYPE     ID INDEX   OPKTS  OBYTES  ODROPS
nxge1   tx     hw     0      32   1.44K       0
nxge1   tx     hw     1       0       0       0
nxge1   tx     hw     2   1.48M  97.95M       0
nxge1   tx     hw     3   2.45M 161.87M       0
nxge1   tx     hw     4   1.47M  97.25M       0
nxge1   tx     hw     5       3     276       0
nxge1   tx     hw     6   1.97M 130.05M       0
nxge1   tx     hw     7   4.59M 302.80M       0
nxge1   tx     hw     8   2.42M 302.80M       0
nxge1   tx     hw     9       0       0       0
nxge1   tx     hw     10  3.23M 213.05M       0
nxge1   tx     hw     11  1.88M 123.93M       0
```

EXAMPLE 10 Displaying Both RX and TX Lane Statistics

The following command displays both receive-side and transmit-side lane statistics.

```
# dlstat show-link -rt nxge0
 LINK  TYPE      ID  INDEX     PKTS    BYTES
nxge0    rx   local     --        0        0
nxge0    rx   other     --        0        0
nxge0    rx      hw      0        0        0
nxge0    rx      hw      1        0        0
nxge0    rx      hw      2        0        0
nxge0    rx      hw      3        0        0
nxge0    rx      hw      4        0        0
nxge0    rx      hw      5        0        0
nxge0    rx      hw      6        0        0
nxge0    rx      hw      7        0        0
nxge0    tx   local     --        0        0
nxge0    tx   other     --        3      126
nxge0    tx      hw      0        0        0
nxge0    tx      hw      1        0        0
nxge0    tx      hw      2        0        0
nxge0    tx      hw      3        0        0
nxge0    tx      hw      4        0        0
nxge0    tx      hw      5        0        0
nxge0    tx      hw      6        0        0
nxge0    tx      hw      7        0        0
nxge0    tx      hw      8        0        0
```

EXAMPLE 10 Displaying Both RX and TX Lane Statistics *(Continued)*

```
nxge0    tx      hw      9        0        0
nxge0    tx      hw     10        0        0
nxge0    tx      hw     11        0        0
```

EXAMPLE 11 Selecting a Particular Set of Statistics

The following command shows how you can select a set of statistics of particular interest.

```
# dlstat show-link -r -o LINK,TYPE,ID,INDEX,INTRS,POLLS nxge1
 LINK TYPE     ID INDEX   INTRS   POLLS
nxge1   rx  local    --       0       0
nxge1   rx  other    --       0       0
nxge1   rx     hw     1       0       0
nxge1   rx     hw     2   2.47M 753.90K
nxge1   rx     hw     3       0       0
nxge1   rx     hw     4   8.24M   7.72M
nxge1   rx     hw     5   6.96M   3.68M
nxge1   rx     hw     6   5.82M   3.36M
nxge1   rx     hw     7       0       0
```

EXAMPLE 12 Displaying Historical Network Usage

Network usage history statistics can be stored by using the extended accounting facility, acctadm(1M), with a command such as the following:

```
# acctadm -e basic -f /var/log/net.log net
acctadm net
Network accounting: active
    Network accounting file: /var/log/net.log
  Tracked Network resources: basic
Untracked Network resources: src_ip,dst_ip,src_port,dst_port,protocol,
                             dsfield
```

The saved historical data can then be retrieved in summary form with commands such as the following:

```
# dlstat show-link -h -f /var/log/net.log
LINK      DURATION  IPACKETS RBYTES      OPACKETS OBYTES      BANDWIDTH
e1000g0   80        1031     546908      0        0           2.44 Kbps

# dlstat show-ether -P vdp ixgbe1
LINK          IPKTS    OPKTS  KeepAlives
ixgbe1          3       2       1

# dlstat show-ether -P ecp ixgbe1
LINK          IPKTS    OPKTS  IERRORS  OERRORS RETRANSMITS TIMEOUTS
ixgbe1          3       2       0       0       1           0
```

EXAMPLE 13 Displaying Help

The following command lists all of the `dlstat` subcommands.

```
# dlstat help
The following subcommands are supported:
Stats subcommands : show-aggr, show-link, show-phys
For more info, run: dlstat help subcommand
```

The following command illustrates the use of `dlstat help` with a specific subcommand.

```
# dlstat help show-phys
usage:
show-phys    [-r] [-t] [-Z] [-i interval] [-a]
             [-p] [-o field[,...]] [-u R|K|M|G|T|P]
             [-z zone[,...]] [link]

example:
     # dlstat show-phys -r -o all -u K net0
```

Attributes

See `attributes(5)` for descriptions of the following attributes:

`/usr/sbin`

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os
Interface Stability	See below

Screen output is Uncommitted. The invocation is Committed.

See Also

`acctadm(1M)`, `dladm(1M)`, `ifconfig(1M)`, `kstat(1M)`, `netstat(1M)`, `attributes(5)`

Name dmesg – collect system diagnostic messages to form error log

Synopsis

```
/usr/bin/dmesg

/usr/sbin/dmesg
```

Description `dmesg` is made obsolete by `syslogd(1M)` for maintenance of the system error log.

`dmesg` looks in a system buffer for recently printed diagnostic messages and prints them on the standard output.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also `syslogd(1M)`, `attributes(5)`

Name dminfo – report information about a device entry in a device maps file

Synopsis

```
dminfo [-v] [-a] [-f pathname]
dminfo [-v] [-a] [-f pathname] -n dev -name...
dminfo [-v] [-a] [-f pathname] -d dev -path...
dminfo [-v] [-a] [-f pathname] -t dev -type...
dminfo [-v] [-f pathname] -u dm -entry
```

Description `dminfo` reports and updates information about the `device_maps(4)` file.

Options The following options are supported

`-a`	Succeed if any of the requested entries are found. If used with `-v`, all entries that match the requested case(s) are printed.
`-d` *dev–path*	Search by *dev–path*. Search `device_maps(4)` for a device special pathname in the *device_list* field matching the *dev–path* argument. This option cannot be used with `-n`, `-t` or `-u`.
`-f` *pathname*	Use a device_maps file with *pathname* instead of `/etc/security/device_maps`.
`-n` *dev–name*	Search by *dev–name*. Search `device_maps(4)` for a *device_name* field matching *dev–name*. This option cannot be used with `-d`, `-t` or `-u`.
`-t` *dev–type*	Search by *dev–type*. Search `device_maps(4)` for a *device_type* field matching the given *dev–type*. This option cannot be used with `-d`, `-n` or `-u`.
`-u` *dm–entry*	Update the `device_maps(4)` file. This option is provided to add entries to the `device_maps(4)` file. The *dm–entry* must be a complete `device_maps(4)` file entry. The *dm–entry* has fields, as in the `device_maps` file. It uses the colon (`:`) as a field separator, and white space as the *device_list* subfield separators. The *dm–entry* is not made if any fields are missing, or if the *dm–entry* would be a duplicate. The default device maps file can be updated only by the super user.
`-v`	Verbose. Print the requested entry or entries, one line per entry, on the standard output. If no entries are specified, all are printed.

Exit Status

0	Successful completion.
1	Request failed.
2	Incorrect syntax.

Files `/etc/security/device_maps`

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also device_maps(4), attributes(5)

Name dns-sd – Multicast DNS (mDNS) & DNS Service Discovery (DNS-SD) Test Tool

Synopsis

```
dns-sd -R name type domain port [key=value ...]
dns-sd -B type domain
dns-sd -L name type domain
dns-sd -Q FQDN rrtype rrclass
dns-sd -C FQDN rrtype rrclass
dns-sd -P name type domain port host IP [key=value ...]
dns-sd -E | -F | -A | -U | -N | -T | -M | -I
```

Description The `dns-sd` command is a network diagnostic tool, much like `ping(1M)` or `traceroute(1M)`. However, unlike those tools, most of its functionality is not implemented in the `dns-sd` executable itself, but in library code that is available to any application. The library API that `dns-sd` uses is documented in `/usr/include/dns_sd.h`.

The `dns-sd` command is primarily intended for interactive use. Because its command-line arguments and output format are subject to change, invoking it from a shell script can be unpredictable. Additionally, the asynchronous nature of DNS Service Discovery does not easily lend itself to script-oriented programming. This style of asynchronous interaction works best with applications that are either multi-threaded, or use a main event-handling loop to receive keystrokes, network data, and other asynchronous event notifications as they happen.

Options The following options are supported:

`-R` *name type domain port* [*key=value* ...]
Register (advertise) a service in the specified domain with the given *name* and *type* as listening (on the current machine) on the specified *port*.

name can be any arbitrary unicode text, containing any legal unicode characters (including dots, spaces, slashes, colons, and so on without any restrictions), up to 63 UTF-8 bytes long.

type must be of the form "_app-proto._tcp" or "_app-proto._udp", where "app-proto" is an application protocol name registered at `http://www.dns-sd.org`, under the service types (RFC 2782) link.

domain is the domain in which to register the service. In current implementations, only the local multicast domain "local" is supported. In the future, registering will be supported in any arbitrary domain that has a working DNS Update server [RFC 2136]. The domain "." is a synonym for "pick a sensible default", which currently means "local".

port is a number from 0 to 65535, and is the TCP or UDP port number upon which the service is listening. Registering a service on port 0 allows an application to explicitly advertise the non-availability of a service.

Additional attributes of the service may optionally be described by *key/value* pairs, which are stored in the advertised service's DNS TXT record. Allowable keys and values are listed with the service registration at `http://www.dns-sd.org`, under the service types (RFC 2782) link.

`-B` *type domain*
Browse for instances of service *type* in *domain*.

For valid types, see `http://www.dns-sd.org`, under the service types (RFC 2782) link. Omitting the domain name or using "." means "pick a sensible default."

`-L` *name type domain*
Look up and display the information necessary to contact and use the named service. This information includes the hostname of the machine where that service is available, the port number on which the service is listening, and (if present) TXT record attributes describing properties of the service.

In a typical application, browsing happens rarely, while lookup (or "resolving") happens every time the service is used. For example, a user does not browse the network to pick a default printer that often, but once a default printer has been picked, that named service is resolved to its current IP address and port number every time the user presses Cmd-P to print.

`-Q` *FQDN rrtype rrclass*
Generic query for any resource record type and class.

`-C` *FQDN rrtype rrclass*
Generic query for any resource record type and class. This option also reconfirms each result from the query. Reconfirming the record instructs `mdnsd(1M)` to verify the validity of the record. If the record is not valid `mdnsd(1M)` flushes the record from the daemon's cache and also from other `mdnsd(1M)` caches on the network.

`-P` *name type domain port host IP* [*key=value* ...]
Register (advertise) a service in the specified domain with the given *name* and *type* listening on the specified port and accessible on another host. This option should be used to advertise by proxy a service accessible on another host. The host name and IPv4 address to access the service must be specified.

`-E`
Discover recommended registration domains. This option returns the recommended domains to register a service. The recommended registration domains are returned by querying the name servers in `resolv.conf(4)`.

`-F`
Discover recommended browsing domains. This option returns the recommended domains for browsing services. The recommended browsing domains are returned by querying the name servers in `resolv.conf(4)`.

`-A`
Test registering service with Multicast DNS and test the add, update and delete operations of DNS records with Multicast DNS.

`-U`
Test registering service with Multicast DNS and test updating of DNS TXT records for a service registered with Multicast DNS.

`-N`
Test adding a large NULL record for a service registered with Multicast DNS.

`-T`
Test adding a large TXT record for a service registered with Multicast DNS.

`-M`
Test creating a registration with multiple TXT records.

`-I`
Test registering and then immediately updating a TXT record.

Examples

EXAMPLE 1 Advertising a printing service

The following command advertises the existence of LPR printing service on port 515 on this machine, so that it will be available to DNS-SD compatible printing clients:

```
dns-sd -R "My Test" _printer._tcp. . 515 pdl=application/postscript
```

For this registration to be useful, the LPR service should be available on port 515. Advertising a service that does not exist is not very useful.

EXAMPLE 2 Advertising a web page

The following command advertises a web page being served by an HTTP server on port 80 on this machine, so that it will appear on the Bonjour list in Safari and other DNS-SD compatible Web clients:

```
dns-sd -R "My Test" _http._tcp . 80 path=/path-to-page.html
```

EXAMPLE 3 Finding the advertised web pages on the local network

The following command finds the advertised web pages on the local network (the same list that Safari shows):

```
dns-sd -B _http._tcp
```

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	service/network/dns/mdns

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Volatile

See Also mdnsd(1M), ping(1M), ping(1M), resolv.conf(4), attributes(5)

Name dnssec-dsfromkey – DNSSEC DS RR generation tool

Synopsis

```
dnssec-dsfromkey [-v level] [-1] [-2] [-a alg] keyfile

dnssec-dsfromkey -s [-v level] [-1] [-2] [-a alg] [-c class]
     [-d dir] keyfile
```

Description `dnssec-dsfromkey`

Options The following options are supported:

`-1`
Use `SHA-1` as the digest algorithm. The default is to use both `SHA-1` and `SHA-256`.

`-2`
Use SHA-256 as the digest algorithm.

`-a` *algorithm*
Select the digest algorithm. The value of *algorithm* must be one of `SHA-1` (`SHA1`) or `SHA-256` (`SHA256`). These values are case-insensitive.

`-v` *level*
Sets the debugging level.

`-s`
Keyset mode: in place of the keyfile name, the argument is the DNS domain name of a keyset file. The `-c` and `-d` options have meaning only in this mode.

`-c` *class*
Specifies the DNS class (default is `IN`); useful only in the keyset mode.

`-d` *directory*
Look for keyset files in directory as the directory; ignored when not in the keyset mode.

Examples To build the SHA-256 DS RR from the `Kexample.com.+003+26160` keyfile name, use a command such as the following:

```
# dnssec-dsfromkey -2 Kexample.com.+003+26160
```

This command would produce output similar to the following:

```
example.com. IN DS 26160 5 2
3A1EADA7A74B8D0BA86726B0C227AA85AB8BBD2B2004F41A868A54F0
C5EA0B94
```

Files The keyfile can be designated by the key identification K*nnnn*.+*aaa*+*iiiii*, or the full file name K*nnnn*.+*aaa*+*iiiii*.`key`, as generated by `dnssec-keygen(1M)`.

The keyset file name is built from the directory, the string `keyset-` and the *dnsname*.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	service/network/dns/bind
Interface Stability	Volatile

See Also dnssec-keygen(1M), dnssec-signzone(1M), attributes(5)

RFC 3658, *RFC 4509*

See the BIND 9 *Administrator's Reference Manual*. As of the date of publication of this man page, this document is available at https://www.isc.org/software/bind/documentation.

Caution A keyfile error can produce a "file not found" message, even if the file exists.

Name dnssec-keyfromlabel – DNSSEC key generation tool

Synopsis

```
dnssec-keyfromlabel -a algorithm -l label [-c class] [-f flag] [-k]
    [-n nametype] [-p protocol] [-t type] [-v level] name
```

Description `dnssec-keyfromlabel` retrieves keys with a specified label from a crypto hardware device and builds key files for DNSSEC (Secure DNS), as defined in RFC 2535 and RFC 4034.

Options The following options are supported:

`-a` *algorithm*
Selects the cryptographic algorithm. The value of *algorithm* must be one of `RSAMD5` (RSA) or `RSASHA1`, `DSA`, `NSEC3RSASHA1`, `NSEC3DSA`, or `DH` (Diffie-Hellman). These values are case-insensitive.

Note that for `DNSSEC`, `RSASHA1` is a mandatory-to-implement algorithm, and DSA is recommended. Note also that `DH` automatically sets the `-k` flag.

`-l` *label*
Specifies the label of keys in the crypto hardware (PKCS#11) device.

`-n` *nametype*
Specifies the owner type of the key. The value of *nametype* must either be `ZONE` (for a `DNSSEC` zone key (`KEY/DNSKEY`)), `HOST` or `ENTITY` (for a key associated with a host (`KEY`)), `USER` (for a key associated with a user (`KEY`)), or `OTHER` (`DNSKEY`). These values are case-insensitive.

`-c` *class*
Indicates that the DNS record containing the key should have the specified class. If not specified, class `IN` is used.

`-f` *flag*
Set the specified flag in the flag field of the `KEY/DNSKEY` record. The only recognized flag is `KSK` (Key Signing Key) `DNSKEY`.

`-h`
Displays a short summary of the options and arguments to `dnssec-keyfromlabel`.

`-k`
Generate `KEY` records rather than `DNSKEY` records.

`-p` *protocol*
Sets the protocol value for the generated key. The protocol is a number between 0 and 255. The default is `3` (`DNSSEC`). Other possible values for this argument are listed in RFC 2535 and its successors.

`-t` *type*
Indicates the use of the key. *type* must be one of `AUTHCONF`, `NOAUTHCONF`, `NOAUTH`, or `NOCONF`. The default is `AUTHCONF`. `AUTH` refers to the ability to authenticate data, and `CONF` the ability to encrypt data.

`-v` *level*
Sets the debugging level.

Generated Key Files

When `dnssec-keyfromlabel` completes successfully, it displays a string of the form K*nnnn*.+*aaa*+*iiiii* to the standard output. This is an identification string for the key files it has generated, which translates as follows.

- *nnnn* is the key name.
- *aaa* is the numeric representation of the algorithm.
- *iiiii* is the key identifier (or footprint).

`dnssec-keyfromlabel` creates two files, with names based on the displayed string. K*nnnn*.+*aaa*+*iiiii*.`key` contains the public key, and K*nnnn*.+*aaa*+*iiiii*.`private` contains the private key.

The first file contains a `DNS KEY` record that can be inserted into a zone file (directly or with an `$INCLUDE` statement).

The second file contains algorithm-specific fields. For obvious security reasons, this file does not have general read permission.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	service/network/dns/bind
Interface Stability	Volatile

See Also

`dnssec-keygen(1M)`, `dnssec-signzone(1M)`, `attributes(5)`

RFC 2539, *RFC 2845*, *RFC 4033*

See the BIND 9 *Administrator's Reference Manual*. As of the date of publication of this man page, this document is available at `https://www.isc.org/software/bind/documentation`.

Name dnssec-keygen – DNSSEC key generation tool

Synopsis

```
dnssec-keygen -a algorithm -b keysize -n nametype [-ehk]
     [-c class] [-f flag] [-g generator] [-p protocol]
     [-r randomdev] [-s strength] [-t type] [-v level] name
```

Description The `dnssec-keygen` utility generates keys for DNSSEC (Secure DNS), as defined in RFC 2535 and RFC 4034. It can also generate keys for use with TSIG (Transaction Signatures), as defined in RFC 2845.

Options The following options are supported:

`-a` *algorithm*
Select the cryptographic algorithm. The value of algorithm must be one of RSAMD5 (RSA) or RSASHA1, DSA, NSEC3RSASHA1, NSEC3DSA, `DH` (Diffie-Hellman), or HMAC-MD5. These values are case insensitive.

For DNSSEC, RSASHA1 is a mandatory-to-implement algorithm and DSA is recommended. For TSIG, HMAC-MD5 is mandatory.

Note – HMAC-MD5 and DH automatically set the `-k` flag.

`-b` *keysize*
Specify the number of bits in the key. The choice of key size depends on the algorithm used. RSAMD5 and RSASHA1 keys must be between 512 and 2048 bits. Diffie-Hellman keys must be between 128 and 4096 bits. DSA keys must be between 512 and 1024 bits and an exact multiple of 64. HMAC-MD5 keys must be between 1 and 512 bits.

`-c` *class*
Indicate that the DNS record containing the key should have the specified class. If not specified, class IN is used.

`-e`
Use a large exponent if generating an RSAMD5 or RSASHA1 key.

`-f` *flag*
Set the specified flag in the flag field of the KEY/DNSKEY record. The only recognized flag is KSK (Key Signing Key) DNSKEY.

`-g` *generator*
Use this *generator* if generating a Diffie Hellman key. Allowed values are 2 and 5. If no generator is specified, a known prime from RFC 2539 will be used if possible; otherwise the default is 2.

`-h`
Print a short summary of the options and arguments to `dnssec-keygen`.

`-k`
Generate KEY records rather than DNSKEY records.

`-n` *nametype*
Specify the owner type of the key. The value of *nametype* must either be `ZONE` (for a DNSSEC zone key (KEY/DNSKEY)), `HOST` or `ENTITY` (for a key associated with a host (KEY)), USER (for a key associated with a user(KEY)), or `OTHER` (DNSKEY). These values are case insensitive. Defaults to ZONE for DNSKEY generation.

`-p` *protocol*
Set the protocol value for the generated key. The *protocol* argument is a number between 0 and 255. The default is 3 (DNSSEC) Other possible values for this argument are listed in RFC 2535 and its successors.

`-r` *randomdev*
Specify the source of randomness. If the operating system does not provide a `/dev/random` or equivalent device, the default source of randomness is keyboard input. *randomdev* specifies the name of a character device or file containing random data to be used instead of the default. The special value "`keyboard`" indicates that keyboard input should be used.

`-s` *strength*
Specify the strength value of the key. The *strength* argument is a number between 0 and 15, and currently has no defined purpose in DNSSEC.

`-t` *type*
Indicate the use of the key. `type` must be one of `AUTHCONF`, `NOAUTHCONF`, `NOAUTH`, or `NOCONF`. The default is `AUTHCONF`. `AUTH` refers to the ability to authenticate data, and `CONF` the ability to encrypt data.

`-v` *level*
Set the debugging level.

Generated Keys When `dnssec-keygen` completes successfully, it prints a string of the form K*nnnn*`.+`*aaa*`+`*iiiii* to the standard output. This is an identification string for the key it has generated.

- *nnnn* is the key name.
- *aaa* is the numeric representation of the algorithm.
- *iiiii* is the key identifier (or footprint).

The `dnssec-keygen` utility creates two files, with names based on the printed string.

- K*nnnn*`.+`*aaa*`+`*iiiii*`.key` contains the public key.
- K*nnnn*`.+`*aaa*`+`*iiiii*`.private` contains the private key.

The `.key` file contains a DNS `KEY` record that can be inserted into a zone file (directly or with a `$INCLUDE` statement).

The `.private` file contains algorithm specific fields. For obvious security reasons, this file does not have general read permission.

Both `.key` and `.private` files are generated for symmetric encryption algorithm such as HMAC-MD5, even though the public and private key are equivalent.

Examples

EXAMPLE 1 Generating a 768-bit DSA Key

To generate a 768-bit DSA key for the domain `example.com`, the following command would be issued:

```
dnssec-keygen -a DSA -b 768 -n ZONE example.com
```

The command would print a string of the form:

```
Kexample.com.+003+26160
```

The following files would be created:

```
Kexample.com.+003+26160.key
Kexample.com.+003+26160.private
```

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	service/network/dns/bind
Interface Stability	Volatile

See Also

`dnssec-signzone(1M)`, `attributes(5)`

RFC 2539, *RFC 2845*, *RFC 4033*

See the BIND 9 *Administrator's Reference Manual*. As of the date of publication of this man page, this document is available at `https://www.isc.org/software/bind/documentation`.

Name dnssec-makekeyset – DNSSEC zone signing tool

Synopsis

```
dnssec-makekeyset [-ahp] [-s start-time] [-e end-time]
     [-r randomdev] [-t ttl] [-v level] key...
```

Description The `dnssec-makekeyset` utility generates a key set from one or more keys created by `dnssec-keygen(1M)`. It creates a file containing a `KEY` record for each key, and self-signs the key set with each zone key. The output file is of the form *keyset-nnnn.*, where *nnnn* is the zone name.

Options

`-a` Verify all generated signatures.

`-e` *end-time* Specify the date and time when the generated SIG records expire. As with *start-time*, an absolute time is indicated in `YYYYMMDDHHMMSS` notation. A time relative to the start time is indicated with +*N*, which is *N* seconds from the start time. A time relative to the current time is indicated with now+*N*. If no *end-time* is specified, 30 days from the start time is used as a default.

`-h` Print a short summary of the options and arguments to `dnssec-makekeyset()`.

`-p` Use pseudo-random data when signing the zone. This is faster, but less secure, than using real random data. This option may be useful when signing large zones or when the entropy source is limited.

`-r` *randomdev* Specify the source of randomness. If the operating system does not provide a `/dev/random` or equivalent device, the default source of randomness is keyboard input. The *randomdev* argument specifies the name of a character device or file containing random data to be used instead of the default. The special value `keyboard` indicates that keyboard input should be used.

`-s` *start-time* Specify the date and time when the generated `SIG` records become valid. This can be either an absolute or relative time. An absolute start time is indicated by a number in `YYYYMMDDHHMMSS` notation; 20000530144500 denotes 14:45:00 UTC on May 30th, 2000. A relative start time is indicated by +N, which is N seconds from the current time. If no `start-time` is specified, the current time is used.

`-t` *ttl* Specify the TTL (time to live) of the `KEY` and `SIG` records. The default is 3600 seconds.

`-v` *level* Set the debugging level.

Operands The following operands are supported:

key The list of keys to be included in the keyset file. These keys are expressed in the form K*nnnn*.+*aaa*+*iiiii* as generated by `dnssec-keygen`.

Examples

EXAMPLE 1 Generates a keyset containing the DSA key for example.com.

The following command generates a keyset containing the DSA key for `example.com` generated in the `dnssec-keygen(1M)` manual page.

```
dnssec-makekeyset -t 86400 -s 20000701120000 -e +2592000 \
Kexample.com.+003+26160
```

In this example, `dnssec-makekeyset()` creates the file `keyset-example.com`. This file contains the specified key and a self-generated signature.

The DNS administrator for `example.com` could send `keyset-example.com.` to the DNS administrator for `.com` for signing, if the .com zone is DNSSEC-aware and the administrators of the two zones have some mechanism for authenticating each other and exchanging the keys and signatures securely.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWbind9
Interface Stability	Volatile

See Also

`dnssec-keygen(1M)`, `dnssec-signkey(1M)`, `attributes(5)`

RFC 2535

BIND 9 Administrator Reference Manual

Notes

Source for BIND9 is available in the SUNWbind9S package.

Name dnssec-signkey – DNSSEC key set signing tool

Synopsis

```
dnssec-signkey [-ahp] [-c class] [-e end-time]
     [-r randomdev] [-s start-time] [-v level] keyset key...
```

Description The `dnssec-signkey` utility signs a keyset. Typically the keyset will be for a child zone and will have been generated by `dnssec-makekeyset(1M)`. The child zone's keyset is signed with the zone keys for its parent zone. The output file is of the form `signedkey-`*nnnn*`.`, where *nnnn* is the zone name.

Options The following options are supported:

`-a` Verify all generated signatures.

`-c` *class* Specify the DNS class of the key sets.

`-e` *end-time* Specify the date and time when the generated SIG records expire. As with *start-time*, an absolute time is indicated in `YYYYMMDDHHMMSS` notation. A time relative to the start time is indicated with +*N*, which is *N* seconds from the start time. A time relative to the current time is indicated with now+*N*. If no *end-time* is specified, 30 days from the start time is used as a default.

`-h` Prints a short summary of the options and arguments to `dnssec-signkey()`.

`-p` Use pseudo-random data when signing the zone. This is faster, but less secure, than using real random data. This option may be useful when signing large zones or when the entropy source is limited.

`-r` *randomdev* Specify the source of randomness. If the operating system does not provide a `/dev/random` or equivalent device, the default source of randomness is keyboard input. *randomdev* specifies the name of a character device or file containing random data to be used instead of the default. The special value `keyboard` indicates that keyboard input should be used.

`-s` *start-time* Specify the date and time when the generated SIG records become valid. This can be either an absolute or relative time. An absolute start time is indicated by a number in `YYYYMMDDHHMMSS` notation; 20000530144500 denotes 14:45:00 UTC on May 30th, 2000. A relative start time is indicated by +*N*, which is *N* seconds from the current time. If no *start-time* is specified, the current time is used.

`-v` *level* Set the debugging level.

Operands The following operands are supported:

key The keys used to sign the child's keyset.

keyset The file containing the child's keyset.

Examples

EXAMPLE 1 Sign the *keyset* file for example.com.

The DNS administrator for a DNSSEC-aware `.com` zone would use the following command to sign the *keyset* file for `example.com` created by `dnssec-makekeyset` with a key generated by `dnssec-keygen`:

```
dnssec-signkey keyset-example.com. Kcom.+003+51944
```

In this example, `dnssec-signkey` creates the file `signedkey-example.com`, which contains the `example.com` keys and the signatures by the `.com` keys.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWbind9
Interface Stability	Volatile

See Also

`dnssec-keygen(1M)`, `dnssec-makekeyset(1M)`, `dnssec-signzone(1M)`, `attributes(5)`

Notes

Source for BIND9 is available in the SUNWbind9S package.

Name dnssec-signzone – DNSSEC zone signing tool

Synopsis

```
dnssec-signzone [-Aaghptz] [-c class] [-d directory]
     [-e end-time] [-f output-file] [-H iterations] [-I input_format]
     [-i interval] [-k key] [-l domain] [-N soa-serial-format] [-n ncpus]
     [-O output_format] [-o origin] [-r randomdev] [-s start-time]
     [-v level] [-3 salt] zonefile [key]...
```

Description The `dnssec-signzone` utility signs a zone. It generates `NSEC` and `RRSIG` records and produces a signed version of the zone. The security status of delegations from the signed zone (that is, whether the child zones are secure or not) is determined by the presence or absence of a `keyset` file for each child zone.

Options The following options are supported:

`-A`
When generating an NSEC3 chain, set the `OPTOUT` flag on all NSEC3 records and do not generate NSEC3 records for insecure delegations.

`-a`
Verify all generated signatures.

`-c` *class*
Specify the `DNS` class of the zone.

`-d` *directory*
Look for `keyset` files in *directory*.

`-e` *end-time*
Specify the date and time when the generated `RRSIG` records expire. As with `start-time`, an absolute time is indicated in `YYYYMMDDHHMMSS` notation. A time relative to the start time is indicated with +*N*, which is *N* seconds from the start time. A time relative to the current time is indicated with `now`+*N*. If no *end-time* is specified, 30 days from the start time is used as a default.

`-f` *output-file*
The name of the output file containing the signed zone. The default is to append `.signed` to the input file name.

`-g`
Generate DS records for child zones from `keyset` files. Existing DS records will be removed.

`-H` *iterations*
When generating a NSEC3 chain use the number of interations specified by *iterations*. The default is 100.

`-h`
Prints a short summary of the options and arguments to `dnssec-signzone()`.

`-I` *input-format*
The format of the input zone file. Possible formats are `text` (default) and `raw`. This option is primarily intended for dynamic signed zones so that the dumped zone file in a non-text format containing updates can be signed directly. The use of this option serves no purpose for non-dynamic zones.

`-i` *interval*
Specify the cycle interval as an offset from the current time (in seconds). When a previously signed zone is passed as input, records could be resigned. If an `RRSIG` record expires after the cycle interval, it is retained. Otherwise, it is considered to be expiring soon and will be replaced.

The default cycle interval is one quarter of the difference between the signature end and start times. If neither *end-time* or *start-time* are specified, `dnssec-signzone` generates signatures that are valid for 30 days, with a cycle interval of 7.5 days. Any existing `RRSIG` records due to expire in less than 7.5 days would be replaced.

`-j` *jitter*
When signing a zone with a fixed signature lifetime, all `RRSIG` records issued at the time of signing expire simultaneously. If the zone is incrementally signed, that is, a previously-signed zone is passed as input to the signer, all expired signatures have to be regenerated at about the same time. The jitter option specifies a jitter window that will be used to randomize the signature-expire time, thus spreading incremental signature regeneration over time.

Signature lifetime jitter also benefits, to some extent, validators and servers by spreading out cache expiration. That is, if large numbers of `RRSIG`s from all caches do not expire at the same time, there will be less congestion than if all validators needed to refetch at almost the same time.

`-k` *key*
Treat specified *key* as a key-signing key, ignoring any key flags. This option can be specified multiple times.

`-l` *domain*
Generate a DLV set in addition to the key (DNSKEY) and DS sets. The domain is appended to the name of the records.

`-N` *soa-serial-format*
The SOA serial number format of the signed zone. Possible formats are `keep` (default), `increment` and `unixtime`, described as follows.

`keep`
Do not modify the SOA serial number.

`increment`
Increment the SOA serial number using RFC 1982 arithmetic.

`unixtime`
Set the SOA serial number to the number of seconds since epoch.

`-n` *nthreads*
Specifies the number of threads to use. By default, one thread is started for each detected CPU.

`-O` *output_format*
The format of the output file containing the signed zone. Possible formats are `text` (default) and `raw`.

`-o` *origin*
Specify the zone origin. If not specified, the name of the zone file is assumed to be the origin.

`-p`
Use pseudo-random data when signing the zone. This is faster, but less secure, than using real random data. This option may be useful when signing large zones or when the entropy source is limited.

`-r` *randomdev*
Specifies the source of randomness. If the operating system does not provide a `/dev/random` or equivalent device, the default source of randomness is keyboard input. *randomdev* specifies the name of a character device or file containing random data to be used instead of the default `/dev/random`. The special value `keyboard` indicates that keyboard input should be used.

`-s` *start-time*
Specify the date and time when the generated `RRSIG` records become valid. This can be either an absolute or relative time. An absolute start time is indicated by a number in *YYYYMMDDHHMMSS* notation; 20000530144500 denotes 14:45:00 UTC on May 30th, 2000. A relative start time is indicated by *+N*, which is *N* seconds from the current time. If no *start-time* is specified, the current time minus one hour (to allow for clock skew) is used.

`-t`
Print statistics at completion.

`-v` *level*
Set the debugging level.

`-z`
Ignore KSK flag on key when determining what to sign.

`-3` *salt*
Generate a NSEC3 chain with the specified hex-encoded *salt*. A dash (`-`) can be used to indicate that no salt is to be used when generating the NSEC3 chain.

Operands The following operands are supported:

zonefile
The file containing the zone to be signed.

key
Specify which keys should be used to sign the zone. If no keys are specified, then the zone will be examined for `DNSKEY` records at the zone apex. If these are found and there are matching private keys in the current directory, these will be used for signing.

Examples **EXAMPLE 1** Signing a Zone with a DSA Key

The following command signs the `example.com` zone with the DSA key generated in the example in the `dnssec-keygen(1M)` manual page (`Kexample.com.+003+17247`). The zone's keys must be in the master file (`db.example.com`). This invocation looks for keyset files in the current directory, so that DS records can be generated from them (`-g`).

```
% dnssec-signzone -g -o example.com db.example.com \
Kexample.com.+003+17247
db.example.com.signed
%
```

In the above example, `dnssec-signzone` creates the file `db.example.com.signed`. This file should be referenced in a zone statement in a `named.conf` file.

EXAMPLE 2 Re-signing a Previously Signed Zone

The following commands re-sign a previously signed zone with default parameters. The private keys are assumed to be in the current directory.

```
% cp db.example.com.signed db.example.com
% dnssec-signzone -o example.com db.example.com \
db.example.com.signed
%
```

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	service/network/dns/bind
Interface Stability	Volatile

See Also `dnssec-keygen(1M)`, `attributes(5)`

RFC 4033

See the BIND 9 *Administrator's Reference Manual*. As of the date of publication of this man page, this document is available at `https://www.isc.org/software/bind/documentation`.

Name domainname – set or display name of the current domain

Synopsis `domainname [`*name-of-domain*`]`

Description Without an argument, `domainname` displays the name of the current domain name used in RPC exchanges, usually referred to as the NIS domain name. This name typically encompasses a group of `hosts` or `passwd` entries under the same administration. The `domainname` command is used by various components of Solaris to resolve names for entries such as are found in `passwd`, `hosts` and `aliases`. By default, naming services such as NIS use `domainname` to resolve names.

With the "Name Service Management" `rbac(5)` profile, you can permanently set the name of the domain with the following command:

```
% domainname nisdomain.example.com
```

If not yet enabled, the `nis/domain` service is enabled.

The super-user can temporary set the domain name using:

```
# domainname -t nisdomain.example.com
```

The domain name for various naming services can also be set by other means. DNS ignores the domain name set by `domainname` and LDAP uses it as a last resort.

The `sendmail(1M)` daemon, as shipped with Solaris, and the `sendmail` implementation provided by `sendmail.org` (formerly referred to as "Berkeley 8.x sendmail") both attempt to determine a local host's fully qualified host name at startup and both pursue follow-up actions if the initial search fails. It is in these follow-up actions that the two implementations differ.

Both implementations use a standard Solaris or Unix system call to determine its fully qualified host name at startup, following the name service priorities specified in `nsswitch.conf(4)`. To this point, the Solaris and `sendmail.org` versions behave identically.

If the request for a fully qualified host name fails, the `sendmail.org` `sendmail` sleeps for 60 seconds, tries again, and, upon continuing failure, resorts to a short name. The Solaris version of `sendmail` makes the same initial request, but then, following initial failure, calls `domainname`. If successful, the sleep is avoided.

On a Solaris machine, if you run the `sendmail.org` version of sendmail, you get the startup behavior (omitting the `domainname` call) described above. If you run the Solaris `sendmail`, the `domainname` call is made if needed.

If the Solaris `sendmail` cannot determine the fully qualified host name, use `check-hostname(1M)` as a troubleshooting aid. This script can offer guidance as to appropriate corrective action.

Files

- `/etc/defaultdomain`
- `/etc/nsswitch.conf`

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also

`svcs(1)`, `check-hostname(1M)`, `hostconfig(1M)`, `named(1M)`, `sendmail(1M)`, `svcadm(1M)`, `svccfg(1M)`, `ypinit(1M)`, `aliases(4)`, `defaultdomain(4)`, `hosts(4)`, `nsswitch.conf(4)`, `passwd(4)`, `attributes(5)`, `rbac(5)`, `smf(5)`

Notes

The `domainname` service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/system/identity:domain
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. The service's status can be queried using the `svcs(1)` command.

Name drd – Logical Domain Dynamic Reconfiguration daemon

Synopsis

```
/usr/lib/ldoms/drd
```

Description The `drd` daemon is part of the framework that enables the addition and removal of resources from a Logical Domain. This framework is collectively called Dynamic Reconfiguration (DR).

`drd` is responsible for various aspects of DR on a Logical Domain and must be enabled to ensure proper DR functionality. It is started at boot time and has no configuration options.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/ldoms
Interface Stability	Uncommitted

See Also `svcs(1)`, `svcadm(1M)`, `syslog(3C)`, `syslog.conf(4)`, `attributes(5)`, `smf(5)`

Errors `drd` uses `syslog(3C)` to report status and error messages. All of the messages are logged with the `LOG_DAEMON` facility. Error messages are logged with the `LOG_ERR` and `LOG_NOTICE` priorities, and informational messages are logged with the `LOG_INFO` priority. The default entries in the `/etc/syslog.conf` file log all the `drd` error messages to the `/var/adm/messages` log.

Notes The `drd` service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/platform/sun4v/drd:default
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. The service's status can be queried using the `svcs(1)` command.

Name drvconfig – apply permission and ownership changes to devices

Synopsis

```
drvconfig [-bn] [-a alias_name] [-c class_name]
     [-i drivername] [-m major_num] [-r root_dir]
```

Description devfsadm(1M) is now the preferred command and should be used instead of drvconfig.

The default operation of drvconfig is to apply permission and ownership changes to devices. Normally, this command is run automatically after a new driver has been installed (with add_drv(1M)) and the system has been rebooted.

Options The following options are supported:

-a*alias_name*	Add the name *alias_name* to the list of aliases that this driver is known by. This option, if used, must be used with the -m *major_num*, the -b and the -i *drivername* options.
-b	Add a new major number to name binding into the kernel's internal name_to_major tables. This option is not normally used directly, but is used by other utilities such as add_drv(1M). Use of the -b option requires that -i and -m be used also. No /devices entries are created.
-c*class_name*	The driver being added to the system exports the class *class_name*. This option is not normally used directly, but is used by other utilities. It is only effective when used with the -b option.
-i*drivername*	Only configure the devices for the named driver. The following options are used by the implementation of add_drv(1M) and rem_drv(1M), and may not be supported in future versions of Solaris:
-m*major_num*	Specify the major number *major_num* for this driver to add to the kernel's name_to_major binding tables.
-n	Do not try to load and attach any drivers, or if the -i option is given, do not try to attach the driver named *drivername*.
-r*root_dir*	Perform operations under *root_dir*, rather than directly under root.

Exit Status

0	Successful completion.
non-zero	An error occurred.

Files

/devices	Device nodes directory
/etc/minor_perm	Minor mode permissions
/etc/name_to_major	Major number binding
/etc/driver_classes	Driver class binding file

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also sh(1), add_drv(1M), modinfo(1M), modload(1M), modunload(1M), rem_drv(1M), update_drv(1M), path_to_inst(4), attributes(5), devfs(7FS)

Name dsbitmap – size Availability Suite bitmap volumes

Synopsis

```
dsbitmap -h

dsbitmap -p data_volume [bitmap_volume]

dsbitmap -r data_volume [bitmap_volume]
```

Description The `dsbitmap` command calculates the size of the Availability Suite bitmap volume required for use with the specified data volume.

Options The following options are supported:

`-h`
Prints the usage message for the `dsbitmap` command

`-p` *data_volume* [*bitmap_volume*]
For the given *data_volume*, `dsbitmap` calculates and display the required size for the associated Availability Suite Point in Time bitmap volume. The bitmap volume sizes for all possible Availability Suite Point in Time set configurations are displayed. These configurations include Independent shadow, Full size dependent shadow, and Compact dependent shadow. If the optional *bitmap_volume* argument is supplied, `dsbitmap` determines if this volume is large enough to be used as the bitmap volume for *data_volume*.

`-r` *data_volume* [*bitmap_volume*]
For the given *data_volume*, `dsbitmap` calculates and displays the required size for the associated Availability Suite Remote Mirror bitmap volume. The bitmap volume sizes for all possible Availability Suite Remote Mirror set configurations are displayed. These configurations include Sync replication, Async replication with memory queue, disk queue and 32 bit refcount. If the optional *bitmap_volume* argument is supplied, `dsbitmap` determines if this volume is large enough to be used as the bitmap volume for *data_volume*.

Usage `dsbitmap` is typically used by the system administrator during the initial stages of configuring Sun StorageTek Availability Suite software in order to determine the required bitmap volume sizes, and then to check if the bitmap volumes that have been created are suitable.

Exit Status The following exit values are returned:

`0`	Successful completion. If the name of a bitmap volume was specified, that volume is sufficiently large for all potential uses.
`1`	An error occurred.
`2`	An invalid option was supplied on the command line.
`3`	The specified bitmap volume is not large enough to be used as an Availability Suite Remote Mirror bitmap for an asynchronous set with a disk queue, but is large enough to be used for all other Remote Mirror set configurations.
`4`	The specified bitmap volume is not large enough to be used as an Availability Suite Remote Mirror bitmap for any Remote Mirror set configuration.

5 The specified bitmap volume is not large enough to be used as an Availability Suite Remote Mirror bitmap for any Remote Mirror set configuration.

6 The specified bitmap volume is not large enough to be used as an Availability Suite Point in Time bitmap for a compact dependent shadow, but is large enough to be used for all other Point in Time set configurations.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	storage/avs/avs-cache-management
Interface Stability	Committed

See Also `iiadm(1M)`, `sndradm(1M)`, `attributes(5)`

Name dscfg – configuration tool for Sun StorageTek Availability Suite software

Synopsis

```
dscfg

dscfg -L

dscfg -h

dscfg [-C group]

dscfg [-C group] -i [-p parser_config_file]

dscfg [-C group] -a config_file

dscfg [-C group] [-l]

dscfg [-C group] [-l] -s config_location

dscfg -D dgname
```

Description The `dscfg` command controls the Availability Suite configuration by providing facilities to initialize, list, format, restore the configuration database.

Options If no options are specified, `dscfg` displays the current local configuration location. The `dscfg` command supports the following options:

`-L`
: Displays the status of the lock controlling access to the configuration database. If the configuration database is locked, the type of lock (read or write) is displayed along with the process id of the process that holds the lock.

`-h`
: Displays the usage message for the `dscfg` command.

`-i`
: Initializes the configuration database. As it deletes previous or current configuration information, this option prompts you to confirm the deletion before proceeding.

`-p` *parser_config_file*
: When used with the `-i` option, `dscfg` loads the parser configuration file into the persistent configuration, it reformats the configuration database.

`-a` *config_file*
: Restore the specified *config_file* into the configuration. This option does not do any error checking of the file. Use of this option invalidates the configuration file.

`-l`
: Lists the contents of current configuration database in a format that is suitable for the `-a` option. When used in combination with the `-s` option, it displays the contents stored in the location passed to the `-s` option, but does not set the configuration location.

The options below are for Sun Cluster-configured systems only. *group* can be either Sun Cluster device group or as '-' as local devices.

`-C` *group*	Display the location of cluster configuration database.
`-C` *group* `-i`	Initializes the configuration database entries that only associated with group specified. As it deletes previous or current configuration information, this options prompts you to confirm the deletion before proceeding.
`-C` *group* `-p` *parser_config_file*	When used with the `-i` option, `dscfg` loads the parser configuration file into the persistent configuration, it reformats the configuration database entries that only associated with group specified.
`-C` *group* `-a` *config_file*	Restore the specified *config_file* into the configuration database entries that only associated with group specified. This option does not do any error checking of the file. Use of this option invalidates the configuration file.
`-C` *group* `-l`	Lists the contents of current configuration database that is associated with resource group specified.
`-C` *group* `-s` *config_file_location*	Specifies where the Sun Cluster configuration database resides, the location has to be DID device.
`-D` *device_group*	Checks whether the specified *device_group* is known by Sun Cluster and whether it is available on this node. It displays a information that indicates the device group's status and return values are as follows, (use echo $? to retrieve the return value): `1` Device group is in Sun Cluster and is active on this node; `0` Device group is in Sun Cluster but active on another node; `–1` Device group is not in Sun Cluster.

Usage The `dscfg` command is typically run from other scripts, such as package installation scripts, and from `dscfgadm(1M)`. It is not intended for general use. For uses of `dscfg` not covered by `dscfgadm`, please refer to the *Availability Suite Troubleshooting Guide*.

Files

`/etc/dscfg_format`
parser configuration file

`/etc/dscfg_local`
configuration location for all entries with no cluster tags

`/etc/dscfg_cluster`
reference file specifying the location of the Sun Cluster configuration database.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	storage/avs/avs-cache-management
Interface Stability	Committed

See Also dscfgadm(1M), iiadm(1M), scmadm(1M), sndradm(1M), svadm(1M), ds.log(4), attributes(5)

Name dscfgadm – administration command for the Availability Suite configuration location and services

Synopsis

```
dscfgadm -i

dscfgadm -e [-r] [-p]

dscfgadm -d [-r]

dscfgadm -s
```

Description The `dscfgadm` command controls the Availability Suite configuration location and services by providing facilities to set the configuration location, and to enable and disable the Availability Suite services.

Options If you do not specify any arguments to a `dscfgadm` command, the utility interactively steps you through setting the configuration location and starting the services. The configuration location is validated to ensure it meets criteria such as size and file type before it is initialized for use.

The `dscfgadm` command supports the following options.

`-d` [`-r`]
: Disables the Availability Suite SMF services. This includes stopping daemons and suspending any Point-in-Time Copy or Remote Mirror sets that are currently enabled under the Availability Suite software. When executed with no additional options, the `-d` option disables all Availability Suite services. This setting is persistent across system boots.

  `-r`
  : When passed to the `-d` or `-e` option, acts only on the Remote Mirror services.

`-e` [`-r`] [`-p`]
: Enables the Availability Suite SMF services. This includes starting daemons and resuming any Point-in-Time Copy or Remote Mirror sets that have been previously configured under the Availability Suite software. When executed with no additional options, the `-e` option enables all Availability Suite services. This setting is persistent across system boots.

  `-r`
  : When passed to the `-d` or `-e` option, acts only on the Remote Mirror services.

  `-p`
  : When passed to the `-e` option, enables only the Point-in-Time Copy service.

`-i`
: Displays information on the Availability Suite SMF services.

`-s`
: Allows you to set the location of the configuration database containing information specific to Sun Cluster configurations.

`-x`
: Displays verbose debugging information as the program is executing.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	storage/avs/avs-cache-management
Interface Stability	Committed

See Also dscfg(1M), iiadm(1M), scmadm(1M), sndradm(1M), svadm(1M), attributes(5)

Name dscfglockd – Availability Suite Services lock daemon

Synopsis

```
/usr/lib/dscfglockd [-e program | -f file]
```

Description The `dscfglockd` daemon coordinates StorageTek configuration database lock requests between nodes within a cluster.

Options The `dscfglockd` supports the following options:

`-e` *program*
Executes the script and arguments in *program*. The executable and any arguments it needs must be supplied with suitable quoting as a single argument to this option. This argument is processed by `sh(1)`.

`-f` *file*
Reads the list of names for peer hosts from *file*.

If no arguments are specified, `dscfglockd` acts as a local lock daemon, but coordinates lock requests with any other daemons that might contact the local host as part of their own configuration process.

Exit Status

`0` Daemon started successfully.

`>0` Daemon failed to start.

Files

`/lib/svc/method/svc-scm`
Shell script for starting and stopping `dscfglockd`.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	storage/avs/avs-cache-management
Interface Stability	Committed

See Also `sh(1)`, `attributes(5)`

Name dsstat – report Sun StorageTek Availability Suite I/O statistics

Synopsis

```
dsstat -m mode [-r report–options] [-d display-options]
 [-s volume-sets] [-f | -F] [-z] [interval [count]]
```

Description The `dsstat` command collects and reports I/O statistics for the Sun StorageTek Availability Suite products.

Options The `dsstat` supports the following options:

`-m` *mode*
Specifies the mode(s) of operation. Valid modes are:

`cache`
Cache statistics

`ii`
Point-in-Time Copy statistics

`sndr`
Remote Mirror statistics

To display statistics for multiple services simultaneously, use the `-m` switch to specify each of the modes required. For example:

```
% dsstat -m ii -m sndr
% dsstat -m ii -m sndr -m cache
```

To determine which statistics are being reported from which service, use the *role* field, described below. When cache statistics are requested as one of the multiple services, the `rkps` and `wkps` statistics is further divided into `crkps`, `drkps`, `cwkps`, and `dwkps`. If no `-m` switch is specified, then `dsstat` displays default statistics for all of the valid modes as described above. See "Field Descriptions," below for descriptions of these fields.

`-r` *report–options*
Specifies the volume components to be displayed. Each item is represented by a single character, and multiple items can be selected. The *report–options* vary based on the mode(s) specified above. This option is not used for `cache` mode.

Valid *report-options* for `ii` mode are:

`m` Master volume statistics.
- For the "report-options for ii mode": `m`, `s`, `b`, `o`
- For the "report-options for sndr mode": `b`, `n`
- For the "display-options for cache mode": `r`, `w`, `d`, `c`, `s`, `f`
- For the "display-options for ii mode": `r`, `w`, `t`, `s`, `p`, `f`

`s` Shadow volume statistics.

`b` Bitmap volume statistics.

`o` Overflow volume statistics, if attached.

Valid *report–options* for `sndr` mode are:

`b` Bitmap volume statistics.

`n` Network volume statistics.

`-d` *display–options*
Specifies the statistics to be displayed. The types of statistics are represented by a single character; multiple types can be specified.

For `cache` mode, the valid *display-options* are:

`r` Detailed read statistics.

`w` Detailed write statistics.

`s` Summary statistics.

`f` Cache behavior flags.

The following *display–options* are available only for cache mode, they need to be combined with the mode options (`-m`)

`d` Destaged data statistics.

`c` Write cancellation statistics.

The default *display-options* for `cache` are `sf`.

For `ii` mode, the valid *display-options* are:

`r`
Detailed read statistics.

`w`
Detailed write statistics.

`t`
Timing statistics.

`s`
Summary statistics.

`p`
Percentage of volume requiring sync.

`f`
Volume type/status flags.

The default *display-options* for `ii` are `spf`. For `sndr` mode, the valid *display-options* are:

`r` Detailed read statistics.

`w` Detailed write statistics.

`t` Timing statistics.

`s` Summary statistics.

`p` Percentage of volume requiring sync.

`f` Volume type/status flags.

`q` Asynchronous queue statistics.

The following *display-option* is only available for `sndr` mode, and needs to be combined with the mode options (`-m`).

`q` Asynchronous queue statistics.

The default *display-options* for `sndr` are `spf`.

Specifying the summary *display–option* causes `r w t` *display–options* to be ignored.

`-s` *volume–sets* Filters the output to include only the specified *volume–sets*, where *volume-sets* is a comma-delimited list of volume names. When displaying Remote Mirror statistics, the name specified is compared to the Remote Mirror primary volume to determine if they match. Additionally, an Remote Mirror volume should be specified as <*host*>:<*volume*> where <*host*> is the primary or secondary host and <*volume*> is the volume name by which that host recognizes the volume set. When specifying multiple Remote Mirror volumes sets, be aware that each volume set specified is evaluated individually, using the rules above. When displaying Point-in-Time Copy statistics, the volume name specified is compared to the Point-in-Time Copy shadow volume to determine if they match.

`-f` Output field headers at every reporting cycle.

`-F` Output field headers once, when reporting begins.

`-z` Suppress report lines that have zero values or no activity.

Operands The `dsstat` command line supports the following operands:

interval

Specifies the *interval* for each report, in seconds. If no *interval* is specified, a single report with a one second *interval* is output.

Due to the varying number of entries in a given set and the varying number of sets, header information might appear in the middle of a set being reported. To avoid this, use the `-f` or `-F` options to display the header information at the desired time.

count

Specifies the number of reports to generate. If no *count* is specified, output continues until interrupted.

Field Descriptions

Unless otherwise specified, all fields are per-second averages based on the data collected during the specified *interval*.

`name`	Name of the entity being reported
`t`	Volume type
`s`	Volume status
`r`	Cache read behavior
`w`	Cache write behavior
`pct`	Percentage of volume requiring sync
`role`	Role of the item being reported
`tps`	Total number of reads + writes
`rtps`	Number of reads
`wtps`	Number of writes
`kps`	Total kilobytes read + written
`rkps`	Kilobytes read
`wkps`	Kilobytes written
`crkps`	Kilobytes read from cache
`drkps`	Kilobytes read from disk
`cwkps`	Kilobytes written to cache
`dwkps`	Kilobytes written to disk
`ckps`	Kilobytes transferred to or from cache
`dkps`	Kilobytes transferred to or from disk
`svt`	Service time per operation
`hit`	Read hits during *interval*
`ds/s`	Kilobytes destaged from cache
`cn/s`	Number of write cancellations
`q`	Type of asynchronous queuing being used
`qi`	Number of items currently queued

`qk`	Kilobytes currently queued
`qhwi`	High water mark of queued items
`qhwk`	High water mark of queued kilobytes

The `name` field displays only the last 16 characters of the name.

Valid cache behaviors for `cache` are:

`C`	Cache reads/writes
`D`	Disk reads/writes

Valid volume types for `ii` are:

`I`	Independent shadow volume
`D`	Dependent shadow volume

Valid volume status for `ii` is:

`C`	Copy in progress
`-`	No copy in progress

Valid volume types for `sndr` are:

`P`	This is the primary host of this volume set
`S`	This is the secondary host of this volume set

Valid volume status for `sndr` is:

`L`	Changes to this volume are being logged
`Q`	Volume are in queuing mode
`R`	Replicating changes to secondary
`SY`	Synchronization in progress. Sending data.
`RS`	Reverse synchronization in progress. Receiving data.
`SN`	Synchronization needed
`RN`	Reverse synchronization needed
`VF`	Volume failed
`BF`	Bitmap failed
`QF`	Queue failed

Valid queue types for `sndr` are:

`D` Disk-based queuing enabled

`M` Memory-based queuing enabled

Volume roles for `sndr` are:

`net` Network volume statistics

`bmp` Bitmap volume statistics

Volume roles for `ii` are:

`mst` Master volume statistics

`shd` Shadow volume statistics

`bmp` Bitmap volume statistics

`ovr` Overflow volume statistics

Examples

EXAMPLE 1 Report Cache Statistics

The following example shows the display of Report Cache statistics, with detailed breakdowns of read and writes to cache/disk. Reports are generated at five second intervals. Reporting is limited to the set `/dev/rdsk/c1t1d0s0`.

```
 # dsstat -m cache -d rw -s /dev/rdsk/c1t1d0s0 5
- read -             - write -
name                ckps   dkps    hit   ckps   dkps    hit
dev/rdsk/c1t1d0s0      0      0   0.00      0      0   0.00
dev/rdsk/c1t1d0s0      3   2396   0.13    983    763 100.00
dev/rdsk/c1t1d0s0   2399    799  75.00   2815   2686 100.00
dev/rdsk/c1t1d0s0   3200    800  80.00   2755   2908 100.00
dev/rdsk/c1t1d0s0   3999    799  83.33   2809   2868 100.00
dev/rdsk/c1t1d0s0   4800    800  85.71   2867   2931 100.00
```

EXAMPLE 2 Report Master, Shadow and Bitmap Statistics

Report master, shadow and bitmap statistics for Point-in-Time Copy, using default output. Generate reports at two second intervals.

```
# dsstat -m ii -r msb 2
name               t  s    pct role    kps   tps  svt
dev/rdsk/c0t1d0s5  I  C  96.15  mst  19921    38   22
dev/rdsk/c0t1d0s6               shd   9960    19   20
dev/rdsk/c0t1d0s7               bmp     39    77    2
dev/rdsk/c0t1d0s5  I  C  94.24  mst  19623    38   22
dev/rdsk/c0t1d0s6               shd   9939    19   20
dev/rdsk/c0t1d0s7               bmp     39    77    2
dev/rdsk/c0t1d0s5  I  C  92.34  mst  19969    39   22
dev/rdsk/c0t1d0s6               shd   9984    19   20
```

EXAMPLE 2 Report Master, Shadow and Bitmap Statistics *(Continued)*

```
dev/rdsk/c0t1d0s7               bmp      39    78    2
```

EXAMPLE 3 Report Network Statistics for Remote Mirror

Report network statistics for Remote Mirror, using detailed read, write statistics. Report includes volume type/status flags and percentages. Generate reports at two second intervals. Limit reporting to the set `/dev/rdsk/c0t1d0s0`.

```
# dsstat -m sndr -r n -d rwpf -s /dev/rdsk/c0t1d0s0 2
name              t  s    pct role   rkps  rtps   wkps  wtps
dev/rdsk/c0t1d0s0  P  L 100.00  sec      0     0      0     0
dev/rdsk/c0t1d0s0  P SY  99.90  sec      0     0    288     9
dev/rdsk/c0t1d0s0  P SY  97.90  sec      0     0   5296   165
dev/rdsk/c0t1d0s0  P SY  95.81  sec      0     0   5184   161
dev/rdsk/c0t1d0s0  P SY  93.81  sec      0     0   5280   164
dev/rdsk/c0t1d0s0  P SY  91.71  sec      0     0   5198   162
```

Exit Status

The following exit values are returned:

`0` Successful completion.

1 Successful completion, no statistics to report.

2 An invalid argument has been encountered.

3 No memory is available to create `kstat` statistics.

4 An unknown error has occurred.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	storage/avs/avs-cache-management
Interface Stability	Committed

See Also

`dscfg(1M)`, `svadm(1M)`, `ds.log(4)`, `rdc.cf(4)`, `attributes(5)`

Name dsvclockd – DHCP service lock daemon

Synopsis

```
/usr/lib/inet/dsvclockd [-d 1 | 2] [-f ] [-v]
```

Description The `dsvclockd` daemon is a lock manager that works in conjunction with the Dynamic Host Configuration Protocol (DHCP) Data Service Library (`libdhcpsvc`). It provides shared or exclusive access to the `dhcp_network(4)` and `dhcptab(4)` tables. This service is used by the `SUNWbinfiles` and `SUNWfiles` DHCP data store modules. See `dhcp_modules(5)`.

`dsvclockd` is started on demand by `libdhcpsvc`. The `dsvclockd` daemon should be started manually only if command line options need to be specified.

Options The following options are supported:

`-d 1 | 2` Set debug level. Two levels of debugging are currently available, `1` and `2`. Level `2` is more verbose.

`-f` Run in the foreground instead of as a daemon process. When this option is used, messages are sent to standard error instead of to `syslog(3C)`.

`-v` Provide verbose output.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	service/network/dhcp
Interface Stability	Uncommitted

See Also `syslog(3C)`, `dhcp_network(4)`, `dhcptab(4)`, `dhcp_modules(5)`, `attributes(5)`

Name dtrace – DTrace dynamic tracing compiler and tracing utility

Synopsis

```
dtrace [-32 | -64] [-aACeFGHhlqSvVwZ] [-b bufsz] [-c cmd]
     [-D name [=value]] [-I path] [-L path] [-o output]
     [-s script] [-U name] [-x arg [=val]]
     [-X a | c | s | t] [-p pid]
     [-P provider [[predicate] action]]
     [-m [provider:] module [[predicate] action]]
     [-f [[provider:] module:] function [[predicate] action]]
     [-n [[[provider:] module:] function:] name [[predicate] action]]
     [-i probe-id [[predicate] action]]
```

Description DTrace is a comprehensive dynamic tracing framework for the Solaris Operating System. DTrace provides a powerful infrastructure that permits administrators, developers, and service personnel to concisely answer arbitrary questions about the behavior of the operating system and user programs.

The *Solaris Dynamic Tracing Guide* describes how to use DTrace to observe, debug, and tune system behavior. Refer to this book for a detailed description of DTrace features, including the bundled DTrace observability tools, instrumentation providers, and the D programming language.

The `dtrace` command provides a generic interface to the essential services provided by the DTrace facility, including:

- Options that list the set of probes and providers currently published by DTrace
- Options that enable probes directly using any of the probe description specifiers (provider, module, function, name)
- Options that run the D compiler and compile one or more D program files or programs written directly on the command line
- Options that generate anonymous tracing programs
- Options that generate program stability reports
- Options that modify DTrace tracing and buffering behavior and enable additional D compiler features

You can use `dtrace` to create D scripts by using it in a `#!` declaration to create an interpreter file. You can also use `dtrace` to attempt to compile D programs and determine their properties without actually enabling tracing using the `-e` option. See `OPTIONS`. See the *Solaris Dynamic Tracing Guide* for detailed examples of how to use the `dtrace` utility to perform these tasks.

Options The arguments accepted by the `-P`, `-m`, `-f`, `-n`, and `-i` options can include an optional D language *predicate* enclosed in slashes `//` and optional D language *action* statement list enclosed in braces `{}`. D program code specified on the command line must be appropriately quoted to avoid interpretation of meta-characters by the shell.

The following options are supported:

`-32 | -64`
The D compiler produces programs using the native data model of the operating system kernel. You can use the `isainfo -b` command to determine the current operating system data model. If the `-32` option is specified, `dtrace` forces the D compiler to compile a D program using the 32-bit data model. If the `-64` option is specified, `dtrace` forces the D compiler to compile a D program using the 64-bit data model. These options are typically not required as `dtrace` selects the native data model as the default. The data model affects the sizes of integer types and other language properties. D programs compiled for either data model can be executed on both 32-bit and 64-bit kernels. The `-32` and `-64` options also determine the ELF file format (ELF32 or ELF64) produced by the `-G` option.

`-a`
Claim anonymous tracing state and display the traced data. You can combine the `-a` option with the `-e` option to force `dtrace` to exit immediately after consuming the anonymous tracing state rather than continuing to wait for new data. See the *Solaris Dynamic Tracing Guide* for more information about anonymous tracing.

`-A`
Generate `driver.conf(4)` directives for anonymous tracing. This option constructs a set of `dtrace(7D)` configuration file directives to enable the specified probes for anonymous tracing and then exits. By default, `dtrace` attempts to store the directives to the file `/kernel/drv/dtrace.conf`. You can modify this behavior if you use the `-o` option to specify an alternate output file.

`-b` *bufsz*
Set principal trace buffer size (*bufsz*). The trace buffer size can include any of the size suffixes `k`, `m`, `g`, or `t`. If the buffer space cannot be allocated, `dtrace` attempts to reduce the buffer size or exit depending on the setting of the `bufresize` property.

`-c` *cmd*
Run the specified command *cmd* and exit upon its completion. If more than one `-c` option is present on the command line, `dtrace` exits when all commands have exited, reporting the exit status for each child process as it terminates. The process-ID of the first command is made available to any D programs specified on the command line or using the `-s` option through the `$target` macro variable. Refer to the *Solaris Dynamic Tracing Guide* for more information on macro variables.

`-C`
Run the C preprocessor `cpp(1)` over D programs before compiling them. You can pass options to the C preprocessor using the `-D`, `-U`, `-I`, and `-H` options. You can select the degree of C standard conformance if you use the `-X` option. For a description of the set of tokens defined by the D compiler when invoking the C preprocessor, see `-X`.

`-D` *name* `[=`*value*`]`
Define *name* when invoking `cpp(1)` (enabled using the `-C` option). If you specify the equals sign (`=`) and additional *value*, the name is assigned the corresponding value. This option passes the `-D` option to each `cpp` invocation.

`-e`
Exit after compiling any requests and consuming anonymous tracing state (`-a` option) but prior to enabling any probes. You can combine this option with the `-a` option to print anonymous tracing data and exit. You can also combine this option with D compiler options. This combination verifies that the programs compile without actually executing them and enabling the corresponding instrumentation.

`-f`[[*provider*`:`]*module*`:`]*function*[[*predicate*]*action*]]
Specify function name to trace or list (`-l` option). The corresponding argument can include any of the probe description forms *provider:module:function*, *module:function*, or *function*. Unspecified probe description fields are left blank and match any probes regardless of the values in those fields. If no qualifiers other than *function* are specified in the description, all probes with the corresponding *function* are matched. The `-f` argument can be suffixed with an optional D probe clause. You can specify more than one `-f` option on the command line at a time.

`-F`
Coalesce trace output by identifying function entry and return. Function entry probe reports are indented and their output is prefixed with `->`. Function return probe reports are unindented and their output is prefixed with `<-`. System call entry probe reports are indented and their output is prefixed with `=>`. System call return probe reports are unindented and their output is prefixed with `<=`.

`-G`
Generate an ELF file containing an embedded DTrace program. The DTrace probes specified in the program are saved inside of a relocatable ELF object which can be linked into another program. If the `-o` option is present, the ELF file is saved using the pathname specified as the argument for this operand. If the `-o` option is not present and the DTrace program is contained with a file whose name is *filename*`.d`, then the ELF file is saved using the name *filename*`.o`. Otherwise the ELF file is saved using the name `d.out`.

`-H`
Print the pathnames of included files when invoking cpp(1) (enabled using the `-C` option). This option passes the `-H` option to each `cpp` invocation, causing it to display the list of pathnames, one for each line, to `stderr`.

`-h`
Generate a header file containing macros that correspond to probes in the specified provider definitions. This option should be used to generate a header file that is included by other source files for later use with the `-G` option. If the `-o` option is present, the header file is saved using the pathname specified as the argument for that option. If the `-o` option is not present and the DTrace program is contained with a file whose name is *filename*`.d`, then the header file is saved using the name *filename*`.h`.

`-i` *probe-id*[[*predicate*] *action*]
Specify probe identifier (*probe-id*) to trace or list (`-l` option). You can specify probe IDs using decimal integers as shown by `dtrace -l`. The `-i` argument can be suffixed with an optional D probe clause. You can specify more than one `-i` option at a time.

`-I` *path*
Add the specified directory *path* to the search path for `#include` files when invoking cpp(1) (enabled using the `-C` option). This option passes the `-I` option to each `cpp` invocation. The specified *path* is inserted into the search path ahead of the default directory list.

`-L` *path*
Add the specified directory *path* to the search path for DTrace libraries. DTrace libraries are used to contain common definitions that can be used when writing D programs. The specified *path* is added after the default library search path.

`-l`
List probes instead of enabling them. If the `-l` option is specified, `dtrace` produces a report of the probes matching the descriptions given using the `-P`, `-m`, `-f`, `-n`, `-i`, and `-s` options. If none of these options are specified, this option lists all probes.

`-m` [[*provider:*] *module:* [[*predicate*] *action*]]
Specify module name to trace or list (`-l` option). The corresponding argument can include any of the probe description forms *provider:module* or *module*. Unspecified probe description fields are left blank and match any probes regardless of the values in those fields. If no qualifiers other than *module* are specified in the description, all probes with a corresponding *module* are matched. The `-m` argument can be suffixed with an optional D probe clause. More than one `-m` option can be specified on the command line at a time.

`-n` [[[*provider:*] *module:*] *function:*] *name* [[*predicate*] *action*]
Specify probe name to trace or list (`-l` option). The corresponding argument can include any of the probe description forms *provider:module:function:name*, *module:function:name*, *function:name*, or *name*. Unspecified probe description fields are left blank and match any probes regardless of the values in those fields. If no qualifiers other than *name* are specified in the description, all probes with a corresponding *name* are matched. The `-n` argument can be suffixed with an optional D probe clause. More than one `-n` option can be specified on the command line at a time.

`-o` *output*
Specify the *output* file for the `-A`, `-G`, `-h`, and `-l` options, or for the traced data itself. If the `-A` option is present and `-o` is not present, the default output file is `/kernel/drv/dtrace.conf`. If the `-G` option is present and the `-s` option's argument is of the form *filename*`.d` and `-o` is not present, the default output file is *filename*`.o`. Otherwise the default output file is `d.out`.

Note that with successive invocations of `dtrace` with the `-o` option, `dtrace` does not overwrite, but rather appends to the output file.

`-p` *pid*
Grab the specified process-ID *pid*, cache its symbol tables, and exit upon its completion. If more than one `-p` option is present on the command line, `dtrace` exits when all commands have exited, reporting the exit status for each process as it terminates. The first process-ID

is made available to any D programs specified on the command line or using the `-s` option through the `$target` macro variable. Refer to the *Solaris Dynamic Tracing Guide* for more information on macro variables.

`-P` *provider* [[*predicate*] *action*]
Specify provider name to trace or list (`-l` option). The remaining probe description fields module, function, and name are left blank and match any probes regardless of the values in those fields. The `-P` argument can be suffixed with an optional D probe clause. You can specify more than one `-P` option on the command line at a time.

`-q`
Set quiet mode. `dtrace` suppresses messages such as the number of probes matched by the specified options and D programs and does not print column headers, the CPU ID, the probe ID, or insert newlines into the output. Only data traced and formatted by D program statements such as `trace()` and `printf()` is displayed to `stdout`.

`-s`
Compile the specified D program source file. If the `-e` option is present, the program is compiled but instrumentation is not enabled. If the `-l` option is present, the program is compiled and the set of probes matched by it is listed, but instrumentation is not enabled. If none of `-e`, `-l`, `-G`, or `-A` are present, the instrumentation specified by the D program is enabled and tracing begins.

`-S`
Show D compiler intermediate code. The D compiler produces a report of the intermediate code generated for each D program to `stderr`.

`-U` *name*
Undefine the specified *name* when invoking `cpp(1)` (enabled using the `-C` option). This option passes the `-U` option to each `cpp` invocation.

`-v`
Set verbose mode. If the `-v` option is specified, `dtrace` produces a program stability report showing the minimum interface stability and dependency level for the specified D programs. DTrace stability levels are explained in further detail in the *Solaris Dynamic Tracing Guide*.

`-V`
Report the highest D programming interface version supported by `dtrace`. The version information is printed to `stdout` and the `dtrace` command exits. Refer to the *Solaris Dynamic Tracing Guide* for more information about DTrace versioning features.

`-w`
Permit destructive actions in D programs specified using the `-s`, `-P`, `-m`, `-f`, `-n`, or `-i` options. If the `-w` option is not specified, `dtrace` does not permit the compilation or enabling of a D program that contains destructive actions.

`-x` *arg* [=*val*]
Enable or modify a DTrace runtime option or D compiler option. The list of options is found in the *Solaris Dynamic Tracing Guide*. Boolean options are enabled by specifying their name. Options with values are set by separating the option name and value with an equals sign (=).

`-X a | c | s | t`
Specify the degree of conformance to the ISO C standard that should be selected when invoking cpp(1) (enabled using the `-C` option). The `-X` option argument affects the value and presence of the `__STDC__` macro depending upon the value of the argument letter.

The `-X` option supports the following arguments:

`a` Default. ISO C plus K&R compatibility extensions, with semantic changes required by ISO C. This is the default mode if `-X` is not specified. The predefined macro `__STDC__` has a value of 0 when `cpp` is invoked in conjunction with the `-Xa` option.

`c` Conformance. Strictly conformant ISO C, without K&R C compatibility extensions. The predefined macro `__STDC__` has a value of 1 when `cpp` is invoked in conjunction with the `-Xc` option.

`s` K&R C only. The macro `__STDC__` is not defined when `cpp` is invoked in conjunction with the `-Xs` option.

`t` Transition. ISO C plus K&R C compatibility extensions, without semantic changes required by ISO C. The predefined macro `__STDC__` has a value of 0 when `cpp` is invoked in conjunction with the `-Xt` option.

As the `-X` option only affects how the D compiler invokes the C preprocessor, the `-Xa` and `-Xt` options are equivalent from the perspective of D and both are provided only to ease re-use of settings from a C build environment.

Regardless of the `-X` mode, the following additional C preprocessor definitions are always specified and valid in all modes:

- `__sun`
- `__unix`
- `__SVR4`
- `__sparc` (on SPARC systems only)
- `__sparcv9` (on SPARC systems only when 64-bit programs are compiled)
- `__i386` (on x86 systems only when 32-bit programs are compiled)
- `__amd64` (on x86 systems only when 64-bit programs are compiled)
- `__`*‘uname -s‘*`_`*‘uname -r‘* (for example, `__SunOS_5_10`)
- `__SUNW_D=1`
- `__SUNW_D_VERSION=0x`*MMmmmuuu*

Where *MM* is the major release value in hexadecimal, *mmm* is the minor release value in hexadecimal, and *uuu* is the micro release value in hexadecimal. Refer to the *Solaris Dynamic Tracing Guide* for more information about DTrace versioning.

`-Z`

Permit probe descriptions that match zero probes. If the `-Z` option is not specified, `dtrace` reports an error and exits if any probe descriptions specified in D program files (`-s` option) or on the command line (`-P`, `-m`, `-f`, `-n`, or `-i` options) contain descriptions that do not match any known probes.

Operands You can specify zero or more additional arguments on the `dtrace` command line to define a set of macro variables (`$1`, `$2`, and so forth). The additional arguments can be used in D programs specified using the `-s` option or on the command line. The use of macro variables is described further in the *Solaris Dynamic Tracing Guide*.

Exit Status The following exit values are returned:

0 Successful completion.

For D program requests, an exit status of `0` indicates that programs were successfully compiled, probes were successfully enabled, or anonymous state was successfully retrieved. `dtrace` returns `0` even if the specified tracing requests encountered errors or drops.

1 An error occurred.

For D program requests, an exit status of `1` indicates that program compilation failed or that the specified request could not be satisfied.

2 Invalid command line options or arguments were specified.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/dtrace
Interface Stability	See below.

The command-line syntax is Committed. The human-readable output is Uncommitted.

See Also `cpp(1)`, `isainfo(1)`, `ssh(1)`, `libdtrace(3LIB)`, `driver.conf(4)`, `attributes(5)`, `dtrace(7D)`

Solaris Dynamic Tracing Guide

Usage When using the `-p` flag, `dtrace` stops the target processes while it is inspecting them and reporting results. A process can do nothing while it is stopped. This means that, if , for example, the X server is inspected by `dtrace` running in a window under the X server's

control, the whole window system can become deadlocked, because the `proc` tool would be attempting to display its results to a window that cannot be refreshed. In such a case, logging in from another system using `ssh(1)` and killing the offending `proc` tool clears the deadlock.

Name dumpadm – configure operating system crash dump

Synopsis

```
/usr/sbin/dumpadm [-nuy] [-c content-type] [-d dump-device]
     [-m mink | minm | min%] [-s savecore-dir]
     [-r root-dir] [-z on | off]
```

Description The `dumpadm` program is an administrative command that manages the configuration of the operating system crash dump facility. A crash dump is a disk copy of the physical memory of the computer at the time of a fatal system error. When a fatal operating system error occurs, a message describing the error is printed to the console. The operating system then generates a crash dump by writing the contents of physical memory to a predetermined dump device, which is typically a local disk partition. The dump device can be configured by way of `dumpadm`. Once the crash dump has been written to the dump device, the system will reboot.

Fatal operating system errors can be caused by bugs in the operating system, its associated device drivers and loadable modules, or by faulty hardware. Whatever the cause, the crash dump itself provides invaluable information to your support engineer to aid in diagnosing the problem. As such, it is vital that the crash dump be retrieved and given to your support provider. Following an operating system crash, the `savecore(1M)` utility is executed automatically during boot to retrieve the crash dump from the dump device and write it to your file system in compressed form, to a file name `vmdump.`*X*, where *X* is an integer identifying the dump. Afterwards, `savecore(1M)` can be invoked on the same or another system to expand the compressed crash dump to a pair of files named `unix.`*X* and `vmcore.`*X*. The directory in which the crash dump is saved on reboot can be configured using `dumpadm`.

For systems with a UFS root file system, the default dump device is configured to be an appropriate swap partition. Swap partitions are disk partitions reserved as virtual memory backing store for the operating system. Thus, no permanent information resides in swap to be overwritten by the dump. See `swap(1M)`. For systems with a ZFS root file system, dedicated ZFS volumes are used for swap and dump areas. For further information about setting up a dump area with ZFS, see the *ZFS Administration Guide*. To view the current dump configuration, use the `dumpadm` command with no arguments:

```
example# dumpadm

      Dump content: kernel pages
       Dump device: /dev/dsk/c0t0d0s1 (swap)
Savecore directory: /var/crash
  Savecore enabled: yes
   Save compressed: yes
```

When no options are specified, `dumpadm` displays the current crash dump configuration. The example above shows the set of default values: the dump content is set to kernel memory pages only, the dump device is a swap disk partition, the directory for `savecore` files is set to `/var/crash/`. `savecore` is set to run automatically on reboot and save the crash dump in a compressed format.

When one or more options are specified, `dumpadm` verifies that your changes are valid, and if so, reconfigures the crash dump parameters and displays the resulting configuration. You must be `root` to view or change dump parameters.

Upon system installation, `dumpadm` establishes a dump device of sufficient size, based on kernel memory and other internal information, to accommodate a dump file. If you subsequently attempt to create a dump device that is too small to store the dump file, `dumpadm` issues an error message and the operation fails.

Options The following options are supported:

`-c` *content-type*
Modify the dump configuration so that the crash dump consists of the specified dump content. The content should be one of the following:

`kernel`
Kernel memory pages only.

`all`
All memory pages.

`curproc`
Kernel memory pages, and the memory pages of the process whose thread was currently executing on the CPU on which the crash dump was initiated. If the thread executing on that CPU is a kernel thread not associated with any user process, only kernel pages will be dumped.

`-d` *dump-device*
Modify the dump configuration to use the specified dump device. The dump device may one of the following:

dump-device
A specific dump device specified as an absolute pathname, such as `/dev/dsk/`*cNtNdNsN* when the system is running a UFS root file system. Or, specify a ZFS volume, such as `/dev/zvol/dsk/rpool/dump`, when the system is running a ZFS root file system.

`swap`
If the special token `swap` is specified as the dump device, `dumpadm` examines the active swap entries and selects the most appropriate entry to configure as the dump device. See `swap(1M)`. Refer to the `NOTES` below for details of the algorithm used to select an appropriate swap entry. When the system is first installed with a UFS root file system, `dumpadm` uses the value for `swap` to determine the initial dump device setting. A given ZFS volume cannot be configured for both the swap area and the dump device.

`-m` *min*`k` | *min*`m` | *min*`%`
Create a `minfree` file in the current savecore directory indicating that `savecore` should maintain at least the specified amount of free space in the file system where the savecore directory is located. The `min` argument can be one of the following:

k
A positive integer suffixed with the unit k specifying kilobytes.

m
A positive integer suffixed with the unit m specifying megabytes.

%
A % symbol, indicating that the minfree value should be computed as the specified percentage of the total current size of the file system containing the savecore directory.

The savecore command will consult the minfree file, if present, prior to writing the dump files. If the size of these files would decrease the amount of free disk space below the minfree threshold, no dump files are written and an error message is logged. The administrator should immediately clean up the savecore directory to provide adequate free space, and re-execute the savecore command manually. The administrator can also specify an alternate directory on the savecore command-line.

-n
Modify the dump configuration to not run savecore automatically on reboot. This is not the recommended system configuration; if the dump device is a swap partition, the dump data will be overwritten as the system begins to swap. If savecore is not executed shortly after boot, crash dump retrieval may not be possible.

-r *root-dir*
Specify an alternate root directory relative to which dumpadm should create files. If no -r argument is specified, the default root directory / is used.

-s *savecore-dir*
Modify the dump configuration to use the specified directory to save files written by savecore. The directory should be an absolute path and exist on the system. If upon reboot the directory does not exist, it will be created prior to the execution of savecore. See the NOTES section below for a discussion of security issues relating to access to the savecore directory. The default savecore directory is /var/crash/.

-u
Forcibly update the kernel dump configuration based on the contents of /etc/dumpadm.conf. Normally this option is used only on reboot when starting svc:/system/dumpadm:default, when the dumpadm settings from the previous boot must be restored. Your dump configuration is saved in the configuration file for this purpose. If the configuration file is missing or contains invalid values for any dump properties, the default values are substituted. Following the update, the configuration file is resynchronized with the kernel dump configuration.

-y
Modify the dump configuration to automatically run savecore on reboot. This is the default for this dump setting. See NOTES.

`-z on | off`
Modify the dump configuration to control the operation of `savecore` on reboot. The options are `on`, to enable saving core files in a compressed format, and `off`, to automatically uncompress the crash dump file. The default is `on`, because crash dump files can be very large and require less file system space if saved in a compressed format.

Examples

EXAMPLE 1 Reconfiguring The Dump Device To A Dedicated Dump Device:

The following command reconfigures the dump device to a dedicated dump device:

```
example# dumpadm –d /dev/dsk/c0t2d0s2

          Dump content: kernel pages
           Dump device: /dev/dsk/c0t2d0s2 (dedicated)
    Savecore directory: /var/crash
      Savecore enabled: yes
       Save compressed: yes
```

Exit Status

The following exit values are returned:

`0`
Dump configuration is valid and the specified modifications, if any, were made successfully.

`1`
A fatal error occurred in either obtaining or modifying the dump configuration.

`2`
Invalid command line options were specified.

Files

`/dev/dump`
Dump device.

`/etc/dumpadm.conf`
Contains configuration parameters for `dumpadm`. Modifiable only through that command.

savecore-directory`/minfree`
Contains minimum amount of free space for *savecore-directory*. See `savecore(1M)`.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also

`svcs(1)`, `uname(1)`, `savecore(1M)`, `svcadm(1M)`, `swap(1M)`, `attributes(5)`, `smf(5)`

Notes The system crash dump service is managed by the service management facility, smf(5), under the service identifier:

```
svc:/system/dumpadm:default
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using svcadm(1M). The service's status can be queried using the svcs(1) command.

Dump Device Selection

When the special `swap` token is specified as the argument to `dumpadm -d` the utility will attempt to configure the most appropriate swap device as the dump device. `dumpadm` configures the largest swap block device as the dump device; if no block devices are available for swap, the largest swap entry is configured as the dump device. If no swap entries are present, or none can be configured as the dump device, a warning message will be displayed. While local and remote swap files can be configured as the dump device, this is not recommended.

Dump Device/Swap Device Interaction (UFS File Systems Only)

In the event that the dump device is also a swap device, and the swap device is deleted by the administrator using the `swap -d` command, the `swap` command will automatically invoke `dumpadm -d swap` in order to attempt to configure another appropriate swap device as the dump device. If no swap devices remain or none can be configured as the dump device, the crash dump will be disabled and a warning message will be displayed. Similarly, if the crash dump is disabled and the administrator adds a new swap device using the `swap -a` command, `dumpadm -d swap` will be invoked to re-enable the crash dump using the new swap device.

Once `dumpadm -d swap` has been issued, the new dump device is stored in the configuration file for subsequent reboots. If a larger or more appropriate swap device is added by the administrator, the dump device is not changed; the administrator must re-execute `dumpadm -d swap` to reselect the most appropriate device fom the new list of swap devices.

Minimum Free Space

If the `dumpadm -m` option is used to create a `minfree` file based on a percentage of the total size of the file system containing the `savecore` directory, this value is not automatically recomputed if the file system subsequently changes size. In this case, the administrator must re-execute `dumpadm -m` to recompute the `minfree` value. If no such file exists in the `savecore` directory, `savecore` will default to a free space threshold of one megabyte. If no free space threshold is desired, a minfree file containing size 0 can be created.

Security Issues

If, upon reboot, the specified `savecore` directory is not present, it will be created prior to the execution of `savecore` with permissions 0700 (read, write, execute by owner only) and owner `root`. It is recommended that alternate savecore directories also be created with similar permissions, as the operating system crash dump files themselves may contain secure information.

Default for `savecore`

System installation software might reserve a dedicated dump device (for example, a disk slice or a ZFS volume). In such a case, the `dumpadm` default can be set to `-n`, meaning that `savecore` does not run automatically when the system reboots. A crash image will be preserved on the dump device. Run `/usr/sbin/savecore` manually as root to retrieve the crash image and copy it to a file under `/var/crash`. The crash image will remain on the dump device until overwritten by a later one.

Name editmap – query and edit single records in database maps for sendmail

Synopsis

```
editmap -C file [-N] [-f] [-q | -u | -x] maptype mapname key
     ["value"]...
```

Description The `editmap` command queries or edits one record in a database maps used by the keyed map lookups in `sendmail(1M)`. Arguments are passed on the command line and output (for queries) is directed to standard output.

Depending on how it is compiled, `editmap` handles up to three different database formats, selected using the *maptype* parameter. See `OPERANDS`.

If the `TrustedUser` option is set in the `sendmail` configuration file and `editmap` is invoked as root, the generated files are owned by the specified `TrustedUser`.

Options The following options are supported:

`-C` *file* Use the specified `sendmail` configuration file (*file*) to look up the `TrustedUser` option.

`-f` Disable the folding of all upper case letters in the key to lower case. Normally, all upper case letters in the key are folded to upper case. This is intended to mesh with the `-f` flag in the `K` line in `sendmail.cf`. The value is never case folded.

`-N` Include the null byte that terminates strings in the map (for alias maps).

`-q` Query the map for the specified key. If found, print value to standard output and exit with `0`. If not found then print an error message to `stdout` and exit with `EX_UNAVAILABLE`.

`-u` Update the record for key with value or inserts a new record if one doesn't exist. Exits with `0` on success or `EX_IOERR` on failure.

`-x` Delete the specific key from the map. Exit with `0` on success or `EX_IOERR` on failure.

Operands The following operands are supported:

key The left hand side of a record.

Each record is of the form:

key value

key and *value* are separated by white space.

mapname File name of the database map being created.

maptype Specifies the database format. The following *maptype* parameters are available:

`dbm` Specifies DBM format maps.

`btree` Specifies B-Tree format maps.

`hash` Specifies hash format maps.

value The right hand side of a record.

Each record is of the form:

key value

key and *value* are separated by white space.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	service/network/smtp/sendmail

See Also `makemap(1M)`, `sendmail(1M)`, `attributes(5)`

Name edquota – edit user quotas for ufs file system

Synopsis

```
edquota [-p proto_user] username...

edquota -t
```

Description `edquota` is a quota editor. One or more users may be specified on the command line. For each user a temporary file is created with an ASCII representation of the current disk quotas for that user for each mounted ufs file system that has a `quotas` file, and an editor is then invoked on the file. The quotas may then be modified, new quotas added, etc. Upon leaving the editor, `edquota` reads the temporary file and modifies the binary quota files to reflect the changes made.

The editor invoked is `vi(1)` unless the `EDITOR` environment variable specifies otherwise.

Only the super-user may edit quotas. In order for quotas to be established on a file system, the root directory of the file system must contain a file, owned by root, called `quotas`. (See `quotaon(1M)`.)

proto_user and *username* can be numeric, corresponding to the UID of a user. Unassigned UIDs may be specified; unassigned names may not. In this way, default quotas can be established for users who are later assigned a UID.

If no options are specified, the temporary file created will have one or more lines of the format, where a block is considered to be a 1024 byte (1K) block:

```
fs mount_point blocks (soft =number, \
     hard =number ) inodes (soft =number, \
     hard =number)
```

The *number* fields may be modified to reflect desired values.

Options The following options are supported:

`-p` Duplicate the quotas of the *proto_user* specified for each *username* specified. This is the normal mechanism used to initialize quotas for groups of users.

`-t` Edit the soft time limits for each file system. If the time limits are zero, the default time limits in `/usr/include/sys/fs/ufs_quota.h` are used. The temporary file created will have one or more lines of the form

fs *mount_point* blocks time limit = *number tmunit*, files time limit = *number tmunit*

tmunit may be one of "month", "week", "day", "hour", "min" or "sec"; characters appended to these keywords are ignored, so you may write "months" or "minutes" if you prefer. The *number* and *tmunit* fields may be modified to set desired values. Time limits are printed in the greatest possible time unit such that the value is greater than or equal to one. If "default" is printed after the *tmunit*, this indicates that the value shown is zero (the default).

Usage See `largefile(5)` for the description of the behavior of `edquota` when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

Files

`quotas`	quota file at the file system root
`/etc/mnttab`	table of mounted file systems

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also `vi(1)`, `quota(1M)`, `quotacheck(1M)`, `quotaon(1M)`, `repquota(1M)`, `attributes(5)`, `largefile(5)`, `quotactl(7I)`

Notes All UIDs can be assigned quotas.

Name eeprom – EEPROM display and load utility

Synopsis

```
/usr/sbin/eeprom [-] [-f device] [parameter[=value]]
```

Description `eeprom` displays or changes the values of parameters in the EEPROM. It processes parameters in the order given. When processing a *parameter* accompanied by a *value*, `eeprom` makes the indicated alteration to the EEPROM; otherwise, it displays the *parameter*'s value. When given no parameter specifiers, `eeprom` displays the values of all EEPROM parameters. A '`-`' (hyphen) flag specifies that parameters and values are to be read from the standard input (one *parameter* or *parameter*=*value* per line).

Only the super-user may alter the EEPROM contents.

`eeprom` verifies the EEPROM checksums and complains if they are incorrect.

platform-name is the name of the platform implementation and can be found using the `-i` option of `uname(1)`.

SPARC SPARC based systems implement firmware password protection with `eeprom`, using the `security-mode`, `security-password` and `security-#badlogins` properties.

x86 EEPROM storage is simulated using a file residing in the platform-specific boot area. The `/boot/solaris/bootenv.rc` file simulates EEPROM storage.

Because x86 based systems typically implement password protection in the system BIOS, there is no support for password protection in the `eeprom` program. While it is possible to set the `security-mode`, `security-password` and `security-#badlogins` properties on x86 based systems, these properties have no special meaning or behavior on x86 based systems.

Options

`-f` *device*
Use *device* as the EEPROM device.

Operands

x86 Only *acpi-user-options*
A configuration variable that controls the use of Advanced Configuration and Power Interface (ACPI), a power management specification. The acceptable values for this variable depend on the release of the Solaris operating system you are using.

For all releases of Solaris 10 and Solaris 11, a value of of `0x0` means that there will be an attempt to use ACPI if it is available on the system. A value of `0x2` disables the use of ACPI.

For the Solaris 10 1/06 release, a value of `0x8` means that there will be an attempt to use ACPI in a mode compatible with previous releases of Solaris 10 if it is available on the system. The default for Solaris 10 1/06 is `0x8`.

For releases of Solaris 10 after the 1/06 release and for Solaris 11, the default is `0x0`.

Most users can safely accept the default value, which enables ACPI if available. If issues related to the use of ACPI are suspected on releases of Solaris after Solaris 1/06, it is suggested to first try a value of `0x8` and then, if you do not obtain satisfactory results, `0x02`.

console
Specifies the console device. Possible values are `ttya`, `ttyb`, `text`, `graphics` and `force-text`. In `text` mode, console output goes to the frame buffer and input comes from the keyboard. A variant of text mode, `graphics` displays an image with an animation until either a key is pressed or console interaction is required by console login, `sulogin`, or `kmdb`. A further variant of `text`, `force-text` will avoid using a VGA adapter as a bitmapped device setting it to VGA text mode. When this property is not present, the console device falls back to the device specified by `input-device` and `output-device`. When neither the `console` property nor the `input-device` and `output-device` property pair are present, the console defaults to the frame buffer and keyboard.

`screen-`*#columns* `screen-`*#rows*
When `screen` is set to either `graphics` or `text` on a bitmapped device, `screen-`*#columns* and `screen-`*#rows* allow the desired number columns and rows of text to be specified. They default to `80` and `24` respectively.

Nvram Configuration Parameters

Not all OpenBoot systems support all parameters. Defaults vary depending on the system and the PROM revision. See the output in the "Default Value" column of the `printenv` command, as entered at the `ok` (OpenBoot) prompt, to determine the default for your system.

auto-boot?
If `true`, boots automatically after power-on or reset. Defaults to `true`. On x86, this parameter is controlled by the grub menu file. See `installgrub(1M)`.

ansi-terminal?
Configuration variable used to control the behavior of the terminal emulator. The value `false` makes the terminal emulator stop interpreting ANSI escape sequences; instead, echoes them to the output device. Defaults to `true`.

boot-args
Holds a string of arguments that are passed to the boot subsystem. For example, you can use `boot-args=' - install dhcp'` to request a customer jumpstart installation. See `boot(1M)`, `kmdb(1)`, and `kernel(1M)`.

boot-command
Command executed if `auto-boot?` is `true`. Defaults to `boot`.

boot-device
Device from which to boot. *boot-device* may contain 0 or more device specifiers separated by spaces. Each device specifier may be either a prom device alias or a prom device path. The boot prom will attempt to open each successive device specifier in the list beginning with the first device specifier. The first device specifier that opens successfully will be used as the device to boot from. Defaults to `disk net`.

boot-device-index
Keeps track onf the device index into the `boot-device` variable.

boot-file
File to boot (an empty string lets the secondary booter choose default). Defaults to empty string.

boot-from
Boot device and file (OpenBoot PROM version 1.*x* only). Defaults to `vmunix`.

boot-from-diag
Diagnostic boot device and file (OpenBoot PROM version 1.*x* only). Defaults to `le( )unix`.

boot-ncpus
Configuration variable that controls the number of processors with which the system should boot. By default, the system boots with maximum supported number of processors.

com*X*-noprobe
Where *X* is the number of the serial port, prevents device probe on serial port *X*.

diag-device
Diagnostic boot source device. Defaults to `net`.

diag-file
File from which to boot in diagnostic mode. Defaults to empty string.

diag-level
Diagnostics level. Values include `off`, `min`, `max` and `menus`. There may be additional platform-specific values. When set to `off`, POST is not called. If POST is called, the value is made available as an argument to, and is interpreted by POST. Defaults to `platform-dependent`.

diag-switch?
If `true`, run in diagnostic mode. Defaults to `false` on most desktop systems, `true` on most servers.

error-reset-recovery
Recover after an error reset trap. Defaults to platform-specific setting.

On platforms supporting this variable, it replaces the `watchdog-reboot?`, `watchdog-sync?`, `redmode-reboot?`, `redmode-sync?`, `sir-sync?`, and `xir-sync?` parameters.

The options are:

none
Print a message describing the reset trap and go to OpenBoot PROM's user interface, *aka* `OK` prompt.

sync
Invoke OpenBoot PROM's `sync` word after the reset trap. Some platforms may treat this as `none` after an externally initiated reset (XIR) trap.

boot
Reboot after the reset trap. Some platforms may treat this as `none` after an XIR trap.

fcode-debug?
If `true`, include name parameter for plug-in device FCodes. Defaults to `false`.

hardware-revision
System version information.

input-device
Input device used at power-on (usually `keyboard`, `ttya`, or `ttyb`). Defaults to `keyboard`.

keyboard-click?
If `true`, enable keyboard click. Defaults to `false`.

keyboard-layout
A string that specifies the layout name for non-self-identifying keyboards (type 7c). Invoke `kbd -s` to obtain a list of acceptable layout names. See kbd(1).

keymap
Keymap for custom keyboard.

last-hardware-update
System update information.

load-base
Default load address for client programs. Default value is `16384`.

local-mac-address?
If true, network drivers use their own MAC address, not the system's. Defaults to `false`.

mfg-mode
Manufacturing mode argument for POST. Possible values include `off` or `chamber`. The value is passed as an argument to POST. Defaults to `off`.

mfg-switch?
If true, repeat system self-tests until interrupted with STOP-A. Defaults to `false`.

multipath-boot?
If true, is used by the PROM to cycle through the list of I/O devices provided in the `boot-device` variable, until a successful boot is performed with a device from the list.

nvramrc
Contents of NVRAMRC. Defaults to empty.

network-boot-arguments
Arguments to be used by the PROM for network booting. Defaults to an empty string. `network-boot-arguments` can be used to specify the boot protocol (RARP/DHCP) to be used and a range of system knowledge to be used in the process.

The syntax for arguments supported for network booting is:

`[`*protocol*`,] [`*key*`=`*value*`,]*`

All arguments are optional and can appear in any order. Commas are required unless the argument is at the end of the list. If specified, an argument takes precedence over any default values, or, if booting using DHCP, over configuration information provided by a DHCP server for those parameters.

protocol, above, specifies the address discovery protocol to be used.

Configuration parameters, listed below, are specified as *key*=*value* attribute pairs.

`tftp-server`
IP address of the TFTP server

`file`
file to download using TFTP or URL for WAN boot

`host-ip`
IP address of the client (in dotted-decimal notation)

`router-ip`
IP address of the default router (in dotted-decimal notation)

`subnet-mask`
subnet mask (in dotted-decimal notation)

`client-id`
DHCP client identifier

`hostname`
hostname to use in DHCP transactions

`http-proxy`
HTTP proxy server specification (IPADDR[:PORT])

`tftp-retries`
maximum number of TFTP retries

`dhcp-retries`
maximum number of DHCP retries

If no parameters are specified (that is, `network-boot-arguments` is an empty string), the PROM will use the platform-specific default address discovery protocol.

Absence of the protocol parameter when other configuration parameters are specified implies manual configuration.

Manual configuration requires that the client be provided with all the information necessary for boot. If using manual configuration, information required by the PROM to load the second-stage boot program must be provided in `network-boot-arguments` while

information required for the second-stage boot program can be specified either as arguments to the `boot` program or by means of the `boot` program's interactive command interpreter.

Information required by the PROM when using manual configuration includes the booting client's IP address, name of the boot file, and the address of the server providing the boot file image. Depending on network configuration, it might be required that the subnet mask and address of the default router to use also be specified.

oem-banner
Custom OEM banner (enabled by setting `oem-banner?` to `true`). Defaults to empty string.

oem-banner?
If `true`, use custom OEM banner. Defaults to `false`.

oem-logo
Byte array custom OEM logo (enabled by setting `oem-logo?` to `true`). Displayed in hexadecimal.

oem-logo?
If `true`, use custom OEM logo (else, use Sun logo). Defaults to `false`.

pci-mem64?
If true, the OpenBoot PROM allocates 64-bit PCI memory addresses to a PCI device that can support 64-bit addresses.

This variable is available on SPARC platforms only and is optional. Some versions of SunOS do not support PCI `MEM64` addresses and will fail in unexpected ways if the OpenBoot PROM allocates PCI `MEM64` addresses.

The default value is system-dependent. If the variable exists, the default value is appropriate to the lowest version of the SunOS that shipped with a specific platform.

output-device
Output device used at power-on (usually `screen`, `ttya`, or `ttyb`). Defaults to `screen`.

redmode-reboot?
Specify `true` to reboot after a redmode reset trap. Defaults to `true`. (Sun Enterprise 10000 only.)

redmode-sync?
Specify `true` to invoke OpenBoot PROM's `sync` word after a redmode reset trap. Defaults to `false`. (Sun Enterprise 10000 only.)

rootpath
Specifies the root device of the operating system.

sbus-probe-list
Designate which SBus slots are probed and in what order. Defaults to `0123`.

screen-#columns
Number of on-screen columns (characters/line). Defaults to `80`.

screen-#rows
Number of on-screen rows (lines). Defaults to `34`.

scsi-initiator-id
SCSI bus address of host adapter, range 0-7. Defaults to `7`.

sd-targets
Map SCSI disk units (OpenBoot PROM version 1.*x* only). Defaults to `31204567`, which means that unit 0 maps to target `3`, unit 1 maps to target `1`, and so on.

security-#badlogins
Number of incorrect security password attempts.This property has no special meaning or behavior on x86 based systems.

security-mode
Firmware security level (options: `none`, `command`, or `full`). If set to `command` or `full`, system will prompt for PROM security password. Defaults to `none`.This property has no special meaning or behavior on x86 based systems.

security-password
Firmware security password (never displayed). Can be set only when `security-mode` is set to `command` or `full`.This property has no special meaning or behavior on x86 based systems.

```
example# eeprom security-password=
Changing PROM password:
New password:
Retype new password:
```

selftest-#megs
Megabytes of RAM to test. Ignored if `diag-switch?` is `true`. Defaults to `1`.

sir-sync?
Specify `true` to invoke OpenBoot PROM's `sync` word after a software-initiated reset (SIR) trap. Defaults to `false`. (Sun Enterprise 10000 only.)

skip-vme-loopback?
If `true`, POST does not do VMEbus loopback tests. Defaults to `false`.

st-targets
Map SCSI tape units (OpenBoot PROM version 1.*x* only). Defaults to `45670123`, which means that unit 0 maps to target `4`, unit 1 maps to target `5`, and so on.

sunmon-compat?
If `true`, display Restricted Monitor prompt (>). Defaults to `false`.

testarea
One-byte scratch field, available for read/write test. Defaults to `0`.

tpe-link-test?
Enable 10baseT link test for built-in twisted pair Ethernet. Defaults to `true`.

ttya-mode
TTYA (baud rate, #bits, parity, #stop, handshake). Defaults to `9600,8,n,1,–`.

Fields, in left-to-right order, are:

Baud rate:
110, 300, 1200, 4800, 9600 . . .

Data bits:
5, 6, 7, 8

Parity:
n(none), e(even), o(odd), m(mark), s(space)

Stop bits:
1, 1.5, 2

Handshake:
–(none), h(hardware:rts/cts), s(software:xon/xoff)

ttyb-mode
TTYB (baud rate, #bits, parity, #stop, handshake). Defaults to `9600,8,n,1,–`.

Fields, in left-to-right order, are:

Baud rate:
110, 300, 1200, 4800, 9600 . . .

Data bits:
5, 6, 7, 8

Stop bits:
1, 1.5, 2

Parity:
n(none), e(even), o(odd), m(mark), s(space)

Handshake:
–(none), h(hardware:rts/cts), s(software:xon/xoff)

ttya-ignore-cd
If `true`, operating system ignores carrier-detect on TTYA. Defaults to `true`.

ttyb-ignore-cd
If `true`, operating system ignores carrier-detect on TTYB. Defaults to `true`.

ttya-rts-dtr-off
If `true`, operating system does not assert DTR and RTS on TTYA. Defaults to `false`.

ttyb-rts-dtr-off
If `true`, operating system does not assert DTR and RTS on TTYB. Defaults to `false`.

use-nvramrc?
If `true`, execute commands in NVRAMRC during system start-up. Defaults to `false`.

verbosity
Controls the level of verbosity of PROM messages. Can be one of `debug`, `max`, `normal`, `min`, or `none`. Defaults to `normal`.

version2?
If `true`, hybrid (1.*x*/2.*x*) PROM comes up in version 2.*x*. Defaults to `true`.

watchdog-reboot?
If `true`, reboot after watchdog reset. Defaults to `false`.

watchdog-sync?
Specify `true` to invoke OpenBoot PROM's `sync` word after a watchdog reset trap. Defaults to `false`. (Sun Enterprise 10000 only.)

xir-sync?
Specify `true` to invoke OpenBoot PROM's `sync` word after an XIR trap. Defaults to `false`. (Sun Enterprise 10000 only.)

Examples

EXAMPLE 1 Changing the Number of Megabytes of RAM.

The following example demonstrates the method for changing from one to two the number of megabytes of RAM that the system will test.

```
example# eeprom selftest-#megs
selftest-#megs=1

example# eeprom selftest-#megs=2

example# eeprom selftest-#megs
selftest-#megs=2
```

EXAMPLE 2 Setting the auto-boot? Parameter to `true`.

The following example demonstrates the method for setting the `auto-boot?` parameter to `true`.

```
example# eeprom auto-boot?=true
```

When the `eeprom` command is executed in user mode, the parameters with a trailing question mark (`?`) need to be enclosed in double quotation marks (“ “) to prevent the shell from interpreting the question mark. Preceding the question mark with an escape character (`\`) will also prevent the shell from interpreting the question mark.

```
example% eeprom "auto-boot?"=true
```

EXAMPLE 3 Using network-boot-arguments

To use DHCP as the boot protocol and a hostname of `abcd.example.com` for network booting, set these values in `network-boot-arguments` as:

```
example# eeprom network-boot-arguments="dhcp,hostname=abcd.example.com"
```

...then boot using the command:

```
ok boot net
```

Note that network boot arguments specified from the PROM command line cause the contents of `network-boot-arguments` to be ignored. For example, with `network-boot-arguments` set as shown above, the `boot` command:

```
ok boot net:dhcp
```

...causes DHCP to be used, but the `hostname` specified in `network-boot-arguments` will not be used during network boot.

EXAMPLE 4 Setting System Console to Auxiliary Device

The command below assigns the device `/dev/term/a` as the system console device. You would make such an assignment prior to using `tip(1)` to establish a `tip` connection to a host.

On a SPARC machine:

```
# eeprom output-device=/dev/term/a
```

On an x86 machine:

```
# eeprom console=ttya
```

On a SPARC machine, the preceding command would be sufficient for assigning the console to an auxiliary device. For an x86 machine, you might, in addition, need to set the characteristics of the serial line, for which you would have to consult the BIOS documentation for that machine. Also, on some x86 machines, you might use a device other than device `a`, as shown above. For example, you could set console to `ttyb` if the second serial port is present.

EXAMPLE 5 Specifying that SPARC System Boots into `kmdb`

To specify that a SPARC machine boots into `kmdb(1)`, enter the following command:

```
# eeprom boot-command="boot -k"
```

Files

`/boot/solaris/bootenv.rc`
File storing `eeprom` values on x86 machines.

`/dev/openprom`
Device file

`/usr/platform/`*platform-name*`/sbin/eeprom`
Platform-specific version of `eeprom`. Use `uname -i` to obtain *platform-name*.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also kmdb(1), passwd(1), sh(1), svcs(1), tip(1), uname(1), boot(1M), kernel(1M), init(1M), svcadm(1M), attributes(5), smf(5)

OpenBoot 3.x Command Reference Manual

ONC+ Developer's Guide

Name efdaemon – embedded FCode interpreter daemon

Synopsis

```
/usr/lib/efcode/sparcv9/efdaemon [-d]
```

Description `efdaemon`, the embedded `FCode` interpreter daemon, invokes the embedded `FCode` interpreter when the daemon receives an interpretation request. A new session of the interpreter is started for each unique request by invoking the script `/usr/lib/efcode/efcode`.

`efdaemon` is used on selected platforms as part of the processing of some dynamic reconfiguration events.

Options The following option is supported:

`-d` Set debug output. Log debug messages as `LOG_DEBUG` level messages by using `syslog()`. See `syslog(3C)`.

Files

`/dev/fcode`	`FCode` interpreter pseudo device, which is a portal for receipt of `FCode` interpretation requests
`/usr/lib/efcode/efcode`	Shell script that invokes the embedded `FCode` interpreter
`/usr/lib/efcode/interpreter`	Embedded `FCode` interpreter
`/usr/lib/efcode/sparcv9/interpreter`	Embedded `FCode` interpreter

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWefcx, SUNWefcux, system/embedded-fcode-interpreter, SUNWefclx

See Also `svcs(1)`, `prtconf(1M)`, `svcadm(1M)`, `syslog(3C)`, `attributes(5)`, `smf(5)`

Notes The `efdaemon` service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/platform/sun4u/efdaemon:default
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. The service's status can be queried using the `svcs(1)` command.

Name embedded_su – allow an application to prompt for credentials and execute commands as the super user or another user

Synopsis

```
/usr/lib/embedded_su [-] [username [arg...]]
```

Description The `embedded_su` command allows an application to prompt the user for security credentials and then use those credentials to execute a program as another user or role (see `rbac(5)` for information on role-based access control). The default *username* is `root` (super user).

`embedded_su` is identical to `su(1M)`, except that the user interaction is packaged in a form suitable for another program to interpret and display. Typically, `embedded_su` would be used to allow a graphical program to prompt for the super user password and execute a command as the super user, without requiring that the requesting program be run as the super user.

PROTOCOL `embedded_su` implements a simple protocol over standard input, standard output, and standard error. This protocol consists of three phases, roughly corresponding to PAM initialization, the PAM dialog, and PAM completion.

Phase 1: Initialization

After starting `embedded_su`, the application must send an initialization block on `embedded_su`'s standard input. This block is a text block, as described under "Text Blocks". There are currently no initialization parameters defined; the application should send an empty block by sending a line consisting solely of a period (.).

Phase 2: Conversation

`embedded_su` then emits zero or more conversation blocks on its standard output. Each conversation block may require zero or more responses.

A conversation block starts with a line consisting of the word `CONV`, followed by whitespace, followed by the number of messages in the conversation block as a decimal integer. The number of messages may be followed by whitespace and additional data. This data, if present, must be ignored.

Each message consists of a line containing a header followed by a text block, as described under "Text Blocks". A single newline is appended to each message, allowing the message to end with a line that does not end with a newline.

A message header line consists of a PAM message style name, as described in `pam_start(3PAM)`. The message header values are:

`PAM_PROMPT_ECHO_OFF` The application is to prompt the user for a value, with echoing disabled.

`PAM_PROMPT_ECHO_ON` The application is to prompt the user for a value, with echoing enabled.

`PAM_ERROR_MSG` The application is to display the message in a form appropriate for displaying an error.

PAM_TEXT_INFO — The application is to display the message in a form appropriate for general information.

The PAM message style may be followed by whitespace and additional data. This data, if present, must be ignored.

After writing all of the messages in the conversation block, if any of them were PAM_PROMPT_ECHO_OFF or PAM_PROMPT_ECHO_ON, embedded_su waits for the response values. It expects the response values one per line, in the order the messages were given.

Phase 3: Completion

After zero or more conversation blocks, embedded_su emits a result block instead of a conversation block.

Upon success, embedded_su emits a single line containing the word "SUCCESS". The word SUCCESS may be followed by whitespace and additional data. This data, if present, must be ignored.

Upon failure, embedded_su emits a single line containing the word "ERROR", followed by a text block as described under "Text Bocks". The text block gives an error message. The word ERROR may be followed by whitespace and additional data. This data, if present, must be ignored.

Text Blocks

Initialization blocks, message blocks, and error blocks are all text blocks. These are blocks of text that are terminated by a line containing a single period (.). Lines in the block that begin with a "." have an extra "." prepended to them.

Internationalization — All messages are localized to the current locale; no further localization is required.

SECURITY — embedded_su uses pam(3PAM) for authentication, account management, and session management. Its primary function is to export the PAM conversation mechanism to an unprivileged program. Like su(1M), the PAM configuration policy can be used to control embedded_su. The PAM service name used is "embedded_su".

embedded_su is almost exactly equivalent to su(1M) for security purposes. The only exception is that it is slightly easier to use embedded_su in writing a malicious program that might trick a user into providing secret data. For those sites needing maximum security, potentially at the expense of application functionality, the EXAMPLES section shows how to disable embedded_su.

Examples

In the following examples, left angle brackets (<<<) indicate a line written by embedded_su and read by the invoking application. Right angle brackets (>>>) indicate a line written by the application and read by embedded_su.

EXAMPLE 1 Executing a command with the Correct Password

The following example shows an attempt to execute "somecommand" as "someuser", with the correct password supplied:

```
/usr/lib/embedded_su someuser -c somecommand
   >>>.
   <<<CONV 1
   <<<PAM_PROMPT_ECHO_OFF
   <<<Password:
   <<<.
   >>>[ correct password ]
   <<<SUCCESS
   [ somecommand executes  ]
```

EXAMPLE 2 Executing a command with the Incorrect Password

The following example shows an attempt to execute "somecommand" as "someuser", with the incorrect password supplied:

```
/usr/lib/embedded_su someuser -c somecommand
   >>>.
   <<<CONV 1
   <<<PAM_PROMPT_ECHO_OFF
   <<<Password:
   <<<.
   >>>[ incorrect password ]
   [ delay ]
   <<<ERROR
   <<<embedded_su:Sorry
   <<<.
   [ exit ]
```

EXAMPLE 3 Message Examples

A `pam_message` structure with *msg_style* equal to `PAM_TEXT_INFO` and `msg` equal to "foo" produces:

```
PAM_TEXT_INFO
foo
.
```

A `pam_message` structure with *msg_style* equal to `PAM_ERROR_MESSAGE` and `msg` equal to "bar\n" produces:

```
PAM_ERROR_MESSAGE
bar
[ blank line ]
.
```

EXAMPLE 3 Message Examples *(Continued)*

A `pam_message` structure with *msg_style* equal to `PAM_ERROR_MESSAGE` and `msg` equal to "aaa\nbbb" produces:

```
PAM_ERROR_MESSAGE
aaa
bbb
.
```

A `pam_message` structure with *msg_style* equal to `PAM_TEXT_INFO` and `msg` equal to empty quotation marks ("") produces:

```
PAM_TEXT_INFO
[ blank line ]
.
```

A `pam_message` structure with *msg_style* equal to `PAM_TEXT_INFO` and `msg` equal to NULL produces:

```
PAM_TEXT_INFO
.
```

EXAMPLE 4 Disabling `embedded_su`

To disable `embedded_su`, either add a line to the `/etc/pam.conf` file similar to:

```
embedded_su  auth  requisite  pam_deny.so.1
```

...or add the following entry to `/etc/pam.d/embedded_su`:

```
auth  requisite  pam_deny.so.1
```

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os
Interface Stability	Committed

See Also

`su(1M)`, `pam(3PAM)`, `pam_start(3PAM)`, `attributes(5)`, `rbac(5)`

Name emCCR – configure data collection for OCM

Synopsis

```
/usr/lib/ocm/ccr/bin/emCCR [automatic_update on|off]
     [clear -diagnostic[=SR= n[,FILE=path] [-c completed] [-f force]]
     [[-annotation="annotation string"] collect]
     [enable_diagchecks | disable_diagchecks]
     [enable_target | disable_target] [getupdates] [help] [hold]
     [register] [resume] [set collection_interval="string"] [start]
     [status [-diagnostic[=SR=n[,FILE=path]]]] [stop]
     [[-register] [-verbose] test]
     [update_components [[-silent] -staged_dir="dir"| -distribution"name"]]
     [upload -diagnostic=SR=n,FILE=path [-restart] [-force]]
```

Description The `emCCR` utility is used to configure the collection of data for the Oracle Configuration Manager (OCM). `emCCR` is implemented as a set of subcommands, with options to those subcommands.

Sub-commands The following options are supported:

`emCCR automatic_update [on|off]`
Enable or disable the automatic retrieval of new software updates from automatic collections. By default, automatic updates are enabled.

`emCCR clear -diagnostic`

`clear -diagnostic`
Clears all diagnostic upload files.

`clear -diagnostic=`*SR*`=`*x*`-`*y*
Clears all uploads for a particular service request. The SR number must be in the format *x*-*y* where:

x represents one digit

y represents multiple digits

`clear -diagnostic=`*SR*`=`*x*`-`*y*`,FILE=`*path*
Clears a particular upload. *path* must include the full path to the diagnostic package, such as `/scratch/test/support_info.zip`.

`clear -diagnostic -completed`
Clears completed uploads only.

`clear -diagnostic=`*SR*`=`*x*`-`*y*`,FILE=`*path* `-completed`
Clears completed uploads for a particular service request. If `FILE=`*path* is added, only the upload for that instance is cleared.

`clear -diagnostic -force`
Clears all uploads even if there are errors.

`clear -diagnostic=`*SR*`=`*x-y*`,FILE=`*path* `-force`
Clears all uploads for a particular service request. If `FILE=`*path* is added, only the upload for that instance is cleared.

`emCCR collect`
In connected mode, this command will perform an immediate discovery, collection, and uploading of configuration data. In disconnected mode, this command will perform a manual discovery and collection of configuration data. The data is stored in:

```
... /ccr/state/upload/ocmconfig.jar
```

`-annotation=`"*string*"
Adds a note, consisting of "*string*", to the data collected.

`emCCR` [`enable_diagchecks` | `disable_diagchecks`]
Use this command to enable or disable all diagnostic checks collections.

`emCCR` [`enable_target` | `disable_target`]
Use this command to enable or disable the collection of configuration information for targets discovered by the Oracle Configuration Manager. When you enter this command, the list of targets that can be enabled or disabled is displayed. Enter the number for the target that is to be enabled or disabled. If you press Return without entering a target number no enabling or disabling occurs.

`emCCR getupdates`
Use this command to retrieve any new software updates from the content server and deploy these updates.

`emCCR help`
Use this command to list the available options.

`emCCR hold`
Use this command to put Oracle Configuration Manager on hold. When in hold, configuration data will not be automatically collected and uploaded to the Oracle repository. You can manually collect and upload the information with the `collect` command. To restart the collection of data, use the `resume` command.

`emCCR register`
Use this command to reregister any information that has changed.

`emCCR resume`
Use this command to resume the automatic collection and uploading of the configuration data.

`emCCR set collection_interval=`"*string*"
Use this command to define the schedule to collect configuration data. By default the collection is daily at the time the collector was first configured. The data can be collected daily, weekly, monthly or the time of collection can be changed as defined by *string* where:

`FREQ=DAILY`[`\;BYHOUR=`0-23][`\;BYMINUTE=`0 to 59]
Selects daily collections. You can optionally set the hour and minute to do the collection.

`FREQ=WEEKLY\;BYDAY=MON` to `SUN` [`\;BYHOUR=0-23`][`\;BYMINUTE=0` to 59]
Selects weekly collections. You can optionally set the day, hour and minute to do the collection.

`FREQ=MONTHLY\;BYMONTHDAY=1` to 31 [`\;BYHOUR=0-23`][`\;BYMINUTE=0` to 59]
Selects monthly collections. You can optionally set the date, hour and minute to do the collection.

[`\;BYHOUR=0-23`][`\;BYMINUTE=0` to 59]
Selects a new time to collect the data. The frequency will not be changed.

`emCCR start`
Use this command to start the scheduler.

`emCCR status` [`-diagnostic`]

`status`	Reports the status of the collector.
`status -diagnostic`	Reports the status of all diagnostics uploads.
`status -diagnostic=`*SR*`=`*x*`-`*y*	Reports all uploads for a particular service request. The *SR* number must be in the format *x*-*y* where: *x* represents one digit *y* represents multiple digits
`status -diagnostic=`*SR*`=`*x*`-`*y*`,FILE=`*path*	Reports a particular upload. *path* must include the full path to the diagnostic file.

`emCCR stop`
Use this command to stop the scheduler.

`emCCR` [`-register`] [`-verbose`] `test`
Use this command to test the connection to the server at Oracle, where:

`-register` Specifies that the client will be registered during the test.

`-verbose` Selects to display detailed information about the connection process.

`emCCR update_components`
Use this command to deploy any updates to the Oracle Configuration Manager client and to deploy the Diagnostic Check package.

[`-silent`] `-staged_dir=`"*dir*"
Specifies the directory in which the Oracle Configuration Manager packages have been staged.

`-silent` Specifies that no output be displayed to screen.

[`-silent`] `-distribution=`"*path*"
Specifies the path to the Oracle Configuration Manager Kit to be used for deployment.

`-silent` Specifies that no output be displayed to screen.

`emCCR upload -diagnostic`

`upload -diagnostic=`*SR*`=`*x*`-`*y*`,FILE=`*path*
Uploads diagnostic information for a particular instance. The SR number must be in the format *x*-*y* where:

x represents one digit

y represents multiple digits

The path must include the full path to the diagnostic file.

`upload -diagnostic=`*SR*`=`*x*`-`*y*`,FILE=`*path* `-restart`
Continues the upload for a particular service request. If `FILE=`*path* is added, only the upload for that instance is done.

`upload -diagnostic=`*SR*`=`*x*`-`*y*`,FILE=`*path* `-force`
Restarts the upload for a particular service request. If `FILE=`*path* is added, only the upload for that instance is done.

Files

`/usr/lib/ocm`
Directory to store all OCM related files and data.

`/usr/lib/ocm/ccr/config`
Contains configuration files.

`/usr/lib/ocm/ccr/log`
Contains log files.

`/usr/lib/ocm/ccr/state`
Contains state files and collected data.

`/usr/lib/ocm/ccr/state/review/targetMap.xml`
Summaries of the data that was collected.

`/usr/lib/ocm/ccr/state/review`
Data that was uploaded to the server.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWocm

See Also

`configCCR(1M)`, `emocmrsp(1M)`, `attributes(5)`

Oracle Configuration Manager Installation and Adminstration Guide

Name emocmrsp – create an OCM response file

Synopsis

```
/usr/lib/ocm/ccr/bin/emocmrsp [-help] [-no_banner] [-output file]
     [-repeater URL] [-verbose file] [-verify file]
     [[CSI-number] [MyOracleSupportID]]
```

Description The `emocmrsp` command creates a response file that can be used to configure the Oracle Configuration Manager (OCM). This utility walks you through the interrogation phase of an installation and records your responses to the prompts. The information is recorded in an OCM private format response file. By default, the response file is created in the current directory with the filename `ocm.rsp`.

Options The following options are supported:

`-help`
Displays usage information.

`-no_banner`
Indicates that the banner for the response utility is not to be displayed.

`-output` *file*
Creates a response file with the specified name. By default, the response file is created in your current working directory with the name `ocm.rsp`.

`-repeater` *URL*
Defines the URL of the Oracle Support Hub.

`-verbose` *file*
Displays the contents of the specified response file.

`-verify` *file*
Verifies that the contents of the specified response file are valid.

Operands *CSI-number*
Customer Support Identifier

MyOracleSupportID
My Oracle Support User Name

Files `/usr/lib/ocm`
Directory to store all OCM related files and data.

`/usr/lib/ocm/ccr/config`
Contains configuration files.

`/usr/lib/ocm/ccr/log`
Contains log files.

`/usr/lib/ocm/ccr/state`
Contains state files and collected data.

`/usr/lib/ocm/ccr/state/review/targetMap.xml`
Summaries of the data that was collected.

`/usr/lib/ocm/ccr/state/review`
Data that was uploaded to the server.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWocm

See Also `configCCR(1M)`, `emCCR(1M)`, `attributes(5)`

Oracle Configuration Manager Installation and Adminstration Guide

Name etrn – start mail queue run

Synopsis

```
etrn [-b] [-v] server-host [client-hosts]
```

Description SMTP's ETRN command allows an SMTP client and server to interact, giving the server an opportunity to start the processing of its queues for messages to go to a given host. This is meant to be used in start-up conditions, as well as for mail nodes that have transient connections to their service providers.

The etrn utility initiates an SMTP session with the host *server-host* and sends one or more ETRN commands as follows: If no *client-hosts* are specified, etrn looks up every host name for which sendmail(1M) accepts email and, for each name, sends an ETRN command with that name as the argument. If any *client-hosts* are specified, etrn uses each of these as arguments for successive ETRN commands.

Options The following options are supported:

`-b` System boot special case. Make sure localhost is accepting SMTP connections before initiating the SMTP session with server-host.

This option is useful because it prevents race conditions between sendmail(1M) accepting connections and *server-host* attempting to deliver queued mail. This check is performed automatically if no *client-hosts* are specified.

`-v` The normal mode of operation for etrn is to do all of its work silently. The `-v` option makes it verbose, which causes etrn to display its conversations with the remote SMTP server.

Environment Variables No environment variables are used. However, at system start-up, `svc:/network/smtp:sendmail` reads `/etc/default/sendmail`. In this file, if the variable ETRN_HOSTS is set, `svc:/network/smtp:sendmail` parses this variable and invokes etrn appropriately. ETRN_HOSTS should be of the form:

```
"s1:c1.1,c1.2        s2:c2.1 s3:c3.1,c3.2,c3.3"
```

That is, white-space separated groups of *server:client* where *client* can be one or more comma-separated names. The *:client* part is optional. *server* is the name of the server to prod; a mail queue run is requested for each *client* name. This is comparable to running:

```
/usr/lib/sendmail -qR client
```

on the host *server*.

Examples **EXAMPLE 1** Using etrn

Inserting the line:

```
ETRN_HOSTS="s1.domain.com:clnt.domain.com s2.domain.com:clnt.domain.com"
```

EXAMPLE 1 Using `etrn` *(Continued)*

in `/etc/default/sendmail` results in `svc:/network/smtp:sendmail` invoking `etrn` such that ETRN commands are sent to both `s1.domain.com` and `s2.domain.com`, with both having `clnt.domain.com` as the ETRN argument.

The line:

```
ETRN_HOSTS="server.domain.com:client1.domain.com,client2.domain.com"
```

results in two ETRN commands being sent to `server.domain.com`, one with the argument `client1.domain.com`, the other with the argument `client2.domain.com`.

The line:

```
ETRN_HOSTS="server1.domain.com server2.domain.com"
```

results in set of a ETRN commands being sent to both `server1.domain.com` and `server2.domain.com`; each set contains one ETRN command for each host name for which `sendmail(1M)` accepts email, with that host name as the argument.

Files

`/etc/mail/sendmail.cf`	`sendmail` configuration file
`/etc/default/sendmail`	Variables used by `svc:/network/smtp:sendmail`

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	service/network/smtp/sendmail
Interface Stability	Committed

See Also `sendmail(1M)`, `attributes(5)`

RFC 1985

Notes Not all SMTP servers support ETRN.

Name fbconfig – frame buffer configuration utility

Synopsis

```
fbconfig [-dev device_file] [-help]

fbconfig -list

fbconfig -gui

fbconfig -res \?

fbconfig [-dev device_file] [-file machine | system]
     [device_specific_options] [-res video-mode] [defaults]
     [-prconf] [-prdid[raw] [parsed]] [-propt]
```

Description `fbconfig` is the generic command line interface to query and/or to configure a frame buffer device. Depending on the command line options, `fbconfig` can invoke a GUI program, or it can invoke a device-specific configuration program. The choice of device configuration program depends both on the device (specified with `-dev`) and on the currently configured X server.

The options recognized by `fbconfig` are shown in the first three command forms in the `SYNOPSIS` section. The remaining forms illustrate options that may be passed to a device-specific configuration program that performs the actual operations. The interpretation of these options will depend upon the specific configuration program that is invoked. The options shown are supported by most device configuration programs.

If the `-dev` option is omitted, the default frame buffer (`/dev/fb` or `/dev/fb0`) is assumed.

Options The following options are supported by `fbconfig`:

`-dev` *device_file*
: Specify the frame buffer device by its full pathname or simple filename. Pathnames of installed devices can be displayed using the `-list` option. If the `-dev` option is omitted, the default device is `/dev/fb`.

`-gui`
: Invoke the Graphical User Interface (GUI). The GUI is available if the `SUNWdcm` package is installed. All other options are ignored.

  The GUI can configure devices (as an alternative to the `fbconfig` command line) and can update the `Xservers` file without directly editing the file. The GUI allows the user that is logged in on the graphics device or devices to configure which graphics displays the window system should use, their screen layout (where they appear on the user's desktop), and screen properties (X attributes).

  In addition, the GUI allows advanced users to create a new video format (resolution) that some graphics devices can select from `fbconfig` command line or from the device-dependent portion of the GUI. The GUI's online help explains all options and features.

`-help`
Displays the supported `fbconfig` command line options, along with a brief explanation of each. Also displays the `-help` text of the device-specific configuration program, if any. The frame buffer device can be specified using the `-dev` option, otherwise the default is used. Other `fbconfig` options are ignored. This is the default `fbconfig` option.

`-list`
Displays the pathnames of the installed frame buffer devices, the device model of each, and the configuration program that would be invoked for each device with the currently configured X server. Other `fbconfig` options are ignored.

```
Device File Name            Device Model     Config Program
----------------            ------------     --------------
/dev/fbs/kfb0               XVR-2500         SUNWkfb_config
/dev/fbs/kfb1               XVR-2500         SUNWkfb_config
/dev/fbs/nfb0 [a|b]         XVR-300          SUNWnfb_config
/dev/fbs/pfb0 [a|b]         XVR-100          SUNWpfb_config
```

The following options are commonly supported by the device-specific configuration programs:

`-defaults`
Sets configuration options to their default values.

`-file machine | system`
Specifies which `xorg.conf` configuration file to use.

`-prconf`
Display the current configuration for the frame buffer and display device(s).

`-predid [raw] [parsed]`
Display the E-EDID (Enhanced Extended Display Identification Data) information obtained from the display device(s), which must be online, connected to the frame buffer. The output will be raw hexadecimal and/or human-readable text. The default is `parsed`.

`-propt`
Display the current software configuration.

`-res \?`
Display the video modes (resolutions) that can be used with the `-res` *video-mode* option.

Other supported options are determined by the device-specific configuration program:

device-specific-options
The syntax and descriptions of additional device-specific options are displayed in the `-help` output of `fbconfig` or the device-specific program. They are also contained in the man page for the device-specific program.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/graphics/fbconfig

See Also fbconf_xorg(1M), attributes(5)

Xserver(1), *Xorg(1)*, *Xsun(1)*, which are part of the OpenWindow, not the SunOS, man page collection.

Limitations Because of limitations in the m64 kernel driver and related software, fbconfig (with the -prconf option) is unable to distinguish between a current depth of 24 or 8+24. The -propt option returns the depth specified in the OWconfig file, which will be in effect following the next restart of the window system. The xwininfo utility, usually shipped in the package containing frame buffer software (such as SUNWxwplt), reports current depth of a specified window.

Name fbconf_xorg – configure frame buffer devices for Xorg

Synopsis

```
/usr/lib/fbconfig/fbconf_xorg [-dev device-file]
     [-file machine | system | config-path]
     [-res video-mode [nocheck | noconfirm]]
     [device-specific-options]
     [-defaults] [-prconf] [-predid [raw] [parsed]] [-propt]

/usr/lib/fbconfig/fbconf_xorg [-dev device-file] [-prconf] [-propt]

/usr/lib/fbconfig/fbconf_xorg [-dev device-file] [-help] [-res ?]
```

Description The `fbconf_xorg` utility configures Frame Buffer devices and some of the X11 window system defaults for Xorg by updating the `xorg.conf` configuration file. Users should not normally need to run `fbconf_xorg` directly, as it is executed by `fbconfig(1M)` when the Xorg server is in use.

The first `fbconf_xorg` command form shown in the `SYNOPSIS` section stores the specified option values in the `xorg.conf` configuration file. These settings are used to initialize the frame buffer device the next time the window system is run on that device. The persistence of these settings across window system sessions and system reboots is determined by the `xorg.conf` file.

The second and third command forms display information, and do not alter the `xorg.conf` file. The presence of the `-help` and/or `-res ?` options causes any other option (not shown in the third form) to be ignored.

You can configure only one frame buffer device at a time. To configure multiple devices, invoke the `fbconf_xorg` utility separately for each device.

Only frame buffer device options can be specified with `fbconf_xorg`. Use the normal window system options to specify default depth (see `svccfg(1M)`), default visual class, and so forth. Specify these as device modifiers on the command line as specified in *Xserver(1)*.

You can specify which `xorg.conf` file to open. By default, `fbconf_xorg` opens the machine-specific file, `/etc/X11/xorg.conf`. Use the `-file` option to specify an alternate file. For example, the system-global file, `/usr/lib/X11/xorg.conf`, can be opened instead.

These standard `xorg.conf` files can be written only by the superuser or someone with the Desktop Configuration role.

Options The following options are supported for all frame buffer devices:

`-defaults`
: Sets configuration options for the specified device to their default values. This does not affect the `-res` video mode setting. See the device-specific portions of the `DEFAULTS` section below.

`-dev` *device-file*
Specifies the frame buffer device by either its full pathname or simple filename (for example, `/dev/fbs/efb0` or `efb0`). Pathnames of installed devices can be displayed using the `-list` option to `fbconfig(1M)`. If the `-dev` option is omitted, the default device, `/dev/fb`, is used.

`-file machine | system |` *config-path*
Specifies which `xorg.conf` file to open. If `machine` is specified, the machine-specific `/etc/X11/xorg.conf` file is opened. If system is specified, the global `/usr/lib/X11/xorg.conf` file is opened. The absolute pathname of a configuration file can be used instead. If the specified file does not exist and is to be updated, it is created. The file system that contains the `xorg.conf` file must be writeable by someone with superuser-like privileges. This option has no effect unless other options are specified. The default is `machine`.

`-help`
Display the `fbconf_xorg` command line options that are supported in conjunction with the frame buffer device, along with a brief explanation of each option. The frame buffer device can be specified using the `-dev` option.

`-prconf`
Display the current configuration for the frame buffer device and attached display device(s). The frame buffer device can be specified using the `-dev` option.

The `-prconf` output might resemble:

```
Monitor/Resolution Information:
    Monitor manufacturer: SUN
    Product Code:  4
    Serial Number: 12212555
    Manufacture date: 2000, week 9
    EDID Version:  1.1
    Monitor dimensions: 36x29 cm
    Default Gamma: 2.62
    Monitor preferred resolution: SUNW_STD_1280x1024x60
    Monitor supported resolutions from EDID: SUNW_STD_1280x1024x60,
        SUNW_STD_1280x1024x76, 1152x900x66, VESA_STD_1280x1024x75,
        VESA_STD_1280x1024x60, SUNW_STD_1152x900x66,
        VESA_STD_720x400x70, VESA_STD_640x480x60,
        VESA_STD_640x480x67, VESA_STD_640x480x72,
        VESA_STD_640x480x75, VESA_STD_800x600x56,
        VESA_STD_800x600x60, VESA_STD_800x600x72,
        VESA_STD_800x600x75, VESA_STD_832x624x75,
        VESA_STD_1024x768x60, VESA_STD_1024x768x70, 1024x768x75
    Current resolution setting: FALLBACK_1152x900x66

Framelock Configuration:
    Slave Mode: Disabled
```

`-predid` [`raw`] [`parsed`]
Display the E-EDID (Enhanced Extended Display Identification Data) information obtained from the display device(s), which must be online, connected to the frame buffer. The frame buffer device can be specified using the `-dev` option. The output is raw hexadecimal and/or human-readable (parsed) text. The default is parsed.

The `-predid` raw output might resemble:

```
--- EDID Data for /dev/fbs/kfb0 ---

Block 0: EDID Base Block
        0x00:  00 FF FF FF FF FF FF 00 04 43 06 F2 01 00 00 00
        0x10:  01 11 01 04 0F 2B 20 78 2B 9C 68 A0 57 4A 9B 26
        0x20:  12 48 4C FF FF 80 A9 59 A9 4F A9 4A A9 45 81 99
        0x30:  81 80 61 59 45 59 48 3F 40 30 62 B0 32 40 40 C0
        0x40:  13 00 AB 40 11 00 00 1E 00 00 00 FD 00 32 5A 1E
        0x50:  6E 17 04 11 00 C8 90 00 50 3C 00 00 00 F7 00 0A
        0x60:  F7 0F 03 87 C0 00 00 00 00 00 00 00 00 00 00 FC
        0x70:  00 41 42 43 20 4C 43 44 32 31 0A 20 20 20 00 0B
```

`-propt`
Display all option settings for the frame buffer device, either as they currently are or as they are represented in the `xorg.conf` configuration file when `fbconf_xorg` completes. The device can be specified using the `-dev` option, and the file using the `-file` option.

The `-propt` output might resemble:

```
--- Graphics Configuration for /dev/fbs/efb0 ---

xorg.conf: machine -- /etc/X11/xorg.conf
      Screen section:  "efb0"
      Device section:  "efb0"
      Monitor section: none

Video Mode: Not set

Screen Information:
      DoubleWide: Disable
      DoubleHigh: Disable
      Clone: Disable
      Offset/Overlap: [0, 0]
      Outputs:        Direct

Visual Information:
      Gamma Correction: Using default gamma value 2.22
```

`-res ?`
Display a list of video modes that can be used with the `-res` *video-mode* option.

The ? argument might need to be escaped or placed in quotes (\?, "?", or '?'), to protect it from misinterpretation by the shell.

The -res ? output might resemble:

```
Video modes accepted by the -res option:
     AUTO                      [1][2]
     NONE                      [1][2]
     SUNW_STD_1920x1200x75
     SUNW_STD_1920x1200x70
     SUNW_DIG_1920x1200x60
     SUNW_STD_1920x1080x72
     SUNW_DIG_1920x1080x60
       ...
       ...
     VESA_STD_640x480x75       [1]
     VESA_STD_640x480x72       [1]
     VESA_STD_640x480x60       [1]

[1] Resolution is supported by monitor
[2] Preferred resolution for monitor
```

Abbreviations such as "1280x1024x75" might also be used.

-res *video-mode* [nocheck | noconfirm]
Set the video mode for the display device that is connected to the frame buffer device.

A list of video modes can be displayed using the -res ? option.

The basic format of a video-mode is *width*x*height*x*rate*, where:

- *width* is the screen width in pixels.
- *height* is the screen height in pixels.
- *rate* is the vertical frequency of the screen refresh.

A video-mode argument might have an @ (at sign) instead of x preceding the refresh rate. For instance, 1280x1024x76 and 1280x1024@76 are equivalent.

A video-mode name might carry additional information, as with SUNW_STD_1280x1024x76.

The -res argument, auto, represents the video mode that is currently programmed into the device. The argument, none, is a synonym for auto.

Note that some video modes might be inappropriate for certain frame buffer devices and/or display devices.

The -res option accepts suboption keywords following the video-mode specification.

nocheck
The nocheck suboption causes the video-mode argument to be accepted, regardless of whether it is supported by the currently attached monitor, whether it is known within

the current configuration, and so forth. Note that using an unchecked, inappropriate video mode can leave the system without usable video output. This suboption is useful if a different monitor is to be connected to the frame buffer device. This suboption also implies `noconfirm`.

`noconfirm`
If the video-mode argument is unable to be validated, the default action is to display a warning message and ask the user whether to continue. The `noconfirm` suboption suppresses this confirmation request. This suboption is useful when `fbconf_xorg` is being run from a shell script.

The following device-specific options are supported for certain frame buffer devices. Unless specified otherwise, these options do not take effect until the user logs out and back in.

`-deflinear true | false`
This option selects the default X visual. Two types of visuals are supported, linear and nonlinear. Linear visuals are gamma corrected. Nonlinear visuals are not.

If the value of this option is `true`, the default visual is set to default depth 24 and the default class is `TrueColor` with gamma correction enabled. If `false`, a nonlinear visual that satisfies the other default visual selection options, such as the default depth and default class, is chosen as the default visual.

The `-deflinear`, `-defoverlay`, and `-deftransparent` options each select the default X visual. Only one of these might be enabled at a time. Enabling one causes the others to be disabled.

`-defoverlay true | false`
This option selects the default X visual. Some devices might provide an 8-bit PseudoColor visual whose pixels are disjoint from the rest of the visuals. This is called the overlay visual. Windows created in this visual does not damage windows created in other visuals. The converse, however, is not true: Windows created in other visuals damage overlay windows.

If the value of this option is `true`, the overlay visual is the default visual. The default depth is 8–bit and the default class is PseudoColor. If `false`, the non-overlay visual that satisfies the other default visual selection options, such as the default depth and the default class, is chosen as the default visual.

The `-deflinear`, `-defoverlay`, and `-deftransparent` options each select the default X visual. Only one of these might be enabled at a time. Enabling one causes the others to be disabled.

`-deftransparent true | false`
This option selects the default X visual. Some devices might provide an 8-bit PseudoColor visual whose pixels are disjoint from the rest of the visuals. This is called the overlay visual. Windows created in this visual does not damage windows created in other visuals.

If the value of this option is `true`, the overlay visual used as the default is a transparent overlay visual. A visual with transparency supports a colormap with 255 colors and one transparent pixel. The default depth is 8–bit and the default class is PseudoColor. If `false`, the nonoverlay visual that satisfies the other default visual selection options, such as the default depth and the default class, is chosen as the default visual.

The `-deflinear`, `-defoverlay`, and `-deftransparent` options each select the default X visual. Only one of these might be enabled at a time. Enabling one causes the others to be disabled.

`-doublehigh` `enable` | `disable`
Configures the two outputs of the frame buffer device into one vertical virtual display. The default is `disable`. The `-doublewide` and `-doublehigh` options are mutually exclusive. Enabling one causes the other to be disabled.

`-doublewide` `enable` | `disable`
Configures the two outputs of the frame buffer device into one horizontal virtual display. The default is `disable`. The `-doublewide` and `-doublehigh` options are mutually exclusive. Enabling one causes the other to be disabled.

`-g` *gamma-correction-value*
Sets the gamma correction value. All linear visuals provide gamma correction. The gamma correction value should be in the range, `0.1` to `10.0`. The default is `2.22`. This option can be used while the window system is running. Changing the gamma correction value affects all of the windows displayed by linear visuals.

`-gfile` *gamma-correction-file*
Loads the gamma correction table from the file specified by *gamma-correction-file*. This text file specifies the gamma correction values for the R, G, and B channels. Three consecutive values form an RGB triplet. For a `kfb` device, there must be exactly 256 RGB triplets. A value might be represented in hexadecimal, decimal, or octal format (for example, `0x3FF`, `1023`, or `01777`, respectively). Values are separated by one or more whitespace or new line characters. Comments begin with a hash sign character (#) and end at the end of the line.

You can load the gamma correction table with this option while the window system is running. The new gamma correction affects all the windows being displayed using the linear visuals. When gamma correction is done using a user-specified table, the gamma correction value (`-g`) is undefined. By default, the window system assumes a gamma correction value of `2.22` and loads the gamma table it creates corresponding to this value.

The following is an example of a *gamma-correction-file* file:

```
# Gamma Correction Table
0x00 0x00 0x00
0x01 0x01 0x01
0x02 0x02 0x02
 ...  ...  ...
```

```
                ...  ...  ...
               0xFF 0xFF 0xFF
```

`-multisample` `available` | `disable` | `forceon`
If set to disable, no multisample is possible. If set to available, multisample is possible but is selected on a per-window basis using a library interface. If set to `forceon`, all Sun OpenGL windows are rendered using multisampling. To query the number of samples used, specify the `-propt` option.

`-offset` *x-val y-val*
Adjusts the position of the specified stream by the value specified. This option is only implemented in `-doublewide` and `-doublehigh` modes. For `-doublewide`, use the *x-val* to position the rightmost stream. Negative is left (overlaps with the left stream). For `-doublehigh`, use the *y-val* to position the bottom stream. Negative is up (overlaps with top stream). The default is `[0,0]`.

`-samples` `1` | `2` | `4` | `8` | `16`
Requests the number of samples to compute per display pixel. The requested number of samples per pixel is used if `-multisample` is not disabled and resources exist for the request. To query the number of samples used, specify the `-propt` option or run the `xglinfo` utility. The `xglinfo` utility can return the number of multisamples after you specify the option `-multisample` available. The default is `4`.

`-slave` `disable` | `multiview`
If you set the `multiview` argument for the `-slave` option, the device synchronizes video with a master through the multiview genlock ribbon cable. The system should be powered off whenever connecting or disconnecting this cable. Both devices should be running the same resolution and the option should be issued when the window system is running. The default is `disable`.

Defaults Certain options have implied default arguments. The default argument is used when the option is not present on the `fbconf_xorg` command line. For instance, a default argument for `-dev` is `/dev/fb`.

Options that set configuration state do not have implied defaults. The `-res` option is one example. If a configuration option is omitted from the `fbconf_xorg` command line, the corresponding `xorg.conf` configuration setting remains unchanged. The exception is that if configuration options are mutually exclusive, setting one automatically unsets each of the others. An example is `-deflinear`, `-defoverlay`, and `-deftransparent`.

If a configuration setting is not present in the configuration file when the window system is run, a default value is used. For instance, the default state associated with `-res` is `auto`. A setting might not be present in the file, or the file itself might not exist, until `fbconf_xorg` has been invoked with the corresponding command line option.

The `-defaults` option sets the default values for most configuration settings.

Options and their defaults are shown below.

Device-independent defaults:

```
Option          Default Argument
-dev            /dev/fb
-file           machine

Option          Default State
-res            auto
```

Examples

EXAMPLE 1 Switching the Resolution of the Monitor Type

The following example sets the video mode for the monitor on the `/dev/fbs/efb0` device to 1280 x 1024 at 76 Hz:

```
example% fbconf_xorg -dev efb0 -res 1280x1024x76
```

Exit Status

The following exit values are returned:

`0`
Execution completed successfully.

`1`
Invalid command line usage.

`2`
An error occurred.

Files

`/dev/fb`	Symbolic link to the default frame buffer device.
`/dev/fbs/efbn`*n*	Device special file for an `efb` frame buffer
`/usr/lib/fbconfig/SunModes_xorg.conf`	Video mode definitions included in new configuration files.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/graphics/fbconfig/fbconfig-kfb
Interface Stability	Uncommitted

See Also

`fbconfig(1M)`, `svccfg(1M)`, `attributes(5)`, `efb(7D)`, `fbio(7I)`

See the *Xorg(1)* and *Xserver(1)* man pages in the X Server man page collection.

Name fcinfo, fcadm – Fibre Channel HBA Port Command Line Interface

Synopsis

```
fcinfo hba-port [-lite] [HBA_port_WWN]...

fcadm hba-port [-lite] [HBA_port_WWN]...

fcinfo remote-port [-ls] [-p HBA_port_WWN]
     [REMOTE_port_WWN]...

fcadm remote-port [-ls] [-p HBA_port_WWN]
     [REMOTE_port_WWN]...

fcinfo logical-unit | lu [-v] [OS device_path]

fcadm logical-unit | lu [-v] [OS device_path]

fcadm create-npiv-port -p Virtual_Port_WWN [-n Virtual_Node_WWN]
     PhysicalPort_port_WWN

fcadm delete-npiv-port -p Virtual_Port_WWN [-n Virtual_Node_WWN]
     PhysicalPort_port_WWN

fcadm create-fcoe-port [-tf] -p Port_WWN [-n Node_WWN]
     MAC_Interface

fcadm delete-fcoe-port MAC_Interface

fcadm list-fcoe-ports MAC_Interface

fcadm force-lip Port_WWN

fcinfo [-V]

fcadm [-V]

fcinfo [-?]

fcadm [-?]
```

Description `fcinfo` and `fcadm` are command line interfaces that collect administrative information on fibre channel host bus adapter (HBA) ports on a host. They also collect data on any fibre channel targets that might be connected to those ports in a Storage Area Network (SAN).

A port can be either in initiator mode or target mode. The same FC attribute information and remote port information are applied to both the initiator and the target modes port except that SCSI-related operations described below do not apply for the target mode port.

SUBCOMMANDS The following subcommands are supported by both `fcinfo` and `fcadm`:

`hba-port` Lists information for the HBA port referenced by the specified *HBA_port_WWN*. If *HBA_port_WWN* is not specified, all initiator mode and target mode fibre channel HBA ports on the host will be listed.

`remote-port` Lists the *remote-port* information for those remote ports that are specified. If no *REMOTE_port_WWN* is specified, all remote ports that are visible through *HBA_port_WWN* are listed.

`logical-unit` | `lu` Lists the logical unit referenced by the specified *device_path*. If *device_path* is not specified, all fibre channel logical units will be listed. This subcommand applies only to the initiator mode.

The following subcommands, which administer N_Port ID Virtualization (NPIV) HBA ports, are supported only by `fcadm`:

`create-npiv-port`
Create an NPIV virtual port on the specified adapter.

`delete-npiv-port`
Delete an NPIV virtual port. This delete only ports create by `fcadm`.

The following subcommands, which administer Fibre Channel over Ethernet (FCoE) ports, are supported only by `fcadm`:

`create-fcoe-port`
Creates a FCoE port associated with the given MAC interface and the given *Port_WWN*/*Node_WWN*. If `-p` or `-n` is not specified, the Port WWN or Node WWN will be generated automatically. If the specified MAC interface does not support multiple unicast address, no FCoE port will be created and you receive a message indicating that you can add an `-f` option to force enabling promiscuous mode on the specified MAC interface to enable FCoE.

`delete-fcoe-port`
Deletes the FCoE port associated with the specified MAC interface.

`list-fcoe-ports`
List information of FCoE ports.

`force-lip`
Force the link to reinitialize. When issuing this subcommand on the target port side, causes a reset of the target port. When issuing it from the host port side, resets the host port. When an FC switch is connected, other FC ports in the SAN will get a RSCN (Remote State Change Notification). Furthermore, other initiators will always rediscover the port after this, and the FC login session will be established or reused. Also, I/Os will be continued. This command is disruptive to I/Os, but it is not destructive, as it does not cause any data loss.

Options The following options are supported:

`-e`, `--fcoe`
Lists the information for all FCoE ports.

`-f`, `--fcoe-promiscuous`
Used with `create-fcoe-port` to create an FCoE port on a MAC interface associated with a NIC that does not support multiple unicast address. Promiscuous mode will be enabled for the specified MAC interface.

`-l`, `--linkstat`
Lists the link error statistics information for the port referenced by the specified *HBA_port_WWN* or *REMOTE_port_WWN*.

`-n` *HBA_node_WWN*, `--node` *HBA_node_WWN*
When used with NPIV options, specify a virtual node WWN. If used with `create-npiv-port`, it can be omitted, and a random based WWN will be used. It is mandatory for `delete-npiv-port`.

When used with `create-fcoe-port` subcommand, specify the node WWN for the FCoE port. It can be omitted, in which case a WWN will be generated based on the MAC address of the specified MAC interface.

`-p` *HBA_port_WWN*, `--port` *HBA_port_WWN*
Retrieve remote port information from the *HBA_port_WWN* of the local HBA port on the host. When used with the `remote-port` subcommand, the `-p` option is mandatory.

When used with NPIV options, specify a virtual port WWN. If used with `create-npiv-port`, it can be omitted, and a random based WWN will be used. It is mandatory for `delete-npiv-port`.

When used with `create-fcoe-port` subcommand, specify the port WWN for the FCoE port. It can be omitted, in which case a WWN will be generated based on the MAC address of the specified MAC interface.

`-s`, `--scsi-target`
Lists the SCSI target information for all remote ports the user has asked for. The `-p`, `--port` option must always be specified and must be a valid HBA port on the host. This HBA port will be used as the initiator for which to retrieve the SCSI level target information. Note that this will only function on remote port fibre channel World-Wide Names that support an FC4 type of SCSI. This option applies only to an initiator mode port.

`-t` [*HBA_node_WWN*], `-target` [*HBA_node_WWN*]
Lists the information for the port with the target mode referenced by the specified *HBA_node_WWN* or all target mode ports discovered in the host.

When used with `create-fcoe-port`, create a FCoE target mode port.

`-v`, `--verbose`
When used with the `logical-unit` subcommand, the `-v` displays additional information for the logical unit, including SCSI vendor and product information and device type as well as the FC World-Wide names for the local and remote fibre channel ports to which this device is attached.

`-V`, `--version`
Displays the version information.

`-?`, `--help`
Displays the usage information.

Examples

EXAMPLE 1 Listing All HBA Ports

The following command lists all initiator mode fibre channel HBA ports on the host:

```
# fcinfo hba-port

HBA Port WWN: 210000e08b074cb5
        Port Mode: Initiator
        OS Device Name: /dev/cfg/c1
        Manufacturer: QLogic Corp.
        Model: 375-3108-xx
        Firmware Version: 3.3.116
        FCode/BIOS Version: 1.13.08
        Serial Number: not available
        Driver Name: qlc
        Driver Version: 20070212-2.19
        Type: N-port
        State: online
        Supported Speeds: 1Gb 2Gb
        Current Speed: 2Gb
        Node WWN: 200000e08b074cb5
NPIV Port List:
        Virtual Port 1:
         Port WWN: 200000e08b074cb1
         Node WWN: 200000e08b074cb3
HBA Port WWN: 210100e08b274cb5
        Port Mode: Initiator
        OS Device Name: /dev/cfg/c2
        Manufacturer: QLogic Corp.
        Model: 375-3108-xx
        Firmware Version: 3.3.116
        FCode/BIOS Version: 1.13.08
        Serial Number: not available
        Driver Name: qlc
        Driver Version: 20070212-2.19
        Type: N-port
        State: online
        Supported Speeds: 1Gb 2Gb
        Current Speed: 2Gb
        Node WWN: 200100e08b274cb5
HBA Port WWN: 210000e08b072ab5
        Port Mode: Initiator
        OS Device Name: /dev/cfg/c3
```

EXAMPLE 1 Listing All HBA Ports *(Continued)*

```
        Manufacturer: QLogic Corp.
        Firmware Version: 3.3.116
        FCode/BIOS Version: 1.13.08
        Model: 375-3108-xx
        Serial Number: not available
        Driver Name: qlc
        Driver Version: 20070212-2.19
        Type: L-port
        State: online
        Supported Speeds: 1Gb 2Gb
        Current Speed: 2Gb
        Node WWN: 200000e08b072ab5
HBA Port WWN: 210100e08b272ab5
        Port Mode: Initiator
        OS Device Name: /dev/cfg/c4
        Manufacturer: QLogic Corp.
        Model: 375-3108-xx
        Firmware Version: 3.3.116
        FCode/BIOS Version: 1.13.08
        Serial Number: 0402F00-0549112808
        Driver Name: qlc
        Driver Version: 20070212-2.19
        Type: N-port
        State: online
        Supported Speeds: 1Gb 2Gb
        Current Speed: 2Gb
        Node WWN: 200100e08b272ab5
```

EXAMPLE 2 Listing Target Mode HBA Ports

The following command lists all target mode fibre channel HBA ports on the host:

```
# fcinfo hba-port -t
  HBA Port WWN: 210100e08bb09221
     Port Mode: Target
     Port ID: 10700
     OS Device Name: Not Applicable
     Manufacturer: QLogic Corp.
     Model: d30ac7e0
     Firmware Version: 4.0.109
     FCode/BIOS Version: N/A
     Type: F-port
     State: online
     Supported Speeds: not established
     Current Speed: 2Gb
     Node WWN: 200100e08bb09221
  HBA Port WWN: 210000e08b909221
```

EXAMPLE 2 Listing Target Mode HBA Ports *(Continued)*

```
    Port Mode: Target
    Port ID: 10900
    OS Device Name: Not Applicable
    Manufacturer: QLogic Corp.
    Model: d37ad1e0
    Firmware Version: 4.0.109
    FCode/BIOS Version: N/A
    Type: F-port
    State: online
    Supported Speeds: not established
    Current Speed: 2Gb
    Node WWN: 200000e08b909221
  HBA Port WWN: 200000144fc2d508
    Port Mode: Target
    Port ID: 9a0025
    OS Device Name: Not Applicable
    Manufacturer: Sun Microsystems, Inc.
    Model: FCoE Virtual FC HBA
    Firmware Version: N/A
    FCode/BIOS Version: N/A
    Serial Number: N/A
    Driver Name: COMSTAR FCOET
    Driver Version: 1.0
    Type: F-port
    State: online
    Supported Speeds: 1Gb 10Gb
    Current Speed: 10Gb
    Node WWN: 100000144fc2d508
  HBA Port WWN: 200000144fc2d509
    Port Mode: Target
    Port ID: 9a0023
    OS Device Name: Not Applicable
    Manufacturer: Sun Microsystems, Inc.
    Model: FCoE Virtual FC HBA
    Firmware Version: N/A
    FCode/BIOS Version: N/A
    Serial Number: N/A
    Driver Name: COMSTAR FCOET
    Driver Version: 1.0
    Type: F-port
    State: online
    Supported Speeds: 1Gb 10Gb
    Current Speed: 10Gb
    Node WWN: 100000144fc2d509
```

EXAMPLE 3 Listing HBA Ports and Link Statistics

The following command lists information for the HBA ports and the link statistics for those ports:

```
# fcinfo hba-port -l 210000e08b074cb5 210100e08b274cb5

HBA Port WWN: 210000e08b074cb5
        Port Mode: Initiator
        OS Device Name: /dev/cfg/c1
        Manufacturer: QLogic Corp.
        Model: 375-3108-xx
        Firmware Version: 3.3.116
        FCode/BIOS Version: 1.13.08
        Serial Number: 0402F00-0549112808
        Driver Name: qlc
        Driver Version: 20070212-2.19
        Type: N-port
        State: online
        Supported Speeds: 1Gb 2Gb
        Current Speed: 2Gb
        Node WWN: 200000e08b074cb5
        Link Error Statistics:
                Link Failure Count: 0
                Loss of Sync Count: 0
                Loss of Signal Count: 0
                Primitive Seq Protocol Error Count: 0
                Invalid Tx Word Count: 0
                Invalid CRC Count: 0
HBA Port WWN: 210100e08b274cb5
        Port Mode: Initiator
        OS Device Name: /dev/cfg/c2
        Manufacturer: QLogic Corp.
        Model: 375-3108-xx
        Firmware Version: 3.3.116
        FCode/BIOS Version: 1.13.08
        Serial Number: 0402F00-0549112808
        Driver Name: qlc
        Driver Version: 20070212-2.19
        Type: N-port
        State: online
        Supported Speeds: 1Gb 2Gb
        Current Speed: 2Gb
        Node WWN: 200100e08b274cb5
        Link Error Statistics:
                Link Failure Count: 0
                Loss of Sync Count: 0
                Loss of Signal Count: 0
                Primitive Seq Protocol Error Count: 0
```

EXAMPLE 3 Listing HBA Ports and Link Statistics *(Continued)*

```
                Invalid Tx Word Count: 0
                Invalid CRC Count: 0
```

EXAMPLE 4 Listing All Remote Ports

The following command lists all remote ports that are visible through the given HBA port:

```
# fcinfo remote-port -p 210100e08b274cb5

Remote Port WWN: 50020f230000b4af
        Active FC4 Types: SCSI
        SCSI Target: yes
        Port Symbolic Name: unknown
        Node WWN: 50020f200000b4af
Remote Port WWN: 210000e08b07daa6
        Active FC4 Types: SCSI
        SCSI Target: no
        Port Symbolic Name: unknown
        Node WWN: 200000e08b07daa6
Remote Port WWN: 20030003ba27c788
        Active FC4 Types: SCSI
        SCSI Target: yes
        Port Symbolic Name: unknown
        Node WWN: 10000003ba27c788
Remote Port WWN: 210000e08b096a60
        Active FC4 Types: SCSI,IP
        SCSI Target: no
        Port Symbolic Name: unknown
        Node WWN: 200000e08b096a60
```

EXAMPLE 5 Listing Remote Ports and Link Statistics

The following command lists information for the remote ports and the link statistics for those ports:

```
# fcinfo remote-port -l -p 210100e08b272ab5

Remote Port WWN: 210100e08b296a60
        Active FC4 Types: SCSI,IP
        SCSI Target: no
        Port Symbolic Name: unknown
        Node WWN: 200100e08b296a60
        Link Error Statistics:
                Link Failure Count: 0
                Loss of Sync Count: 0
                Loss of Signal Count: 0
                Primitive Seq Protocol Error Count: 0
                Invalid Tx Word Count: 0
```

EXAMPLE 5 Listing Remote Ports and Link Statistics *(Continued)*

```
                Invalid CRC Count: 0
Remote Port WWN: 20030003ba27d56d
        Active FC4 Types: SCSI
        SCSI Target: yes
        Port Symbolic Name: unknown
        Node WWN: 10000003ba27d56d
        Link Error Statistics:
                Link Failure Count: 0
                Loss of Sync Count: 4765165
                Loss of Signal Count: 4765165
                Primitive Seq Protocol Error Count: 0
                Invalid Tx Word Count: 0
                Invalid CRC Count: 0
Remote Port WWN: 210100e08b27f7a6
        Active FC4 Types: SCSI
        SCSI Target: no
        Port Symbolic Name: unknown
        Node WWN: 200100e08b27f7a6
        Link Error Statistics:
                Link Failure Count: 0
                Loss of Sync Count: 0
                Loss of Signal Count: 0
                Primitive Seq Protocol Error Count: 0
                Invalid Tx Word Count: 0
                Invalid CRC Count: 0
Remote Port WWN: 50020f230000b897
        Active FC4 Types: SCSI
        SCSI Target: yes
        Port Symbolic Name: unknown
        Node WWN: 50020f200000b897
        Link Error Statistics:
                Link Failure Count: 0
                Loss of Sync Count: 7
                Loss of Signal Count: 7
                Primitive Seq Protocol Error Count: 0
                Invalid Tx Word Count: 0
                Invalid CRC Count: 0
Remote Port WWN: 210100e08b27daa6
        Active FC4 Types: SCSI
        SCSI Target: no
        Port Symbolic Name: unknown
        Node WWN: 200100e08b27daa6
        Link Error Statistics:
                Link Failure Count: 0
                Loss of Sync Count: 0
                Loss of Signal Count: 0
```

EXAMPLE 5 Listing Remote Ports and Link Statistics *(Continued)*

```
                Primitive Seq Protocol Error Count: 0
                Invalid Tx Word Count: 0
                Invalid CRC Count: 0
Remote Port WWN: 210000e08b074cb5
        Active FC4 Types: SCSI,IP
        SCSI Target: no
        Port Symbolic Name: unknown
        Node WWN: 200000e08b074cb5
        Link Error Statistics:
                Link Failure Count: 0
                Loss of Sync Count: 0
                Loss of Signal Count: 0
                Primitive Seq Protocol Error Count: 0
                Invalid Tx Word Count: 0
                Invalid CRC Count: 0
Remote Port WWN: 210100e08b296060
        Active FC4 Types: SCSI
        SCSI Target: no
        Port Symbolic Name: unknown
        Node WWN: 200100e08b296060
        Link Error Statistics:
                Link Failure Count: 0
                Loss of Sync Count: 0
                Loss of Signal Count: 0
                Primitive Seq Protocol Error Count: 0
                Invalid Tx Word Count: 0
                Invalid CRC Count: 0
```

EXAMPLE 6 Listing All SCSI Targets and Link Statistics

The following command lists all remote ports as well as the link statistics and *scsi-target* information:

```
# fcinfo remote-port  -sl -p 210100e08b272ab5

Remote Port WWN: 210100e08b296a60
        Active FC4 Types: SCSI,IP
        SCSI Target: no
        Port Symbolic Name: unknown
        Node WWN: 200100e08b296a60
        Link Error Statistics:
                Link Failure Count: 0
                Loss of Sync Count: 0
                Loss of Signal Count: 0
                Primitive Seq Protocol Error Count: 0
                Invalid Tx Word Count: 0
                Invalid CRC Count: 0
```

EXAMPLE 6 Listing All SCSI Targets and Link Statistics *(Continued)*

```
Remote Port WWN: 20030003ba27d56d
        Active FC4 Types: SCSI
        SCSI Target: yes
        Port Symbolic Name: unknown
        Node WWN: 10000003ba27d56d
        Link Error Statistics:
                Link Failure Count: 0
                Loss of Sync Count: 4765165
                Loss of Signal Count: 4765165
                Primitive Seq Protocol Error Count: 0
                Invalid Tx Word Count: 0
                Invalid CRC Count: 0
        LUN: 0
          Vendor: SUN
          Product: T4
          OS Device Name: /dev/rdsk/c4t20030003BA27D56Dd0s2
        LUN: 1
          Vendor: SUN
          Product: T4
          OS Device Name: /dev/rdsk/c4t20030003BA27D56Dd1s2
Remote Port WWN: 210100e08b27f7a6
        Active FC4 Types: SCSI
        SCSI Target: no
        Port Symbolic Name: unknown
        Node WWN: 200100e08b27f7a6
        Link Error Statistics:
                Link Failure Count: 0
                Loss of Sync Count: 0
                Loss of Signal Count: 0
                Primitive Seq Protocol Error Count: 0
                Invalid Tx Word Count: 0
                Invalid CRC Count: 0
Remote Port WWN: 50020f230000b897
        Active FC4 Types: SCSI
        SCSI Target: yes
        Port Symbolic Name: unknown
        Node WWN: 50020f200000b897
        Link Error Statistics:
                Link Failure Count: 0
                Loss of Sync Count: 7
                Loss of Signal Count: 7
                Primitive Seq Protocol Error Count: 0
                Invalid Tx Word Count: 0
                Invalid CRC Count: 0
        LUN: 0
          Vendor: SUN
```

EXAMPLE 6 Listing All SCSI Targets and Link Statistics *(Continued)*

```
    Product: T300
    OS Device Name: Unknown
  LUN: 1
    Vendor: SUN
    Product: T300
    OS Device Name: /dev/rdsk/c4t50020F230000B897d1s2
  LUN: 2
    Vendor: SUN
    Product: T300
    OS Device Name: /dev/rdsk/c4t50020F230000B897d2s2
  LUN: 3
    Vendor: SUN
    Product: T300
    OS Device Name: /dev/rdsk/c4t50020F230000B897d3s2
  LUN: 4
    Vendor: SUN
    Product: T300
    OS Device Name: /dev/rdsk/c4t50020F230000B897d4s2
  LUN: 5
    Vendor: SUN
    Product: T300
    OS Device Name: /dev/rdsk/c4t50020F230000B897d5s2
  LUN: 6
    Vendor: SUN
    Product: T300
    OS Device Name: /dev/rdsk/c4t50020F230000B897d6s2
  LUN: 7
    Vendor: SUN
    Product: T300
    OS Device Name: /dev/rdsk/c4t50020F230000B897d7s2
  LUN: 8
    Vendor: SUN
    Product: T300
    OS Device Name: /dev/rdsk/c4t50020F230000B897d8s2
  LUN: 9
    Vendor: SUN
    Product: T300
    OS Device Name: /dev/rdsk/c4t50020F230000B897d9s2
  LUN: 10
    Vendor: SUN
    Product: T300
    OS Device Name: /dev/rdsk/c4t50020F230000B897d10s2
  LUN: 11
    Vendor: SUN
    Product: T300
    OS Device Name: /dev/rdsk/c4t50020F230000B897d11s2
```

EXAMPLE 6 Listing All SCSI Targets and Link Statistics *(Continued)*

```
        LUN: 12
          Vendor: SUN
          Product: T300
          OS Device Name: /dev/rdsk/c4t50020F230000B897d12s2
        LUN: 13
          Vendor: SUN
          Product: T300
          OS Device Name: /dev/rdsk/c4t50020F230000B897d13s2
        LUN: 14
          Vendor: SUN
          Product: T300
          OS Device Name: /dev/rdsk/c4t50020F230000B897d14s2
        LUN: 15
          Vendor: SUN
          Product: T300
          OS Device Name: /dev/rdsk/c4t50020F230000B897d15s2
Remote Port WWN: 210100e08b27daa6
        Active FC4 Types: SCSI
        SCSI Target: no
        Port Symbolic Name: unknown
        Node WWN: 200100e08b27daa6
        Link Error Statistics:
                Link Failure Count: 0
                Loss of Sync Count: 0
                Loss of Signal Count: 0
                Primitive Seq Protocol Error Count: 0
                Invalid Tx Word Count: 0
                Invalid CRC Count: 0
Remote Port WWN: 210000e08b074cb5
        Active FC4 Types: SCSI,IP
        SCSI Target: no
        Port Symbolic Name: unknown
        Node WWN: 200000e08b074cb5
        Link Error Statistics:
                Link Failure Count: 0
                Loss of Sync Count: 0
                Loss of Signal Count: 0
                Primitive Seq Protocol Error Count: 0
                Invalid Tx Word Count: 0
                Invalid CRC Count: 0
Remote Port WWN: 210100e08b296060
        Active FC4 Types: SCSI
        SCSI Target: no
        Port Symbolic Name: unknown
        Node WWN: 200100e08b296060
        Link Error Statistics:
```

EXAMPLE 6 Listing All SCSI Targets and Link Statistics *(Continued)*

```
                Link Failure Count: 0
                Loss of Sync Count: 0
                Loss of Signal Count: 0
                Primitive Seq Protocol Error Count: 0
                Invalid Tx Word Count: 0
                Invalid CRC Count: 0
```

EXAMPLE 7 Listing SCSI Target Information

The following command lists all remote ports as well as the *scsi-target* information:

```
# fcinfo remote-port  -s -p 210100e08b272ab5

Remote Port WWN: 210100e08b296a60
        Active FC4 Types: SCSI,IP
        SCSI Target: no
        Port Symbolic Name: unknown
        Node WWN: 200100e08b296060
Remote Port WWN: 20030003ba27d56d
        Active FC4 Types: SCSI
        SCSI Target: yes
        Port Symbolic Name: unknown
        Node WWN: 10000003ba27d56d
        LUN: 0
          Vendor: SUN
          Product: T4
          OS Device Name: /dev/rdsk/c4t20030003BA27D56Dd0s2
        LUN: 1
          Vendor: SUN
          Product: T4
          OS Device Name: /dev/rdsk/c4t20030003BA27D56Dd1s2
Remote Port WWN: 210100e08b27f7a6
        Active FC4 Types: SCSI
        SCSI Target: no
        Port Symbolic Name: unknown
        Node WWN: 200100e08b27f7a6
Remote Port WWN: 50020f230000b897
        Active FC4 Types: SCSI
        SCSI Target: yes
        Port Symbolic Name: unknown
        Node WWN: 50020f200000b897
        LUN: 0
          Vendor: SUN
          Product: T300
          OS Device Name: Unknown
        LUN: 1
          Vendor: SUN
```

EXAMPLE 7 Listing SCSI Target Information *(Continued)*

```
    Product: T300
    OS Device Name: /dev/rdsk/c4t50020F230000B897d1s2
  LUN: 2
    Vendor: SUN
    Product: T300
    OS Device Name: /dev/rdsk/c4t50020F230000B897d2s2
  LUN: 3
    Vendor: SUN
    Product: T300
    OS Device Name: /dev/rdsk/c4t50020F230000B897d3s2
  LUN: 4
    Vendor: SUN
    Product: T300
    OS Device Name: /dev/rdsk/c4t50020F230000B897d4s2
  LUN: 5
    Vendor: SUN
    Product: T300
    OS Device Name: /dev/rdsk/c4t50020F230000B897d5s2
  LUN: 6
    Vendor: SUN
    Product: T300
    OS Device Name: /dev/rdsk/c4t50020F230000B897d6s2
  LUN: 7
    Vendor: SUN
    Product: T300
    OS Device Name: /dev/rdsk/c4t50020F230000B897d7s2
  LUN: 8
    Vendor: SUN
    Product: T300
    OS Device Name: /dev/rdsk/c4t50020F230000B897d8s2
  LUN: 9
    Vendor: SUN
    Product: T300
    OS Device Name: /dev/rdsk/c4t50020F230000B897d9s2
  LUN: 10
    Vendor: SUN
    Product: T300
    OS Device Name: /dev/rdsk/c4t50020F230000B897d10s2
  LUN: 11
    Vendor: SUN
    Product: T300
    OS Device Name: /dev/rdsk/c4t50020F230000B897d11s2
  LUN: 12
    Vendor: SUN
    Product: T300
    OS Device Name: /dev/rdsk/c4t50020F230000B897d12s2
```

EXAMPLE 7 Listing SCSI Target Information *(Continued)*

```
        LUN: 13
          Vendor: SUN
          Product: T300
          OS Device Name: /dev/rdsk/c4t50020F230000B897d13s2
        LUN: 14
          Vendor: SUN
          Product: T300
          OS Device Name: /dev/rdsk/c4t50020F230000B897d14s2
        LUN: 15
          Vendor: SUN
          Product: T300
          OS Device Name: /dev/rdsk/c4t50020F230000B897d15s2
Remote Port WWN: 210100e08b27daa6
        Active FC4 Types: SCSI
        SCSI Target: no
        Port Symbolic Name: unknown
        Node WWN: 200100e08b27daa6
Remote Port WWN: 210000e08b074cb5
        Active FC4 Types: SCSI,IP
        SCSI Target: no
        Port Symbolic Name: unknown
        Node WWN: 200000e08b074cb5
Remote Port WWN: 210100e08b296060
        Active FC4 Types: SCSI
        SCSI Target: no
        Port Symbolic Name: unknown
        Node WWN: 200100e08b296060
```

EXAMPLE 8 Listing the Logical Unit

The following command lists the logical unit:

```
# fcinfo logical-unit

/dev/rdsk/c0t600C0FF000000000036223AE73EB705d0s2
/dev/rdsk/c0t600C0FF00000000000362272539E5B03d0s2
/dev/rmt/0n
```

EXAMPLE 9 Displaying Additional Information for the Logical Unit

The following command displays additional information about the logical unit using the `-v` option for device `/dev/rmt/On`:

```
# fcinfo lu -v /dev/rmt/On

 OS Device Name: /dev/rmt/0n
        HBA Port WWN: 210000e07c030b91
                  Remote Port WWN: 21010003b7830a6
```

EXAMPLE 9 Displaying Additional Information for the Logical Unit *(Continued)*

```
                    LUN: 0
        Vendor: STK
        Product: 9940A
        Device Type: Tape device
```

The following command displays additional information about the logical unit using the `-v` option for device `/dev/rdsk/c0t600C0FF0000000000036223AE73EB705d0s2`

```
# fcinfo logical-unit -v \
     /dev/rdsk/c0t600C0FF0000000000036223AE73EB705d0s2

 OS Device Name: /dev/rdsk/c0t600C0FF0000000000036223AE73EB705d0s2
        HBA Port WWN: 210100e08b27a8a1
                Remote Port WWN: 256000c0ffc03622
                        LUN: 0
                Remote Port WWN: 216000c0ff803622
                        LUN: 0
        HBA Port WWN: 210000e08b07a8a1
                Remote Port WWN: 256000c0ffc03622
                        LUN: 0
                Remote Port WWN: 216000c0ff803622
                        LUN: 0
        Vendor: SUN
        Product: StorEdge 3510
        Device Type: Disk device
```

EXAMPLE 10 Adding an NIPV Port

The following command adds an NPIV port to the HBA with a port name:

`210000e08b170f1c`

The NPIV port has a port name of `2000000000000001` and a node name of `2100000000000001`.

```
# fcadm create-npiv-port  -p 2000000000000001 -n 2100000000000001 \
210000e08b170f1c

Created NPIV Port:
    HBA Port WWN: 2000000000000001
    Node WWN: 2100000000000001
    Physical HBA Port WWN: 210000e08b170f1c
```

EXAMPLE 11 Adding an NIPV Port with Random WWN

The following command adds an NPIV port to the HBA with a randomly assigned port name of `210000e08b170f1c`.

EXAMPLE 11 Adding an NIPV Port with Random WWN *(Continued)*

```
# fcadm create-npiv-port 210000e08b170f1c

Created NPIV Port:
    HBA Port WWN: 2038295824942801
    Node WWN: 2100392849248001
    Physical HBA Port WWN: 210000e08b170f1c
```

EXAMPLE 12 Deleting an NIPV Port

The following command deletes an NPIV port.

```
# fcadm delete-npiv-port -p 2000000000000001 -w 2100000000000001 \
210000e08b170f1c
```

EXAMPLE 13 Creating an FCoE Target Port

The following command creates an FCoE port associated with the MAC interface `nxge0`.

```
# fcadm create-fcoe-port -t nxge0
```

EXAMPLE 14 Deleting an FCoE Port

The following command deletes the FCoE port associated with the MAC interface `nxge0`.

```
# fcadm delete-fcoe-port -t nxge0
```

EXAMPLE 15 Listing Information for an FCoE Port

The following command lists information for FCoE ports.

```
# fcadm list-fcoe-ports
HBA Port WWN: 200000144fc2d508
        Port Type: Target
        MAC Name: nxge0
        MTU Size: 9194
        MAC Factory Address: 00144fc2d508
        MAC Current Address: 00144fc2d508
HBA Port WWN: 200000144fc2d509
        Port Type: Target
        MAC Name: nxge1
        MTU Size: 9194
        MAC Factory Address: 00144fc2d509
        MAC Current Address: 00144fc2d509
```

EXAMPLE 16 Reinitializing the Link of an FC Port

The following command forces the link connected with the port `200000144fc2d508` to reinitialize.

```
# fcadm force-lip 200000144fc2d508
```

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/storage/fc-utilities
Interface Stability	Committed

See Also attributes(5)

Error Messages Errors that can occur in addition to the errors normally associated with system administration commands:

- *HBA_port_WWN*: `not found`
- *Remote_port_WWN*: `not found`
- *HBA_port_WWN*: `NPIV not supported on this HBA`

Name fdetach – detach a name from a STREAMS-based file descriptor

Synopsis `fdetach` *path*

Description The `fdetach` command detaches a STREAMS-based file descriptor from a name in the file system. Use the *path* operand to specify the path name of the object in the file system name space, which was previously attached. See `fattach(3C)`.

The user must be the owner of the file or a user with the appropriate privileges. All subsequent operations on *path* will operate on the underlying file system entry and not on the STREAMS file. The permissions and status of the entry are restored to the state they were in before the STREAMS file was attached to the entry.

Operands The following operands are supported:

path Specifies the path name of the object in the file system name space, which was previously attached.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also `fattach(3C)`, `fdetach(3C)`, `attributes(5)`, `streamio(7I)`

STREAMS Programming Guide

Name fdisk – create or modify fixed disk partition table

Synopsis

```
fdisk [-o offset] [-s size] [-P fill_patt] [-S geom_file]
     [-w | -r | -d | -n | -I | -B | -t | -T | -g | -G | -R | -E]
     [--F fdisk_file] [ [-v] -W {fdisk_file | –}]
     [-h] [-b masterboot]
     [-A id : act : bhead : bsect : bcyl : ehead : esect :
         ecyl : rsect : numsect]
     [-D id : act : bhead: bsect : bcyl : ehead: esect :
         ecyl : rsect : numsect] rdevice
```

Description This command is used to do the following:

- Create and modify an `fdisk` partition table on x86 systems
- Create and modify an `fdisk` partition table on removable media on SPARC or x86 systems
- Install the master boot record that is put in the first sector of the fixed disk on x86 systems only

This table is used by the first-stage bootstrap (or firmware) to identify parts of the disk reserved for different operating systems, and to identify the partition containing the second-stage bootstrap (the *active* Solaris partition). The *rdevice* argument must be used to specify the raw device associated with the fixed disk, for example, `/dev/rdsk/c0t0d0p0`.

The program can operate in three different modes. The first is interactive mode. In interactive mode, the program displays the partition table as it exists on the disk, and then presents a menu allowing the user to modify the table. The menu, questions, warnings, and error messages are intended to be self-explanatory.

In interactive mode, if there is no partition table on the disk, the user is given the options of creating a default partitioning or specifying the initial table values. The default partitioning allocates the entire disk for the Solaris system and makes the Solaris system partition active. In either case, when the initial table is created, `fdisk` also writes out the first-stage bootstrap (x86 only) code along with the partition table. In this mode, (x86 only) when creating an entry for a non-EFI partition on a disk that is larger than 2 TB (terabytes), `fdisk` warns that the maximum size of the partition is 2 TB. Under these conditions percentages displayed by `fdisk` are based on 2 TB.

The second mode of operation is used for automated entry addition, entry deletion, or replacement of the entire `fdisk` table. This mode can add or delete an entry described on the command line. In this mode the entire `fdisk` table can be read in from a file replacing the original table. `fdisk` can also be used to create this file. There is a command line option that will cause `fdisk` to replace any `fdisk` table with the default of the whole disk for the Solaris system.

The third mode of operation is used for disk diagnostics. In this mode, a section of the disk can be filled with a user-specified pattern and mode sections of the disk can also be read or written.

Note – The third mode of operation is not currently supported for extended partitions

When `fdisk` creates a partition, the space is allocated in the `fdisk` partition table, but the allocated disk space is not initialized. `newfs(1M)` is required to create and write file system metadata to the new partition, and `format(1M)` is required to write the VTOC or EFI/GPT metadata.

Menu Options

The menu options for interactive mode given by the `fdisk` program are:

`Create a partition`
: This option allows the user to create a new partition. The maximum number of partitions is 4. The program will ask for the type of the partition (SOLARIS, MS-DOS, UNIX, or other). It will then ask for the size of the partition as a percentage of the disk. The user may also enter the letter `c` at this point, in which case the program will ask for the starting cylinder number and size of the partition in cylinders. If a `c` is not entered, the program will determine the starting cylinder number where the partition will fit. In either case, if the partition would overlap an existing partition or will not fit, a message is displayed and the program returns to the original menu.

`Change Active (Boot from) partition`
: This option allows the user to specify the partition where the first-stage bootstrap will look for the second-stage bootstrap, otherwise known as the *active* partition.

`Delete a partition`
: This option allows the user to delete a previously created partition. Note that this will destroy all data in that partition.

`Change between Solaris and Solaris2 Partition IDs`
: This option allows the user to switch between the current `fdisk` operating system partition identifier and the previous one. This does not affect any data in the disk partition and is provided for compatibility with older software.

`Edit/View extended partitions`
: This option provides the extended partition menu to the user. Use the extended partition menu to add and delete logical drives, change the sysid of the logical drives, and display logical drive information. To commit the changes made in the extended partition, you must return to the main menu using the extended partition submenu option `r`. There is also an option to display the list of options that the extended partition submenu supports. Given below is the list:

  `a` Add a logical drive.

  Use this submenu option to add a logical drive. There are three pieces of information that are required: The beginning cylinder, the size (in cylinders or in human readable form - KB, MB, or GB), and the partition ID. While specifying the partition ID, there is an option (`I`) that you can use to list the supported partitions.

  `d` Delete a logical drive.

  Use this submenu option to delete a logical drive. The only input required is the number of the logical drive that is to be deleted.

`h` Display the help menu.

This submenu option displays the supported operations in the extended partition submenu.

`i` Change the id of the logical drive.

Use this submenu option to change the system ID of the existing logical drives. A list of supported system IDs is displayed when you use the `I` option when in this submenu.

`p` Display the logical drive layout.

Displays the logical drive information to stdout. This output reflects any changes made during the current run of the `fdisk` program. The changes are not committed to the disk until return to the main menu (using the submenu `r`) and choose the option to commit the changes to the disk.

`r` Return to the main `fdisk` menu.

Exit the extended partition submenu and return to the main menu.

Note the dynamic nature of the numbering of extended partitions. For example, consider a Solaris system with the partitions `p1`, `p2`, `p3`, and `p4`. Following creation of an extended partition, the same system has a logical device node, `p5`, and successive nodes numbered consecutively up to a maximum of `p36`. If one logical drive is deleted, say, `p8`, then all nodes following `p8` (`p9` up to `p36`) move up one in the list of partitions, so that `p9` becomes `p8`, `p10` becomes `p9`, and so forth.

Use the following options to include your modifications to the partition table at this time or to cancel the session without modifying the table:

`Exit` This option writes the new version of the table created during this session with `fdisk` out to the fixed disk, and exits the program.

`Cancel` This option exits without modifying the partition table.

Options The following options apply to `fdisk`:

`-A` *id:act:bhead:bsect:bcyl:ehead:esect:ecyl:rsect:numsect*
Add a partition as described by the argument (see the `-F` option below for the format). Use of this option will zero out the VTOC on the Solaris partition if the `fdisk` table changes.

`-b` *master_boot*
Specify the file *master_boot* as the master boot program. The default master boot program is `/usr/lib/fs/ufs/mboot`.

`-B`
Default to one Solaris partition that uses the whole disk. On an x86 machine, if the disk is larger than 2 TB (terabytes), the default size of the Solaris partition will be limited to 2 TB.

`-d`
Turn on verbose *debug* mode. This will cause `fdisk` to print its state on stderr as it is used. The output from this option should not be used with `-F`.

`-D` *id:act:bhead:bsect:bcyl:ehead:esect:ecyl:rsect:numsect*
Delete a partition as described by the argument (see the `-F` option below for the format). Note that the argument must be an exact match or the entry will not be deleted! Use of this option will zero out the VTOC on the Solaris partition if the `fdisk` table changes.

`-E`
Create an EFI partition that uses the entire disk.

`-F` *fdisk_file*
Use fdisk file *fdisk_file* to initialize table. Use of this option will zero out the VTOC on the Solaris partition if the `fdisk` table changes.

The *fdisk_file* contains four specification lines for the primary partitions followed by specification lines for the logical drives. You must have four lines for the primary partitions if there is at least one logical drive. In this case, if the number of primary partitions to be configured is less than four, the remaining lines should be filled with zeros.

Each line is composed of entries that are position-dependent, are separated by whitespace or colons, and have the following format:

id act bhead bsect bcyl ehead esect ecyl rsect numsect

...where the entries have the following values:

id This is the type of partition and the correct numeric values may be found in `fdisk.h`.

act This is the active partition flag; `0` means not active and `128` means active. For logical drives, this flag will always be set to 0 even if specified as 128 by the user.

bhead This is the head where the partition starts. If this is set to `0`, `fdisk` will correctly fill this in from other information.

bsect This is the sector where the partition starts. If this is set to `0`, `fdisk` will correctly fill this in from other information.

bcyl This is the cylinder where the partition starts. If this is set to `0`, `fdisk` will correctly fill this in from other information.

ehead This is the head where the partition ends. If this is set to `0`, `fdisk` will correctly fill this in from other information.

esect This is the sector where the partition ends. If this is set to `0`, `fdisk` will correctly fill this in from other information.

ecyl This is the cylinder where the partition ends. If this is set to `0`, `fdisk` will correctly fill this in from other information.

rsect The relative sector from the beginning of the disk where the partition starts. This must be specified and can be used by `fdisk` to fill in other fields. For logical drives, you must make sure that there are at least 63 free sectors before the *rsect* specified for a logical drive.

numsect The size in sectors of this disk partition. This must be specified and can be used by `fdisk` to fill in other fields.

`-g`
Get the label geometry for disk and display on stdout (see the `-S` option for the format).

`-G`
Get the physical geometry for disk and display on stdout (see the `-S` option for the format).

`-h`
Issue verbose message; message will list all options and supply an explanation for each.

`-I`
Forgo device checks. This is used to generate a file image of what would go on a disk without using the device. Note that you must use `-S` with this option (see above).

`-n`
Don't update `fdisk` table unless explicitly specified by another option. If no other options are used, `-n` will only write the master boot record to the disk. In addition, note that `fdisk` will not come up in interactive mode if the `-n` option is specified.

`-o` *offset*
Block offset from start of disk. This option is used for `-P`, `-r`, and `-w`. Zero is assumed when this option is not used.

`-P` *fill_patt*
Fill disk with pattern *fill_patt*. *fill_patt* can be decimal or hex and is used as number for constant long word pattern. If *fill_patt* is #, then pattern is block # for each block. Pattern is put in each block as long words and fills each block (see `-o` and `-s`).

`-r`
Read from disk and write to stdout. See `-o` and `-s`, which specify the starting point and size of the operation.

`-R`
Treat disk as read-only. This is for testing purposes.

`-s` *size*
Number of blocks to perform operation on (see `-o`).

`-S` *geom_file*
Set the label geometry to the content of the *geom_file*. The *geom_file* contains one specification line. Each line is delimited by a new-line character (`\n`). If the first character of a line is an asterisk (*), the line is treated as a comment. Each line is composed of entries that are position-dependent, are separated by white space, and have the following format:

pcyl ncyl acyl bcyl nheads nsectors sectsiz

where the entries have the following values:

pcyl	This is the number of physical cylinders for the drive.
ncyl	This is the number of usable cylinders for the drive.
acyl	This is the number of alt cylinders for the drive.
bcyl	This is the number of offset cylinders for the drive (should be zero).
nheads	The number of heads for this drive.
nsectors	The number of sectors per track.
sectsiz	The size in bytes of a sector.

`-t`
Adjust incorrect slice table entries so that they will not cross partition table boundaries.

`-T`
Remove incorrect slice table entries that span partition table boundaries.

`-v`
Output the HBA (virtual) geometry dimensions. This option must be used in conjunction with the `-W` flag. This option will work for platforms which support virtual geometry. (x86 only)

`-w`
Write to disk and read from stdin. See `-o` and `-s`, which specify the starting point and size of the operation.

`-W –`
Output the disk table to `stdout`.

`-W` *fdisk_file*
Create an `fdisk` file *fdisk_file* from disk table. This can be used with the `-F` option above.

Files

`/dev/rdsk/c0t0d0p0`	Raw device associated with the fixed disk.
`/usr/lib/fs/ufs/mboot`	Default master boot program.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	x86 and SPARC
Availability	system/core-os

See Also uname(1), fmthard(1M), format(1M), newfs(1M), parted(1M), prtvtoc(1M), attributes(5)

Diagnostics Most messages will be self-explanatory. The following may appear immediately after starting the program:

`Fdisk: cannot open <device>`
This indicates that the device name argument is not valid.

`Fdisk: unable to get device parameters for device <device>`
This indicates a problem with the configuration of the fixed disk, or an error in the fixed disk driver.

`Fdisk: error reading partition table`
This indicates that some error occurred when trying initially to read the fixed disk. This could be a problem with the fixed disk controller or driver, or with the configuration of the fixed disk.

`Fdisk: error writing boot record`
This indicates that some error occurred when trying to write the new partition table out to the fixed disk. This could be a problem with the fixed disk controller, the disk itself, the driver, or the configuration of the fixed disk.

Name ff – list file names and statistics for a file system

Synopsis `ff [-F` *FSType*`] [-V] [`*generic_options*`] [-o` *specific_options*`]` *special*`...`

Description `ff` prints the pathnames and inode numbers of files in the file system which resides on the special device *special*. Other information about the files may be printed using options described below. Selection criteria may be used to instruct `ff` to only print information for certain files. If no selection criteria are specified, information for all files considered will be printed (the default); the `-i` option may be used to limit files to those whose inodes are specified.

Output is sorted in ascending inode number order. The default line produced by `ff` is:

path-name i-number

The maximum information the command will provide is:

path-name i-number size uid

Options

`-F` Specify the *FSType* on which to operate. The *FSType* should either be specified here or be determinable from `/etc/vfstab` by matching the *special* with an entry in the table, or by consulting `/etc/default/fs`.

`-V` Echo the complete command line, but do not execute the command. The command line is generated by using the options and arguments provided by the user and adding to them information derived from `/etc/vfstab`. This option may be used to verify and validate the command line.

generic_options Options that are supported by most FSType-specific modules of the command. The following options are available:

- `-I` Do not print the i-node number after each path name.
- `-l` Generate a supplementary list of all path names for multiply-linked files.
- `-p` *prefix* The specified *prefix* will be added to each generated path name. The default is '.' (dot).
- `-s` Print the file size, in bytes, after each path name.
- `-u` Print the owner's login name after each path name.
- `-a -`*n* Select if the file has been accessed in *n* days.
- `-m -`*n* Select if the file has been written or created in *n* days.
- `-c -`*n* Select if file's status has been changed in *n* days.
- `-n` *file* Select if the file has been modified more recently than the argument `file`.

`-i` *i-node-list* Generate names for only those i-nodes specified in *i-node-list*. *i-node-list* is a list of numbers separated by commas (with no intervening spaces).

`-o` Specify *FSType*-specific options in a comma separated (without spaces) list of suboptions and keyword-attribute pairs for interpretation by the *FSType*-specific module of the command.

Operands *special* A special device.

Usage See `largefile(5)` for the description of the behavior of `ff` when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

Files `/etc/default/fs` default local file system type. Default values can be set for the following flags in `/etc/default/fs`. For example: `LOCAL=ufs`

`LOCAL` The default partition for a command if no FSType is specified.

`/etc/vfstab` list of default parameters for each file system

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also `find(1)`, `ncheck(1M)`, `stat(2)`, `vfstab(4)`, `attributes(5)`, `largefile(5)` Manual pages for the *FSType*-specific modules of `ff`.

Notes This command may not be supported for all *FSTypes*.

The `-a`, `-m`, and `-c` flags examine the `st_atime`, `st_mtime`, and `st_ctime` fields of the `stat` structure respectively. (See `stat(2)`.)

Name ff_ufs – list file names and statistics for a ufs file system

Synopsis

```
ff -F ufs [generic_options] [-o a,m,s] special...
```

Description `ff` prints the pathnames and inode numbers of files in the file system which resides on the special device *special*.

See `ff(1M)` for information regarding the `ff` command. See `OPTIONS` for information regarding the ufs-specific options.

Options The following options are supported:

`-o` Specify `ufs` file system specific options. The following options available are:

`a` Print the '.' and '. .' directory entries.

`m` Print mode information. This option must be specified in conjunction with the `-i` *i-node-list* option (see `ff(1M)`).

`s` Print only special files and files with set-user-ID mode.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also `find(1)`, `ff(1M)`, `ncheck(1M)`, `attributes(5)`

Name fiocompress – file compression utility

Synopsis

```
/usr/sbin/fiocompress -c [-m] [-b block_size] input_file output_file
/usr/sbin/fiocompress -d input_file output_file
```

Description The `fiocompress` utility is a file compression tool that works together with the `dcfs(7FS)` file system to perform per-file compression. You can use `fiocompress` to decompress a compressed file or mark a compressed file as compressed, causing automatic decompression on read. The primary use of `fiocompress` is to compress files in the boot archive.

Note that this utility is not a Committed interface. See `attributes(5)`.

Options The following options are supported:

`-b` *block_size*
Specify a block size for compression. The default block size is 8192.

`-c`
Compress the specified file.

`-d`
Decompress the specified file.

`-m`
Mark the compressed file for automatic decompression on read. Can be used only in conjunction with `-c`.

Exit Status

`0`
The command completed successfully.

`–1`
The command exited due to an error.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os
Interface Stability	Private

See Also `boot(1M)`, `bootadm(1M)`, `dcfs(7FS)`, `ufs(7FS)`, `attributes(5)`

Notes This compression/decompression utility works only with files stored in a UFS file system.

There is no obvious way to determine whether a given file is compressed, other than copying the file and comparing the number of disk blocks of the copy against the original.

Name flowadm – administer bandwidth resource control for protocols, services, containers, and virtual machines

Synopsis

```
flowadm

flowadm show-flow [-P] [[-p] -o field[,...]] [-l link] [flow]

flowadm add-flow [-t] [-R root-dir] -l link -a attr=value[,...]
     [-p prop=value[,...]] flow
flowadm remove-flow [-t] [-R root-dir] {-l link | flow}

flowadm set-flowprop [-t] [-R root-dir] -p prop=value[,...] flow
flowadm reset-flowprop [-t] [-R root-dir] [-p prop[,...]] flow
flowadm show-flowprop [-P] [[-c] -o field[,...]] [-l link]
     [-p prop[,...]] [flow]

flowadm help [subcommand-name]
```

Description The `flowadm` command is used to create, modify, remove, and show networking bandwidth and associated resources for a type of traffic on a particular link.

The `flowadm` command allows users to manage networking bandwidth resources for a transport, service, or a subnet. The service is specified as a combination of transport and local port. The subnet is specified by its IP address and subnet mask. The command can be used on any type of data link, including physical links, virtual NICs, and link aggregations.

A flow is defined as a set of attributes based on Layer 3 and Layer 4 headers, which can be used to identify a protocol, service, or a virtual machine.

Inbound and outbound packet are matched to flows in a very fast and scalable way, so that limits can be enforced with minimal performance impact.

The `flowadm` command can be used to identify a flow without imposing any bandwidth resource control. This would result in better observability for the flow when used along with `flowstat(1M)`.

Flows can be created, modified, and removed in both global and non-global zones. A zone administrator can create a flow only in his zone, global or non-global. However, a flow created in the global zone can migrate to a non-global zone, as described in the following paragraph. An administrator can modify or remove a flow only from within the zone, global or non-global, in which the flow was created. From the global zone, one can view all flows on a system, within the global and any non-global zones. From a non-global zone, one can view only those flows in that zone.

After an administrator creates a flow in the global zone, the data link associated with that flow can be assigned to a non-global zone. In such a case, the associated flow is also assigned to the same non-global zone. When this non-global zone is halted, the data link and its associated flow return to the global zone.

Different zone names distinguish flows of the same name. For example, one can have three flows named `fastpak`, if each `fastpak` is in a different zone. For example, `zone1/fastpak`, `zone2/fastpak`, and `zone3/fastpak` are all valid zone names.

`flowadm` is implemented as a set of subcommands with corresponding options. Options are described in the context of each subcommand. If `flowadm` is invoked with no subcommand, then all of the flows configured on the system will be displayed. See `EXAMPLES` below for more information.

Sub-commands The following subcommands are supported:

`flowadm show-flow` [`-pP`] [`-o` *field*[,...]] [`-l` *link*] [*flow*]
Show flow configuration information (the default) or statistics, either for all flows, all flows on a link, or for the specified *flow*.

`-o` *field*[,...]
A case-insensitive, comma-separated list of output fields to display. The field name must be one of the fields listed below, or a special value `all`, to display all fields. For each flow found, the following fields can be displayed:

`flow`
The name of the flow.

`link`
The name of the link the flow is on.

`ipaddr`
IP address of the flow. This can be either local or remote depending on how the flow was defined.

`transport`
The name of the layer for protocol to be used.

`port`
Local port of service for flow.

`dsfield`
Differentiated services value for flow and mask used with `DSFIELD` value to state the bits of interest in the differentiated services field of the IP header.

`-p`, `--parseable`
Display using a stable machine-parseable format.

`-P`, `--persistent`
Display persistent flow property information.

`-l` *link*, `--link=`*link*|*flow*
Display information for all flows on the named link or information for the named flow.

`flowadm add-flow` [`-t`] [`-R` *root-dir*] `-l` *link* `-a` *attr=value*[,...] `-p` *prop=value*[,...] *flow*
Adds a flow to the system. The flow is identified by its flow attributes and properties.

As part of identifying a particular flow, its bandwidth resource can be limited.

`-t`, `--temporary`
The changes are temporary and will not persist across reboots. Persistence is the default.

`-R` *root-dir*, `--root-dir=`*root-dir*
Specifies an alternate root directory where `flowadm` should apply persistent creation.

`-l` *link*, `--link=`*link*
Specify the link to which the flow will be added.

`-a` *attr*`=`*value*[,...], `--attr=`*value*
A comma-separated list of attributes to be set to the specified values.

`-p` *prop*`=`*value*[,...], `--prop=`*value*[,...]
A comma-separated list of properties to be set to the specified values.

`flowadm remove-flow` [`-t`] [`-R` *root-dir*] `-l` {*link* | *flow*}
Remove an existing flow identified by its link or name.

`-t`, `--temporary`
The changes are temporary and will not persist across reboots. Persistence is the default.

`-R` *root-dir*, `--root-dir=`*root-dir*
Specifies an alternate root directory where `flowadm` should apply persistent removal.

`-l` *link* | *flow*, `--link=`*link* | *flow*
If a link is specified, remove all flows from that link. If a single flow is specified, remove only that flow.

`flowadm set-flowprop` [`-t`] [`-R` *root-dir*] `-p` *prop*`=`*value*[,...] *flow*
Set values of one or more properties on the flow specified by name. The complete list of properties can be retrieved using the `show-flow` subcommand.

`-t`, `--temporary`
The changes are temporary and will not persist across reboots. Persistence is the default.

`-R` *root-dir*, `--root-dir=`*root-dir*
Specifies an alternate root directory where `flowadm` should apply persistent setting of properties.

`-p` *prop*`=`*value*[,...], `--prop=`*value*[,...]
A comma-separated list of properties to be set to the specified values.

`flowadm reset-flowprop` [`-t`] [`-R` *root-dir*] `-p` [*prop*`=`*value*[,...]] *flow*
Resets one or more properties to their default values on the specified flow. If no properties are specified, all properties are reset. See the `show-flowprop` subcommand for a description of properties, which includes their default values.

`-t`, `--temporary`
Specifies that the resets are temporary. Temporary resets last until the next reboot.

`-R` *root-dir*, `--root-dir=`*root-dir*
Specifies an alternate root directory where `flowadm` should apply persistent setting of properties.

`-p` *prop*=*value*[,...], `--prop=`*value*[,...]
A comma-separated list of properties to be reset.

`flowadm show-flowprop` [`-cP`] [`-l` *link*] [`-p` *prop*[,...]] [*flow*]
Show the current or persistent values of one or more properties, either for all flows, flows on a specified link, or for the specified flow.

By default, current values are shown. If no properties are specified, all available flow properties are displayed. For each property, the following fields are displayed:

`FLOW`
The name of the flow.

`PROPERTY`
The name of the property.

`VALUE`
The current (or persistent) property value. The value is shown as `--` (double hyphen), if it is not set, and `?` (question mark), if the value is unknown. Persistent values that are not set or have been reset will be shown as `--` and will use the system `DEFAULT` value (if any).

`DEFAULT`
The default value of the property. If the property has no default value, `--` (double hyphen), is shown.

`POSSIBLE`
A comma-separated list of the values the property can have. If the values span a numeric range, the minimum and maximum values might be shown as shorthand. If the possible values are unknown or unbounded, `--` (double hyphen), is shown.

Flow properties are documented in the "Flow Properties" section, below.

`-c`, `--parseable`
Display using a stable machine-parseable format.

`-P`, `--persistent`
Display persistent flow property information.

`-p` *prop*[,...], `--prop=`*prop*[,...]
A comma-separated list of properties to show.

`flowadm help` [*subcommand-name*]
Displays all the supported `flowadm` subcommands or usage for the given subcommand. If you display help for a specific subcommand, the command syntax is displayed, along with an example. Using `flowadm help` without any argument displays all the subcommands.

Flow Attributes

The flow operand that identifies a flow in a `flowadm` command is a comma-separated list of one or more keyword, value pairs from the list below.

`local_ip`[*/prefix_len*]
: Identifies a network flow by the local IP address. *value* must be a IPv4 address in dotted-decimal notation or an IPv6 address in colon-separated notation. *prefix_len* is optional.

  If *prefix_len* is specified, it describes the netmask for a subnet address, following the same notation convention of `ifconfig(1M)` and `route(1M)` addresses. If unspecified, the given IP address will be considered as a host address for which the default prefix length for a IPv4 address is `/32` and for IPv6 is `/128`.

`remote_ip`[*/prefix_len*]
: Identifies a network flow by the remote IP address. The syntax is the same as `local_ip` attributes

`transport`={`tcp`|`udp`|`sctp`|`icmp`|`icmpv6`}
: Identifies a layer 4 protocol to be used. It is typically used in combination with `local_port` or `remote_port` to identify the local or remote service that needs special attention.

`local_port`
: Identifies a service specified by the local port.

`remote_port`
: Identifies a service specified by the remote port.

`dsfield`[`:`*dsfield_mask*]
: Identifies the 8-bit differentiated services field (as defined in RFC 2474).

  The optional *dsfield_mask* is used to state the bits of interest in the differentiated services field when comparing with the `dsfield` value. A `0` in a bit position indicates that the bit value needs to be ignored and a `1` indicates otherwise. The mask can range from `0x01` to `0xff`. If *dsfield_mask* is not specified, the default mask `0xff` is used. Both the `dsfield` value and mask must be in hexadecimal.

The following combinations of attributes are supported:

`local_ip=`*address*`[/`*prefixlen*`]`
`remote_ip=`*address*`[/`*prefixlen*`]`
`transport={tcp|udp|sctp|icmp|icmpv6}`
`transport={tcp|udp|sctp},local_port=`*port*
`transport={tcp|udp|sctp},remote_port=`*port*
`dsfield=`*val*`[:`*dsfield_mask*`]`

On a given link, the combinations above are mutually exclusive. All the flows on a given link must have the same combination and only the attribute values differentiate the flows. An attempt to create flows of different combinations will fail.

Restrictions

There are individual flow restrictions and flow restrictions per zone.

Individual Flow Restrictions

Restrictions on individual flows do not require knowledge of other flows that have been added to the link.

An attribute can be listed only once for each flow. For example, the following command is not valid:

```
# flowadm add-flow -l vnic1 -a local_port=80,local_port=8080 httpflow
```

`transport` and `local_port` or `transport` and `remote_port`:

TCP, UDP, or SCTP flows can be specified with a local port or with a remote port. An ICMP or ICMPv6 flow that specifies a port is not allowed.

If either `local_port` or `remote_port` is specifed, the transport must be either TCP, UDP or SCTP.

The following commands are valid:

```
# flowadm add-flow -l e1000g0 -a transport=udp udpflow
# flowadm add-flow -l e1000g0 -a transport=tcp,local_port=80 \
udp80flow
```

The following commands are not valid:

```
# flowadm add-flow -l e1000g0 -a remote_port=25 flow25
# flowadm add-flow -l e1000g0 -a transport=icmpv6,remote_port=16 \
flow16
```

Flow Restrictions Per Zone

Within a zone, no two flows can have the same name. After adding a flow with the link specified, the link will not be required for display, modification, or deletion of the flow.

Flow Properties

The following flow properties are supported. Note that the ability to set a given property to a given value depends on the driver and hardware.

`maxbw`
: Sets the full duplex bandwidth for the flow. The bandwidth is specified as an integer with one of the scale suffixes(`K`, `M`, or `G` for Kbps, Mbps, and Gbps). If no units are specified, the input value will be read as Mbps. The default is no bandwidth limit.

Examples

EXAMPLE 1 Displaying Flow Configuration

The following command invokes flowadm with no arguments, thereby displaying all flows in the system.

```
# flowadm
FLOW        LINK    IPADDR          PROTO  LPORT   RPORT   DSFLD
tcpflow     net0    --              tcp    --      --      --
```

EXAMPLE 1 Displaying Flow Configuration *(Continued)*

```
udpflow     net0    --              udp    --      --      --
```

EXAMPLE 2 Creating a Policy Around a Mission-Critical Port

The command below creates a policy around inbound HTTPS traffic on an HTTPS server so that HTTPS obtains dedicated NIC hardware and kernel TCP/IP resources. The name specified, `https-1`, can be used later to modify or delete the policy.

```
# flowadm add-flow -l bge0 -a transport=TCP,local_port=443 https-1
# flowadm show-flow -l bge0
FLOW         LINK         IP ADDR            PROTO  PORT  RPORT   DSFLD
https1       bge0         --                 tcp    443   --      --
```

EXAMPLE 3 Modifying an Existing Policy to Add Bandwidth Resource Control

The following command modifies the `https-1` policy from the preceding example. The command adds bandwidth control.

```
# flowadm set-flowprop -p maxbw=500M https-1
# flowadm show-flow https-1
FLOW         LINK         IP ADDR            PROTO  PORT  RPORT   DSFLD
https1       bge0         --                 tcp    443   --      --

# flowadm show-flowprop https-1
FLOW        PROPERTY    VALUE     DEFAULT      POSSIBLE
https-1     maxbw       500       --           --
```

EXAMPLE 4 Limiting the UDP Bandwidth Usage

The following command creates a policy for UDP protocol so that it cannot consume more than 100Mbps of available bandwidth. The flow is named `limit-udp-1`.

```
# flowadm add-flow -l bge0 -a transport=UDP -p maxbw=100M, \
limit-udp-1
```

EXAMPLE 5 Setting Policy, Making Use of dsfield Attribute

The following command sets a policy for EF PHB (DSCP value of 101110 from RFC 2598) with a bandwidth of 500 Mbps. The `dsfield` value for this flow will be `0x2e` (101110) with the `dsfield_mask` being `0xfc` (because we want to ignore the 2 least significant bits).

```
# flowadm add-flow -l bge0 -a dsfield=0x2e:0xfc \
-p maxbw=500M efphb-flow
```

EXAMPLE 6 Viewing Flows in Multiple Zones

The following command shows two flows of the same name. The first flow is in the global zone, the second is in the zone `zone1`. The command is invoked from the global zone, enabling you to view all flows on the system

EXAMPLE 6 Viewing Flows in Multiple Zones *(Continued)*

```
# flowadm show-flow
FLOW        LINK        IPADDR            PROTO  LPORT   RPORT   DSFLD
tcpflow     e1000g2     --                tcp    --      --      --
zone1/tcpflow e1000g1   --                tcp    --      --      --
```

EXAMPLE 7 Combining Invalid Flows

All the flows on a given link must have the same combination of attributes. Consider the following sequence:

```
# flowadm add-flow -l e1000g0 -a transport=tcp,local_port=443 httpsflow
# flowadm add-flow -l e1000g0 -a local_ip=192.1.168.157 ngzflow
```

The second command will fail because the first flow uses the combination:

```
transport={tcp|udp|sctp},local_port=port
```

...which is incompatible with the combination used in the second flow:

```
local_ip[/prefixlen]=address
```

EXAMPLE 8 Display Help

The following command lists the `flowadm` subcommands.

```
# flowadm help
The following subcommands are supported:
Flow subcommands : add-flow, remove-flow, reset-flowprop,
set-flowprop, show-flow, show-flowprop

For more info, run: flowadm help subcommand
```

The following command illustrates the use of `flowadm help` with a specific subcommand.

```
# flowadm help add-flow
        usage:
add-flow    [-t] [-R root-dir] -l link -a attr=value[,...]
     [-p prop=value,...] flow

example:
# flowadm add-flow -l net0 -a transport=tcp -p maxbw=100 tcpflow
```

Exit Status

0

All actions were performed successfully.

>0

An error occurred.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/network
Interface Stability	Committed

See Also acctadm(1M), dladm(1M), flowstat(1M), ifconfig(1M), prstat(1M), route(1M), attributes(5), ifconfig(1M)

Notes The show-usage subcommand, present in previous releases of flowadm, has been replaced by the flowstat(1M) -h command.

Name flowstat – report flow statistics

Synopsis

```
flowstat [-r | -t] [-i interval] [-l link] [flow]

flowstat [-S] [-A] [-i interval] [-p] [ -o field[,...]]
     [-u R|K|M|G|T|P] [link] [-l link] [flow]

flowstat -h [-a] -f filename [-d] [-F format] [-s time]
     [-e time] [flow]
```

Description The `flowstat` command reports run time statistics about user defined flows. `flowadm show-flow` provides the flow name information for this command.

Options The `flowstat` command has the following options and operands that are common among a number of command forms shown under "Subcommands," below.

Sub-commands `flowstat` supports the following command forms.

`flowstat` [`-r` | `-t`] [`-i` *interval*] [`-l` *link*] [*flow*]

This form of the command iteratively examines all flows and reports statistics. The output is sorted in descending order of flow utilization. If no flow is specified, the system displays statistics for all flows.

`-r`

Display receive-side statistics only. Includes bytes and packets received, drops, and so forth. See examples for complete listing.

`-t`

Display transmit-side statistics only. Includes bytes and packets sent, drops, and so forth. See examples below.

`-i` *interval*

Specify an interval in seconds at which statistics are refreshed. The default interval is one second.

`-l` *link* | *flow*]

Display statistics for all flows on the specified link or statistics for the specified flow.

`flowstat` [`-S`] [`-A`] [`-i` *interval*] [`-p`] [`-o` *field*[,...]] [`-u` R|K|M|G|T|P] [`-l` *link*] [*flow*]

This form of the command allows you to specify which statistics to display.

`-A`

Dump all statistics fields for this flow. Output statistics of this command are inclusive of all the statistics reported by all other `flowstat` commands.

`-i` *interval*

Specify an interval in seconds at which statistics are refreshed. The default interval is one second.

{`-l` *link*} | *flow*

Display statistics for all flows on the specified link or for the specified flow.

`-o` *field*[,...]
Display a case-insensitive, comma-separated list of output fields to display. The field name must be one of the fields listed below, or the special value `all` to display all supported fields.

List of supported RX fields:

- `flow`
- `rbytes`
- `ipkts`
- `idrops`

List of TX fields:

- `flow`
- `obytes`
- `opkts`
- `odrops`

`-p`
Display output in a stable, machine-parseable format.

`-S`
Continuously display network utilization by flow in a manner similar to the way that `prstat(1M)` displays CPU utilization by process.

`-u` `R`|`K`|`M`|`G`|`T`|`P`
If used, allows choosing the unit in which to display all statistics, for example, `R`: Raw Numbers, `K`:Kilobits, `M`:Megabits, `T`:Terabits, `P`:Petabits. If not used, then different units, as appropriate, are used to display the statistics.

`flowstat -h` [`-a`] `-f` *filename* [`-d`] [`-F` *format*] [`-s` *time*] [`-e` *time*] [*flow*]
Show the network usage history from a stored extended accounting file. Use of this syntax requires that net accounting has been previously configured and enabled by using `acctadm(1M)`. The default output is the summary of network usage of the existing links for the entire period when extended accounting was enabled.

`-a`
Display all historical network usage for the specified period when extended accounting is enabled. This includes usage information for the flows that have already been deleted.

`-f` *filename*
Specify the file from which extended accounting records of network flow usage history are read.

`-d`
Display the dates for which there is logging information. The date is in the format *mm*/*dd*/*yyyy*.

`-F` *format*
Specify the output format of the network flow usage history information. `gnuplot` is the only supported format.

`-s` *time*
`-e` *time*
Specify start and stop times for data display. Time is in the format *MM*/*DD*/*YYYY*,*hh*:*mm*:*ss*. *hh* uses 24-hour clock notation.

Operands `flowstat` command forms have a single, optional operand.

flow
If specified, report only on the named flow. Otherwise, report on all flows. A flow has a name of the form *zonename*/*flowname*. A *flowname* without a *zonename* modifier is understood to be in the global zone.

Examples **EXAMPLE 1** Displaying Statistics

To display statistics for all the flows, enter following command. Statistics are displayed as 3-digit numbers with the appropriate unit. Default interval is one second.

```
# flowstat -i 1
 FLOW   IPKTS  RBYTES  IDROPS   OPKTS  OBYTES  ODROPS
flow1 528.54K 787.39M       0 179.39K  11.85M       0
flow2 742.81K   1.10G       0       0       0       0
flow3       0       0       0       0       0       0
flow1  67.73K 101.02M       0  21.04K   1.39M       0
flow2       0       0       0       0       0       0
flow3       0       0       0       0       0       0
            .       .       .       .       .       .
            .       .       .       .       .       .
            .       .       .       .       .       .
```

EXAMPLE 2 Displaying RX-Side Statistics

The following command displays receive-side statistics

```
# flowstat -r
 FLOW   IPKTS  RBYTES  IDROPS
flow1   4.01M   5.98G       0
flow2 742.81K   1.10G       0
flow3       0       0       0
```

EXAMPLE 3 Displaying TX-Side Statistics

The following command displays transmit-side statistics at a five-second interval.

```
# flowstat -t
 FLOW   OPKTS  OBYTES  ODROPS
flow1  24.37M   1.61G       0
flow2       0       0       0
```

EXAMPLE 3 Displaying TX-Side Statistics *(Continued)*

```
flow3       4     216       0
```

EXAMPLE 4 Displaying Particular Set of Statistics

The following command displays a specified set of statistics fields.

```
# flowstat -o FLOW,IPKTS
 FLOW   IPKTS
flow1  68.58M
flow2 742.81K
flow3       4
```

EXAMPLE 5 Show Historical Network Usage

Flow usage statistics can be stored by using the extended accounting facility, `acctadm(1M)`.

```
# acctadm -e extended -f /var/log/net.log net
# acctadm net
        Network accounting: active
    Network accounting file: /var/log/net.log
  Tracked Network resources: extended
Untracked Network resources: none
```

The saved historical data can be retrieved as follows:

```
# flowstat -h -f /var/log/net.log
LINK      DURATION  IPACKETS RBYTES      OPACKETS OBYTES      BANDWIDTH
flowtcp   100       1031     546908      0        0            43.76Kbps
flowudp   0         0        0           0        0             0.00Mbps
```

Display logging information for `flowtcp` starting at February 19, 2008 at 10:38:46 and ending on the same day at 10:40:06:

```
# flowstat -h -s 02/19/2008,10:39:06 -e 02/19/2008,10:40:06 \
-f /var/log/net.log flowtcp
FLOW      START       END         RBYTES   OBYTES     BANDWIDTH
flowtcp   10:39:06    10:39:26    1546     6539        3.23 Kbps
flowtcp   10:39:26    10:39:46    3586     9922        5.40 Kbps
flowtcp   10:39:46    10:40:06    240      216       182.40 bps
flowtcp   10:40:06    10:40:26    0        0           0.00 bps
```

Generate the same output information as above as a plotfile:

```
# flowstat -h -s 02/19/2008,10:39:06 -e 02/19/2008,10:40:06 \
-F gnuplot -f /var/log/net.log flowtcp
# Time tcp-flow
10:39:06 3.23
10:39:26 5.40
10:39:46 0.18
```

EXAMPLE 5 Show Historical Network Usage *(Continued)*

```
10:40:06 0.00
```

Exit Status

`0`
All actions were performed successfully.

`>0`
An error occurred.

Attributes

See `attributes(5)` for descriptions of the following attributes:

`/usr/sbin`

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os
Interface Stability	See below

Screen output is Uncommitted. The invocation is Committed.

See Also

`acctadm(1M)`, `dladm(1M)`, `dlstat(1M)`, `flowadm(1M)`, `ifconfig(1M)`, `prstat(1M)`, `route(1M)`, `attributes(5)`, `dlpi(7P)`

Name fmadm – fault management configuration tool

Synopsis `fmadm [-q] [`*subcommand* `[`*arguments*`]]`

Description The `fmadm` utility can be used by administrators and service personnel to view and modify system configuration parameters maintained by the Solaris Fault Manager, `fmd`(1M). `fmd` receives telemetry information relating to problems detected by the system software, diagnoses these problems, and initiates proactive self-healing activities such as disabling faulty components.

`fmadm` can be used to:

- view the set of diagnosis engines and agents that are currently participating in fault management,
- view the list of system components that have been diagnosed as faulty, and
- perform administrative tasks related to these entities.

The Fault Manager attempts to automate as many activities as possible, so use of `fmadm` is typically not required. When the Fault Manager needs help from a human administrator, service repair technician, or Oracle, it produces a message indicating its needs. It also refers you to a knowledge article on Sun's web site. The web site might ask you to use `fmadm` or one of the other fault management utilities to gather more information or perform additional tasks. The documentation for `fmd`(1M), `fmdump`(1M), and `fmstat`(1M) describe more about tools to observe fault management activities.

One responsibility of the Fault Manager is to keep track of the location of components. At the chassis level, the `fmadm *-alias` subcommands manages a chassis *chassis-name.chassis-serial* to *alias-id* mapping. The administered *alias-id* is intended to describe, in some meaningful way, the physical location of a chassis.

The `fmadm` utility requires the user to possess the `SYS_CONFIG` privilege. Refer to the *Oracle Solaris 11.1 Administration: Security Services* for more information about how to configure Solaris privileges. The `fmadm load` subcommand requires that the user possess all privileges.

SUBCOMMANDS `fmadm` accepts the following subcommands. Some of the subcommands accept or require additional options and operands. The `load`, `unload`, `reset`, and `rotate` subcommands are intended for trained technical personnel. We recommend against use of these subcommands without the specific guidance of, for example, a Knowledge Base article.

`fmadm acquit` *fmri* | *label* [*uuid*]

Notify the Fault Manager that the specified resource is not to be considered to be a suspect in the fault event identified by *uuid*, or if no UUID is specified, then in any fault or faults that have been detected. The `fmadm acquit` subcommand should be used only at the direction of a documented Sun repair procedure. Administrators might need to apply additional commands to re-enable a previously faulted resource.

`fmadm acquit` *uuid*

Notify the Fault Manager that the fault event identified by *uuid* can be safely ignored. The `fmadm acquit` subcommand should be used only at the direction of a documented Sun repair procedure. Administrators might need to apply additional commands to re-enable any previously faulted resources.

`fmadm config`

Display the configuration of the Fault Manager itself, including the module name, version, and description of each component module. Fault Manager modules provide services such as automated diagnosis, self-healing, and messaging for hardware and software present on the system.

`fmadm faulty` [`-afgiprsv`] [`-n` *max*] [`-u` *uid*]

Display status information for resources that the Fault Manager currently believes to be faulty.

The following options are supported:

`-a` Display all faults. By default, the `fmadm faulty` command only lists output for resources that are currently present and faulty. If you specify the `-a` option, all resource information cached by the Fault Manager is listed, including faults which have been automatically corrected or where no recovery action is needed. The listing includes information for resources that might no longer be present in the system.

`-f` Display faulty `fru's` (Field replaceable units).

`-g` Group together faults which have the same fru, class and fault message.

`-i` Display persistent cache identifier for each resource in the Fault Manager.

`-n` *max* If faults or resources are grouped together with the `-a` or `-g` options, limit the output to *max* entries.

`-p` Pipe output through pager with form feed between each fault.

`-r` Display Fault Management Resource with their Identifier (FMRI) and their fault management state.

`-s` Display 1 line fault summary for each fault event.

`-u` *uid* Only display fault with given uid.

`-v` Display full output.

The percentage certainty is displayed if a fault has multiple suspects, either of different classes or on different `fru`'s. If more than one resource is on the same `fru` and it is not 100% certain that the fault is associated with the `fru`, the maximum percentage certainty of the possible suspects on the `fru` is displayed.

The Fault Manager associates the following states with every resource for which telemetry information has been received:

`ok`
The resource is present and in use and has no known problems so far as the Fault Manager is concerned.

`unknown`
The resource is not present or not usable but has no known problems. This might indicate the resource has been disabled or deconfigured by an administrator. Consult appropriate management tools for more information.

`faulted`
The resource is present but is not usable because one or more problems have been diagnosed by the Fault Manager. The resource has been disabled to prevent further damage to the system.

`degraded`
The resource is present and usable, but one or more problems have been diagnosed in the resource by the Fault Manager.

If all affected resources are in the same state, this is reflected in the message at the end of the list. Otherwise the state is given after each affected resource.

`fmadm flush` *fmri*
Flush the information cached by the Fault Manager for the specified resource, named by its FMRI. This subcommand should only be used when indicated by a documented Sun repair procedure. Typically, the use of this command is not necessary as the Fault Manager keeps its cache up-to-date automatically. If a faulty resource is flushed from the cache, administrators might need to apply additional commands to enable the specified resource.

`fmadm load` *path*
Load the specified Fault Manager module. *path* must be an absolute path and must refer to a module present in one of the defined directories for modules. Typically, the use of this command is not necessary as the Fault Manager loads modules automatically when Solaris initially boots or as needed.

`fmadm unload` *module*
Unload the specified Fault Manager module. Specify *module* using the basename listed in the `fmadm config` output. Typically, the use of this command is not necessary as the Fault Manager loads and unloads modules automatically based on the system configuration

`fmadm repaired` *fmri* | *label*
Notify the Fault Manager that a repair procedure has been carried out on the specified resource. The `fmadm repaired` subcommand should be used only at the direction of a documented Sun repair procedure. Administrators might need to apply additional commands to re-enable a previously faulted resource.

`fmadm replaced` *fmri* | *label*
Notify the Fault Manager that the specified resource has been replaced. This command should be used in those cases where the Fault Manager is unable to automatically detect the replacement. The `fmadm replaced` subcommand should be used only at the direction of a documented Sun repair procedure. Administrators might need to apply additional commands to re-enable a previously faulted resource.

`fmadm reset` [`-s` *serd*] *module*
Reset the specified Fault Manager module or module subcomponent. If the `-s` option is present, the specified Soft Error Rate Discrimination (SERD) engine is reset within the module. If the `-s` option is not present, the entire module is reset and all persistent state associated with the module is deleted. The `fmadm reset` subcommand should only be used at the direction of a documented Sun repair procedure. The use of this command is typically not necessary as the Fault Manager manages its modules automatically.

`fmadm rotate errlog | fltlog | infolog | infolog_hival`
The `rotate` subcommand is a helper command for `logadm(1M)`, so that `logadm` can rotate live log files correctly. It is not intended to be invoked directly (and invoking it directly is likely to lose log history). Use one of the following commands to cause the appropriate logfile to be rotated, if the current one is not zero in size:

```
# logadm -p now -s 1b /var/fm/fmd/errlog
# logadm -p now -s 1b /var/fm/fmd/fltlog
# logadm -p now -s 1b /var/fm/fmd/infolog
# logadm -p now -s 1b /var/fm/fmd/infolog_hival
```

`fmadm add-alias` *chassis-name.chassis-serial alias-id* ['*comment*']
The `add-alias` subcommand is used to establish *alias-id* as a managed alias for the *chassis-name.chassis-serial* chassis. When a managed alias is defined, the `/dev/chassis` `devchassis(7FS)` name space representation of the chassis will use the more meaningful *alias-id* instead of the *chassis-name.chassis-serial*.

```
# fmadm add-alias SUN-Storage-J4410.1039QAQ007 RACK29.U25-28
```

The command shown above will verify that the new mapping does not conflict with existing mappings. In the case of conflict, no mapping change occurs. This subcommand completes when the associated name space updates are complete. If the updated name space does not use the new *alias-id*, a warning if printed, but the mapping is updated. If the name space update takes too long, a warning is printed.

If an optional comment is provided, the comment is preserved and will be displayed by a subsequent `lookup-alias` or `list-alias` command.

`fmadm remove-alias` *alias-id* | *chassis-name.chassis-serial*
The `remove-alias` subcommand is used to remove an *chassis-name.chassis-serial* to *alias-id* mapping.

```
# fmadm remove-alias RACK29.U25-28
```

The subcommand above completes when the associated name space updates are complete.

`fmadm lookup-alias` *alias-id* | *chassis-name.chassis-serial*
The `lookup-alias` subcommand can be used to determine what the current mapping is. The following is an example command.

```
# fmadm lookup-alias SUN-Storage-J4410.1039QAQ007
```

`fmadm list-alias`
The `list-alias` subcommand is used to display all comments and mappings.

`fmadm sync-alias`
The `sync-alias` subcommand is used to hand-import a set of mappings in bulk. Two copies of the current mappings are maintained:

- `/etc/dev/chassis_aliases`
- `/etc/dev/.chassis_aliase`

To import a set of mappings in bulk, you can update the `/etc/dev/chassis_aliases` file and then run `sync-alias`.

Options The following options are supported:

`-q` Set quiet mode. `fmadm` does not produce messages indicating the result of successful operations to standard output.

Operands The following operands are supported:

cmd The name of a subcommand listed in `SUBCOMMANDS`.

args One or more options or arguments appropriate for the selected *subcommand*, as described in `SUBCOMMANDS`. Among these arguments are `fmri`, `uuid`, and `label`. These identify resources that are the objects of `fmadm` subcommands. Use `fmadm faulty` to obtain the `fmri`, `uuid`, and `label` for a targeted resource. See `EXAMPLES`. In general, `label` is the most user-friendly of these operands.

Examples **EXAMPLE 1** Invoking `faulty` Subcommand

The following command invokes the `faulty` subcommand, which displays the `uuid`, `label`, and `fmri` for a component.

```
# fmadm faulty
------------ ------------------------------------  ------------ ---------
TIME         EVENT-ID                              MSG-ID       SEVERITY
------------ ------------------------------------  ------------ ---------
Sep 09 16:15 96609fae-113c-e48c-b1cf-ebf4b0902d72  DISK-8000-3E Critical

Problem Status  : solved [injected]
Diag Engine     : eft / 1.16
System
   Manufacturer : Oracle-Corp.
   Name         : SUN-FIRE-X4170-SERVER
   Part Number  : unknown
```

EXAMPLE 1 Invoking `faulty` Subcommand *(Continued)*

```
   Serial Number: 0920XF508B

----------------------------------------
Suspect 1 of 1:
  Fault class: fault.io.scsi.cmd.disk.dev.rqs.derr
  Certainty  : 100%
  Affects    : dev:///:devid=id1,sd@n5000c5000940edbb//scsi_vhci/disk@g\
               5000c5000940edbb
  Status     : out of service, but associated components no longer faulty

   FRU
      Status            : replaced
      Location          : DISK 11
      Manufacturer      : SEAGATE
      Name              : SEAGATE-ST330057SSUN300G
      Part Number       : SEAGATE-ST330057SSUN300G
      Revision          : 0205
      Serial Number     : 000930G01CN4----3SJ01CN4
      Chassis
         Manufacturer   : Oracle-Corp.
         Name           : SUN-Storage-J4410
         Part Number    : 594-5329
         Serial Number  : 1037QAQ052
      ...
      ...
```

In the preceding output, the `uuid` is the first item in the `EVENT-ID` column, `96609fae-113c-e48c-b1cf-ebf4b0902d72`. The `label` is in the `FRU` section in the Location line, `DISK 11`.

The `fmris` are available with `fmdump -v`:

```
# fmdump -v
Sep 09 16:15:36.9252 96609fae-113c-e48c-b1cf-ebf4b0902d72 DISK-8000-3E \
Diagnosed 100%  fault.io.scsi.cmd.disk.dev.rqs.derr

Problem in: hc://:scheme=:chassis-mfg=Oracle-Corp.:chassis-name=SUN-\
Storage-J4410:chassis-part=594-5329:chassis-serial=1037QAQ052/ses-\
enclosure=0/bay=11/disk=0

Affects: dev:///:devid=id1,sd@n5000c5000940edbb//\
scsi_vhci/disk@g5000c5000940edbb
FRU: hc://chassis-mfg=Oracle-Corp.:chassis-name=SUN-Storage-J4410\
:chassis-part=594-5329:chassis-serial=1037QAQ052:fru-mfg=SEAGATE\
:fru-name=SEAGATE-ST330057SSUN300G:fru-part=SEAGATE-ST330057SSUN300G\
:fru-revision=0205:fru-serial=000930G01CN4--------3SJ01CN4/\
```

EXAMPLE 1 Invoking `faulty` Subcommand *(Continued)*

```
ses-enclosure=0/bay=11/disk=0

Location: DISK 11
```

Note that `label` is the easiest-to-use identifier.

EXAMPLE 2 Obtaining Module Name

The following command displays the module name for each component. The module name is specified as input to the `fmadm unload` command.

```
# fmadm config
MODULE                   VERSION STATUS  DESCRIPTION
cpumem-retire            1.1     active  CPU/Memory Retire Agent
disk-transport           1.0     active  Disk Transport Agent
eft                      1.16    active  eft diagnosis engine
..
```

Exit Status

The following exit values are returned:

`0` Successful completion.

`1` An error occurred. Errors include a failure to communicate with `fmd` or insufficient privileges to perform the requested operation.

`2` Invalid command-line options were specified.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/fault-management
Interface Stability	See below.

The command-line options are Committed. The human-readable output is not-an-interface.

See Also

`fmd(1M)`, `fmdump(1M)`, `fmstat(1M)`, `logadm(1M)`, `syslogd(1M)`, `attributes(5)`, `devchassis(7FS)`

Oracle Solaris Administration: Common Tasks

Name fmd – fault manager daemon

Synopsis `/usr/lib/fm/fmd/fmd [-V] [-f` *file*`] [-o` *opt*`=`*val*`] [-R` *dir*`]`

Description `fmd` is a daemon that runs in the background on each Solaris system. `fmd` receives telemetry information relating to problems detected by the system software, diagnoses these problems, and initiates proactive self-healing activities such as disabling faulty components. When appropriate, the fault manager also sends a message to the `syslogd(1M)` service to notify an administrator that a problem has been detected. The message directs administrators to a knowledge article on Oracle's web site, `https://support.oracle.com`, which explains more about the problem impact and appropriate responses.

Each problem diagnosed by the fault manager is assigned a Universal Unique Identifier (UUID). The UUID uniquely identifes this particular problem across any set of systems. The `fmdump(1M)` utility can be used to view the list of problems diagnosed by the fault manager, along with their UUIDs and knowledge article message identifiers. The `fmadm(1M)` utility can be used to view the resources on the system believed to be faulty. The `fmstat(1M)` utility can be used to report statistics kept by the fault manager. The fault manager is started automatically when Solaris boots, so it is not necessary to use the `fmd` command directly. Sun's web site explains more about what capabilities are currently available for the fault manager on Solaris.

Notification Services

`syslog` (package `system/fault-management`)

The standard notification mechanism for new diagnoses is by means of `syslog`, using the `syslog-msgs` `fmd` module delivered in the same package as `fmd` itself.

By default, only new problem diagnoses are messaged by means of `syslog-msgs`, using the `syslog` facility and severity as listed in the table below. An administrator can use `svccfg(1M)` to request that other events in the problem resolution lifecycle are messaged through `syslog-msgs`:

```
# svccfg setnotify event syslog:{active|inactive}
```

See `svccfg(1M)` for additional detail.

Event	Disposition	Facility	Severity
problem-diagnosed	active	LOG_DAEMON	LOG_ERR
problem-updated	inactive	LOG_DAEMON	LOG_NOTICE
problem-repaired	inactive	LOG_DAEMON	LOG_NOTICE
problem-resolved	inactive	LOG_DAEMON	LOG_NOTICE

Email (package `system/fault-management/smtp-notify`)

Notification by means of email is an option for which an additional package must be installed. The SMF service, `svc:/system/fm/smtp-notify:default`, is delivered by means of the package `system/fault-management/smtp-notify` and notification preferences configured by means of `svccfg(1M)`. See `smtp-notify(1M)` for additional detail. Note that in addition to configuring notification preferences for the problem lifecycle events listed

above (problem-diagnosed, and so forth) this mechanism can also be configured through `svccfg(1M)` to provide notification of SMF instance state transition and other events.

SNMP (package `system/fault-management/snmp-notify`)
Notification of new events using SNMP traps is an option delivered by the package `system/fault-management/snmp-notify`. The service `svc:/system/fm/snmp-notify:default` is responsible for raising SNMP traps for problem lifecycle and other designated events (including SMF instance state transition events, if so configured). See `snmp-notify(1M)` for additional detail.

Global and Non-Global Solaris Zones

The fault manager service `svc:/system/fmd:default` service is configured in both global and non-global Solaris zones. In non-global zones, various hardware-oriented fault manager modules are not delivered, so it is a cut-down fault manager that runs there. In a non-global zone, the fault manager is focussed on software events.

Options

The following options are supported

`-f` *file*
Read the specified configuration *file* prior to searching for any of the default fault manager configuration files.

`-o` *opt=value*
Set the specified fault manager option to the specified value. Fault manager options are currently a Private interface; see `attributes(5)` for information about Private interfaces.

`-R` *dir*
Use the specified root directory for all pathnames evaluated by the fault manager, instead of the default root (`/`).

`-V`
Print the fault manager's version to stdout and exit.

Exit Status

The following exit values are returned:

`0` Successful completion

`1` An error occurred which prevented the fault manager from initializing, such as failure to open the telemetry transport.

`2` Invalid command-line options were specified.

Files

`/etc/fm/fmd`	Fault manager configuration directory
`/usr/lib/fm/fmd`	Fault manager library directory
`/var/fm/fmd`	Fault manager log directory

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/fault-management
Interface Stability	Committed

See Also svcs(1), fmadm(1M), fmdump(1M), fmstat(1M), smtp-notify(1M), snmp-notify(1M), svccfg(1M), syslogd(1M), attributes(5), smf(5)

http://www.support.oracle.com/msg/

Notes The Fault Manager is managed by the service management facility, smf(5), under the service identifier:

```
svc:/system/fmd:default
```

The service's status can be queried using the svcs(1) command. Administrators should not disable the Fault Manager service.

Name fmdump – fault management log viewer

Synopsis

```
fmdump [[-e | -i | -I] | -A ] [-f] [-mvVp] [-c class] [-R root]
     [-t time] [-T time] [-u uid] [-n name[.name]*[=value]]
     [-E ENA] [file] ...
```

Description The `fmdump` utility can be used to display the contents of any of the log files associated with the Solaris Fault Manager, `fmd(1M)`. The Fault Manager runs in the background on each Solaris system. It receives telemetry information relating to problems detected by the system software, diagnoses these problems, and initiates proactive self-healing activities such as disabling faulty components.

The `fmdump` utility is not intended as the primary administrative interface to the Fault Manager. For that purpose, use `fmadm` (and see `fmadm(1M)`). The `fmdump` utility simply dumps Fault Manager historical logs with little further interpretation, and can include implementation detail without explanation. See the `ATTRIBUTES` section below.

The Fault Manager maintains several sets of log files for use by service personnel and, to a lesser extent, administrators:

error log — A log that records error telemetry, the symptoms of problems detected by the system.

info log — A log that records informational events. This is realized as two sets of log files: high-value informational events, and other informational events.

fault log — A log that records fault diagnosis information, the problems believed to explain the symptoms recorded in the error and info logs.

A log file set consists of the current active log file together with a possible number of older rotated log files in that set. All logs are managed with `logadm(1M)` and have entries in `/etc/logadm.conf`.

Note that the `fmdump` utility dumps the current log file and all rotated log files for the target set. It therefore displays the entire log history. For the fault log, in particular, it is important to recognize that `fmdump` will show all problems *ever* diagnosed and is not limited to still-current problems. (Use `fmadm` faulty for that information.)

By default, `fmdump` displays the contents of the fault log, which records the result of each diagnosis made by the fault manager or one of its component modules. The error log can be selected using `-e`, the info log with `-i`, and the high-value info log with `-I`; or a specific log file path may be specified as [*file*] on the command line (which will dump just that file and not look for rotated versions of the log). One can also use option `-A` to aggregate all logs, or a set of log file paths listed on the command line.

An example of a default `fmdump` display follows:

```
# fmdump
TIME                 UUID                                 SUNW-MSG-ID EVENT
```

```
Mar 23 14:06:35.2682 0a11a1a7-a8ce-c941-8527-8d7a9d320071 ZFS-8000-CS Diagnosed
Mar 25 14:51:41.2261 0a11a1a7-a8ce-c941-8527-8d7a9d320071 FMD-8000-4M Repaired
Mar 25 14:51:41.2523 0a11a1a7-a8ce-c941-8527-8d7a9d320071 FMD-8000-6U Resolved
May 31 23:35:39.9146 c63ac52e-506b-c1cc-e965-ff3b8544490d SMF-8000-YX Diagnosed
...
```

(Output wraps on displays of 80 or fewer characters.)

This dumps the fault log, because no command line options or arguments selected any other log. The fault log records the lifecycle of problems diagnosed by the Fault Manager or its component modules, from initial problem diagnosis to problem resolution.

Each problem recorded in the fault log is identified by:

- The timestamp of the event describing the problem lifecycle state change.
- A Universal Unique Identifier (UUID) that can be used to uniquely identify this particular problem across any set of systems. All events describing problem lifecycle state changes for a given problem will use the same UUID (as above: we see initial diagnosis and, later, repair and resolution all quoting the same problem UUID).
- A message identifier that can be used to access a corresponding knowledge article located at Sun's web site, http://www.oracle.com/us/sun/msg/

If a problem requires action by a human administrator or service technician or affects system behavior, the Fault Manager also issues a human-readable message to `syslogd(1M)`. This message provides a summary of the problem and a reference to the knowledge article on the Sun web site, http://www.oracle.com/us/sun/msg/. The `fmdump` utility can dump `fltlog` entries in a similar format to that rendered to `syslog` through use of the `-m` option.

You can use the `-v` and `-V` options to expand the display from a single-line summary to increased levels of detail for each event recorded in the log. The `-p` option can be used with `-V` to request “prettier” output.

The `-c`, `-t`, `-T`, `-n` and `-u` options can be used to filter the output by selecting only those events that match the specified class, range of times, or uuid. If more than one filter option is present on the command-line, the options combine to display only those events that are selected by the logical `AND` of the options. If more than one instance of the same filter option is present on the command-line, the like options combine to display any events selected by the logical `OR` of the options. For example, the command:

```
# fmdump -u uuid1 -u uuid2 -t 02Dec09
```

...selects events whose attributes are (`uuid1` `OR` `uuid2`) `AND` (time on or after `02Dec09`).

Options The following options are supported:

`-A`

Perform log aggregation. If one of more log file paths are listed on the command line, then aggregate those files; otherwise aggregate all known log types, including all `logadm`-rotated files. Logs are merged in time order but with the characteristic that any two records from the same log file are ordered in the aggregation exactly as they were in the original log file (which is in the order they were received and processed by the Fault Manager, which will be an approximate time order).

You cannot use other log set selection options with `-A`: `-e`, `-i`, or `-I`. Filter options such as `-c`, `-t`, `-T`, and `-n` can be used, but `-u` cannot. Output options `-v`, `-V`, and `-p` are available, but `-m` is not. Option `-f` will follow all the selected logs.

`-c` *class*

Select events that match the specified class. The class argument can use the glob pattern matching syntax described in `sh(1)`. The class represents a hierarchical classification string indicating the type of telemetry event. More information about Sun's telemetry protocol is available at Sun's web site, http://www.oracle.com/us/sun/msg/.

`-e`

Display events from the fault management error log instead of the fault log.

The error log contains Private telemetry information used by Sun's automated diagnosis software. This information is recorded to facilitate post-mortem analysis of problems and event replay, and should not be parsed or relied upon for the development of scripts and other tools. See `attributes(5)` for information about Sun's rules for Private interfaces.

`-E` *ENA*

Select events, of any generation, that match the specified ENA value. For detectors that support ENA, this option can be used to show multiple events associated with the same operation.

`-f`

Follow the growth of the log file (or files if using `-A`) by waiting for additional data. `fmdump` enters an infinite loop where it will sleep for a second, attempt to read and format new data from the log file, and then go back to sleep. This loop can be terminated at any time by sending an interrupt (`Control-C`).

`-m`

Print the localized diagnosis message associated with each entry in the fault log.

`-n` *name*[.*name*]*[=*value*]

Select log events (from the log(s) selected on the command line) that have properties with a matching name (and optionally a matching value). For string properties the value can be a regular expression match. Regular expression syntax is described in the EXTENDED REGULAR EXPRESSIONS section of the `regex(5)` manual page. Be careful when using the characters:

```
$  *  {  ^  |  (  )  \
```

...or a regular expression, because these are meaningful to the shell. It is safest to enclose any of these in single quotes. For numeric properties, the value can be octal, hex, or decimal.

`-p`
Combined with `-V` (very verbose) option, requests that the pretty-printing options that are available are, in fact, performed.

`-R` *dir*
Use the specified root directory for the log files accessed by `fmdump`, instead of the default root (`/`).

`-t` *time*
Select events that occurred at or after the specified time. The time can be specified using any of the following forms:

mm/*dd*/*yy hh*:*mm*:*ss*
Month, day, year, hour in 24-hour format, minute, and second. Any amount of whitespace can separate the date and time. The argument should be quoted so that the shell interprets the two strings as a single argument.

mm/*dd*/*yy hh*:*mm*
Month, day, year, hour in 24-hour format, and minute. Any amount of whitespace can separate the date and time. The argument should be quoted so that the shell interprets the two strings as a single argument.

mm/*dd*/*yy*
12:00:00AM on the specified month, day, and year.

ddMonyy hh:*mm*:*ss*
Day, month name, year, hour in 24-hour format, minute, and second. Any amount of whitespace can separate the date and time. The argument should be quoted so that the shell interprets the two strings as a single argument.

ddMonyy hh:*mm*
Day, month name, year, hour in 24-hour format, and minute. Any amount of whitespace can separate the date and time. The argument should be quoted so that the shell interprets the two strings as a single argument.

Mon dd hh:*mm*:*ss*
Month, day, hour in 24-hour format, minute, and second of the current year.

yyyy-*mm*-*dd* `[T` *hh*:*mm*`[`:*ss*`]]`
Year, month, day, and optional hour in 24-hour format, minute, and second. The second, or hour, minute, and second, can be optionally omitted.

ddMonyy
12:00:00AM on the specified day, month name, and year.

hh:*mm*:*ss*
Hour in 24-hour format, minute, and second of the current day.

hh:*mm*
Hour in 24-hour format and minute of the current day.

T`ns` | *T*`nsec`
T nanoseconds ago where T is an integer value specified in base 10.

T`us` | *T*`usec`
T microseconds ago where T is an integer value specified in base 10.

T`ms` | *T*`msec`
T milliseconds ago where T is an integer value specified in base 10.

T`s` | *T*`sec`
T seconds ago where *T* is an integer value specified in base 10.

T`m` | *T*`min`
T minutes ago where *T* is an integer value specified in base 10.

T`h` | *T*`hour`
T hours ago where *T* is an integer value specified in base 10.

T`d` | *T*`day`
T days ago where *T* is an integer value specified in base 10.

You can append a decimal fraction of the form `.`*n* to any `-t` option argument to indicate a fractional number of seconds beyond the specified time.

`-T` *time*
Select events that occurred at or before the specified time. *time* can be specified using any of the time formats described for the `-t` option.

`-u` *uuid*
Select problem diagnosis events in the fault log that exactly match the specified *uuid*. Each diagnosis is associated with a Universal Unique Identifier (UUID) for identification purposes. The `-u` option can be combined with other options, such as `-v`, to show all of the details associated with a particular diagnosis. Note that multiple fault log events can be associated with the same problem diagnosis UUID—all events describing the lifecycle of a single problem (from initial diagnosis to final resolution) quote the same problem UUID.

If the `-e` option and `-u` option are both present, the error events that are cross-referenced by the specified diagnosis are displayed.

`-v`
Display verbose event detail. The event display is enlarged to show additional common members of the selected events.

`-V`
Display very verbose event detail. The event display is enlarged to show every member of the name-value pair list associated with each event. In addition, for fault logs, the event display includes a list of cross-references to the corresponding errors that were associated with the diagnosis.

Use `-p` with `-V` to request pretty-printing.

Operands The following operands are supported:

file Specifies an alternate log file (or files if using `-A`) to display instead of the system fault log. The `fmdump` utility determines the type of the specified log automatically and produces appropriate output for the selected log.

Examples **EXAMPLE 1** Retrieving Given Class from `fmd` Log

Use any of the following commands to retrieve information about a specified class from the `fmd` log. The complete class name is `ereport.io.ddi.context`.

```
# fmdump -Ve -c 'ereport.io.ddi.context'
# fmdump -Ve -c 'ereport.*.context'
# fmdump -Ve -n 'class=ereport.io.ddi.context'
# fmdump -Ve -n 'class=ereport.*.context'
```

Any of the preceding commands produces the following output:

```
Oct 06 2007 11:53:20.975021712 ereport.io.ddi.context
        nvlist version: 0
                class = ereport.io.ddi.context
                ena = 0x1b03a15ecf00001
                detector = (embedded nvlist)
                nvlist version: 0
                        version = 0x0
                        scheme = dev
                        device-path = /
                (end detector)

                __ttl = 0x1
                __tod = 0x470706b0 0x3a1da690
```

EXAMPLE 2 Retrieving Specific Detector Device Path from `fmd` Log

The following command retrieves a detector device path from the `fmd` log.

```
# fmdump -Ve -n 'detector.device-path=.*/disk@1,0$'
Oct 06 2007 12:04:28.065660760 ereport.io.scsi.disk.rqs
nvlist version: 0
       class = ereport.io.scsi.disk.rqs
       ena = 0x453ff3732400401
       detector = (embedded nvlist)
                nvlist version: 0
                        version = 0x0
                        scheme = dev
                        device-path = /pci@0,0/pci1000,3060@3/disk@1,0
                (end detector)
```

EXAMPLE 2 Retrieving Specific Detector Device Path from `fmd` Log *(Continued)*

```
__ttl = 0x1
__tod = 0x4707094c 0x3e9e758
```

Exit Status The following exit values are returned:

`0` Successful completion. All records in the log file were examined successfully.

`1` A fatal error occurred. This prevented any log file data from being examined, such as failure to open the specified file.

`2` Invalid command-line options were specified.

`3` The log file was opened successfully, but one or more log file records were not displayed, either due to an I/O error or because the records themselves were malformed. `fmdump` issues a warning message for each record that could not be displayed, and then continues on and attempts to display other records.

Files

`/var/fm/fmd`	Fault management log directory
`/var/fm/fmd/errlog`	Fault management error log
`/var/fm/fmd/fltlog`	Fault management fault log
`/var/fm/fmd/infolog_hival`	High-value informational log
`/var/fm/fmd/infolog`	Informational log

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/fault-management
Interface Stability	See below.

The command-line options are Uncommitted. The human-readable error log and informational log output is Private. The human-readable fault log output is Uncommitted.

See Also `sh(1)`, `fmadm(1M)`, `fmd(1M)`, `fmstat(1M)`, `logadm(1M)`, `syslogd(1M)`, `libexacct(3LIB)`, `attributes(5)`, `regex(5)`

Oracle Solaris 11.1 Administration: Security Services

http://www.oracle.com/us/sun/msg/

Notes Fault logs contain references to records stored in error logs that can be displayed using `fmdump -V` to understand the errors that were used in the diagnosis of a particular fault. These links are preserved if an error log is renamed as part of log rotation. They can be broken by removing

an error log file, or by moving it to another filesystem directory. `fmdump` can not display error information for such broken links. It continues to display any and all information present in the fault log.

Name fmstat – report fault management module statistics

Synopsis

```
fmstat [-astTz] [-d u | d ] [-m module] [interval [count]]
```

Description The `fmstat` utility can be used by administrators and service personnel to report statistics associated with the Solaris Fault Manager, `fmd(1M)` and its associated set of modules. The Fault Manager runs in the background on each Solaris system. It receives telemetry information relating to problems detected by the system software, diagnoses these problems, and initiates proactive self-healing activities such as disabling faulty components.

You can use `fmstat` to view statistics for diagnosis engines and agents that are currently participating in fault management. The documentation for `fmd(1M)`, `fmadm(1M)`, and `fmdump(1M)` describes more about tools to observe fault management activities.

If the `-m` option is present or the `-t` option is present, `fmstat` reports any statistics kept by the specified fault management module. The module list can be obtained using `fmadm config`.

If the `-m` option is not present, `fmstat` reports the following statistics for each of its client modules:

`module`	The name of the fault management module, as reported by `fmadm config`.
`ev_recv`	The number of telemetry events received by the module.
`ev_acpt`	The number of events accepted by the module as relevant to a diagnosis.
`wait`	The average number of telemetry events waiting to be examined by the module.
`svc_t`	The average service time for telemetry events received by the module, in milliseconds.
`%w`	The percentage of time that there were telemetry events waiting to be examined by the module.
`%b`	The percentage of time that the module was busy processing telemetry events.
`open`	The number of active cases (open problem investigations) owned by the module.
`solve`	The total number of cases solved by this module since it was loaded.
`memsz`	The amount of dynamic memory currently allocated by this module.
`bufsz`	The amount of persistent buffer space currently allocated by this module.

The `fmstat` utility requires the user to posses the `SYS_ADMIN` privilege. Refer to the *Oracle Solaris 11.1 Administration: Security Services* for more information about how to configure Solaris privileges.

Options The following options are supported:

`-a` Print all statistics for a module, including those kept on its behalf by `fmd`. If the `-a` option is not present, only those statistics kept by the module are reported. If the `-a` option is used without the `-m` *module*, a set of global statistics associated with `fmd` are displayed.

`-d u | d` Display a time stamp.

Specify `u` for a printed representation of the internal representation of time. See `time(2)`. Specify `d` for standard date format. See `date(1)`.

`-m` *module* Print a report on the statistics associated with the specified fault management module, instead of the default statistics report. Modules can publish an arbitrary set of statistics to help Sun service the fault management software itself. The module statistics constitute a Private interface. See `attributes(5)` for information on Sun's rules for Private interfaces. Scripts should not be written that depend upon the values of fault management module statistics as they can change without notice.

`-s` Print a report on Soft Error Rate Discrimination (SERD) engines associated with the module instead of the default module statistics report. A SERD engine is a construct used by fault management software to determine if a statistical threshold measured as *N* events in some time *T* has been exceeded. The `-s` option can only be used in combination with the `-m` option.

`-t` Print a report on the statistics associated with each fault management event transport. Each fault management module can provide the implementation of one or more event transports.

`-T` Print a table of the authority information associated with each fault management event transport. If the `-m` option is present, only transports associated with the specified module are displayed.

`-z` Omit statistics with a zero value from the report associated with the specified fault management module. The `-z` option can only be used in combination with the `-m` option.

Operands The following operands are supported:

count Print only count reports, and then exit.

interval Print a new report every *interval* seconds.

If no *interval* and no *count* are specified, a single report is printed and `fmstat` exits. If an *interval* is specified but no *count* is specified, `fmstat` prints reports every *interval* seconds indefinitely until the command is interrupted.

Exit Status The following exit values are returned:

`0` Successful completion.

1 A fatal error occurred. A fatal error could be the failure to communicate with fmd(1M). It could also be that insufficient privileges were available to perform the requested operation.

2 Invalid command-line options were specified.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/fault-management
Interface Stability	See below.

The command-line options are Committed. The human-readable default report is Uncommitted. The human-readable module report is Private.

See Also fmadm(1M), fmd(1M), fmdump(1M), attributes(5)

Oracle Solaris 11.1 Administration: Security Services

Name fmthard – populate label on hard disks

Synopsis

```
SPARC  fmthard -d data | -n volume_name | -s datafile [-i] /dev/rdsk/c?
           [t?] d?s2

  x86  fmthard -d data | -n volume_name | -s datafile [-i] /dev/rdsk/c?
           [t?] d?s2
```

Description The `fmthard` command updates the VTOC (Volume Table of Contents) on hard disks and, on x86 systems, adds boot information to the Solaris `fdisk` partition. One or more of the options `-s` *datafile*, `-d` *data*, or `-n` *volume_name* must be used to request modifications to the disk label. To print disk label contents, see `prtvtoc(1M)`. The `/dev/rdsk/c?[t?]d?s2` file must be the character special file of the device where the new label is to be installed. On x86 systems, `fdisk(1M)` must be run on the drive before `fmthard`.

If you are using an x86 system, note that the term "partition" in this page refers to *slices* within the x86 `fdisk` partition on x86 machines. Do not confuse the partitions created by `fmthard` with the partitions created by `fdisk`.

Options The following options are supported:

`-d` *data*
: The *data* argument of this option is a string representing the information for a particular partition in the current VTOC. The string must be of the format *part:tag:flag:start:size* where *part* is the partition number, *tag* is the ID TAG of the partition, *flag* is the set of permission flags, *start* is the starting sector number of the partition, and *size* is the number of sectors in the partition. See the description of the *datafile* below for more information on these fields.

`-i`
: This option allows the command to create the desired VTOC table, but prints the information to standard output instead of modifying the VTOC on the disk.

`-n` *volume_name*
: This option is used to give the disk a *volume_name* up to 8 characters long.

`-s` *datafile*
: This option is used to populate the VTOC according to a *datafile* created by the user. If the *datafile* is – (a hyphen), `fmthard` reads from standard input. The *datafile* format is described below. This option causes all of the disk partition timestamp fields to be set to zero.

  Every VTOC generated by `fmthard` will also have partition 2, by convention, that corresponds to the whole disk. If the input in *datafile* does not specify an entry for partition 2, a default partition 2 entry will be created automatically in VTOC with the tag `V_BACKUP` and size equal to the full size of the disk.

The *datafile* contains one specification line for each partition, starting with partition 0. Each line is delimited by a new-line character (`\n`). If the first character of a line is an asterisk (*), the line is treated as a comment. Each line is composed of entries that are position-dependent, separated by white space and having the following format:

partition tag flag starting_sector size_in_sectors

where the entries have the following values:

partition
The partition number. Currently, for Solaris SPARC, a disk can have up to 8 partitions, `0`–`7`. Even though the *partition* field has 4 bits, only 3 bits are currently used. For x86, all 4 bits are used to allow slices `0`–`15`. Each Solaris `fdisk` partition can have up to 16 slices.

tag
The partition tag: a decimal number. The following are reserved codes: `0` (`V_UNASSIGNED`), `1` (`V_BOOT`), `2` (`V_ROOT`), `3` (`V_SWAP`), `4` (`V_USR`), `5` (`V_BACKUP`), `6` (`V_STAND`), `7` (`V_VAR`), `8` (`V_HOME`), `12` (`V_SYSTEM`), and `24` (`V_BIOS_BOOT`).

flag
The flag allows a partition to be flagged as unmountable or read only, the masks being: `V_UNMNT 0x01`, and `V_RONLY 0x10`. For mountable partitions use `0x00`.

starting_sector
The sector number (decimal) on which the partition starts.

size_in_sectors
The number (decimal) of sectors occupied by the partition.

You can save the output of a `prtvtoc` command to a file, edit the file, and use it as the *datafile* argument to the `-s` option.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also

`uname(1)`, `format(1M)`, `prtvtoc(1M)`, `attributes(5)`

x86 Only `fdisk(1M)`, `installgrub(1M)`

Notes

Special care should be exercised when overwriting an existing VTOC, as incorrect entries could result in current data being inaccessible. As a precaution, save the old VTOC.

For disks under two terabytes, `fmthard` cannot write a VTOC on an unlabeled disk. Use `format(1M)` for this purpose.

Name format – disk partitioning and maintenance utility

Synopsis

```
format [-f command-file] [-l log-file] [-x data-file]
     [-d disk-name] [-t disk-type] [-p partition-name]
     [-s] [-m] [-M] [-e] [disk-list]

format -L label-type -d disk-name
```

Description `format` enables you to format, label, repair, and analyze disks on your system. Unlike previous disk maintenance programs, `format` runs under SunOS. Because there are limitations to what can be done to the system disk while the system is running, `format` is also supported within the memory-resident system environment. For most applications, however, running `format` under SunOS is the more convenient approach.

`format` first uses the disk list defined in *data-file* if the `-x` option is used. `format` then checks for the `FORMAT_PATH` environment variable, a colon-separated list of filenames and/or directories. In the case of a directory, `format` searches for a file named `format.dat` in that directory; a filename should be an absolute pathname, and is used without change. `format` adds all disk and partition definitions in each specified file to the working set. Multiple identical definitions are silently ignored. If `FORMAT_PATH` is not set, the path defaults to `/etc/format.dat`.

disk-list is a list of disks in the form `c?t?d?`, or `/dev/rdsk/c?t?d?s?`, `/dev/chassis/?/disk`. With the last two forms, shell wildcard specifications are supported. For example, specifying `/dev/rdsk/c2*` causes `format` to work on all drives connected to controller `c2` only. If no *disk-list* is specified, `format` lists all the disks present in the system that can be administered by `format`.

Removable media devices are listed only when users execute `format` in expert mode (option `-e`). This feature is provided for backward compatibility. Use `rmformat(1)` for rewritable removable media devices.

Options The following options are supported:

`-d` *disk-name* — Specify which disk should be made current upon entry into the program. The disk is specified by its logical name (for instance, `-d c0t1d0` or `/dev/chassis/SYS/HD0/disk`). This can also be accomplished by specifying a single disk in the disk list.

`-e` — Enable SCSI expert menu. Note this option is not recommended for casual use.

`-f` *command-file* — Take command input from *command-file* rather than the standard input. The file must contain commands that appear just as they would if they had been entered from the keyboard. With this option, `format` does not issue `continue?` prompts; there is no need to specify y(es) or n(o) answers in the *command-file*. In non-interactive mode, `format` does not initially expect the input of a disk selection number. The user

must specify the current working disk with the `-d` *disk-name* option when format is invoked, or specify `disk` and the disk selection number in the *command-file*.

`-l` *log-file* — Log a transcript of the `format` session to the indicated *log-file*, including the standard input, the standard output and the standard error.

`-L` *label-type* — Immediately, and non-interactively, write a default label of type *label-type*, to the disk specified with `-d`. *label-type* must be either `efi` or `vtoc`. Existing label, if any, will be overwritten with *label-type*. On an x86 machine, the whole disk will default to one Solaris partition labeled with *label-type*; all `fdisk` partitions will be lost.

`-m` — Enable extended messages. Provides more detailed information in the event of an error.

`-M` — Enable extended and diagnostic messages. Provides extensive information on the state of a SCSI device's mode pages, during formatting.

`-p` *partition-name* — Specify the partition table for the disk which is current upon entry into the program. The table is specified by its name as defined in the data file. This option can be used only if a disk is being made current, and its type is either specified or available from the disk label.

`-s` — Silent. Suppress all of the standard output. Error messages are still displayed. This is generally used in conjunction with the `-f` option.

`-t` *disk-type* — Specify the type of disk which is current upon entry into the program. A disk's type is specified by name in the data file. This option can only be used if a disk is being made current as described above.

`-x` *data-file* — Use the list of disks contained in *data-file*.

Usage When you invoke format with no options or with the `-e`, `-l`, `-m`, `-M`, or `-s` options, the program displays a numbered list of available disks and prompts you to specify a disk by list number. If the machine has more than a screenful of disks, press SPACE to see the next screenful of disks.

You can specify a disk by list number even if the disk is not displayed in the current screenful. For example, if the current screen shows disks 11-20, you can enter `25` to specify the twenty-fifth disk on the list. If you enter a number for a disk that is not currently displayed, `format` prompts you to verify your selection. If you enter a number from the displayed list, `format` silently accepts your selection.

After you specify a disk, `format` displays its main menu. This menu enables you to perform the following tasks:

analyze	Run read, write, compare tests, and data purge. The data purge function implements the National Computer Security Center Guide to Understanding Data Remnance (NCSC-TG-025 version 2) Overwriting Algorithm. See NOTES.
backup	Search for backup labels.
cache	Enable, disable, and query the state of the write cache and read cache. This menu item only appears when `format` is invoked with the `-e` option, and is only supported on SCSI devices..
current	Display the device name, the disk geometry, and the pathname to the disk device.
defect	Retrieve and print defect lists. This option is supported only on SCSI devices. IDE disks perform automatic defect management. Upon using the `defect` option on an IDE disk, you receive the message: `Controller does not support defect management or disk supports automatic defect management.`
disk	Choose the disk that will be used in subsequent operations (known as the current disk.)
fdisk	Run the `fdisk(1M)` program to create a `fdisk` partition for Solaris software (x86 based systems only).
format	Format and verify the current disk. This option is supported only on SCSI devices. IDE disks are pre-formatted by the manufacturer. Upon using the `format` option on an IDE disk, you receive the message: `Cannot format this drive. Please use your manufacturer-supplied formatting utility.`
inquiry	Display the vendor, product name, and revision level of the current drive.
label	Write a new label to the current disk.
partition	Create and modify slices.
quit	Exit the format menu.
repair	Repair a specific block on the disk.
save	Save new disk and slice information.
type	Select (define) a disk type.
verify	Read and display labels. Print information such as the number of cylinders, alternate cylinders, heads, sectors, and the partition table.
volname	Label the disk with a new eight character volume name.

Environment Variables

`FORMAT_PATH`	a colon-separated list of filenames and/or directories of disk and partition definitions. If a directory is specified, `format` searches for the file `format.dat` in that directory.

Files

`/etc/format.dat`	default data file

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also

`fmthard(1M)`, `prtvtoc(1M)`, `rmformat(1)`, `format.dat(4)`, `attributes(5)`, `sd(7D)`

Oracle Solaris Administration: Common Tasks

x86 Only

`fdisk(1M)`

Warnings

When the `format` function is selected to format the Maxtor 207MB disk, the following message displays:

```
Mode sense page(4) reports rpm value as 0, adjusting it to 3600
```

This is a drive bug that may also occur with older third party drives. The above message is not an error; the drive will still function correctly.

Cylinder 0 contains the partition table (disk label), which can be overwritten if used in a raw disk partition by third party software.

`format` supports writing EFI-compliant disk labels in order to support disks or LUNs with capacities greater than one terabyte. However, care should be exercised since many software components, such as filesystems and volume managers, are still restricted to capacities of one terabyte or less. See the *Oracle Solaris Administration: Common Tasks* for additional information.

By default, on an unlabeled disk, EFI labels will be written on disks larger than 2 TB. When `format` is invoked with the `-e` option, on writing the label, the label type can be chosen. Booting is not currently supported on a disk with an EFI label.

Notes

`format` provides a help facility you can use whenever format is expecting input. You can request help about what information is expected by simply entering a question mark (`?`) and `format` prints a brief description of what type of input is needed. If you enter a `?` at the menu prompt, a list of available commands is displayed.

For SCSI disks, formatting is done with both Primary and Grown defects list by default. However, if only Primary list is extracted in defect menu before formatting, formatting will be done with Primary list only.

Changing the state of the caches is only supported on SCSI devices, and not all SCSI devices support changing or saving the state of the caches.

The NCSC-TG-025 algorithm for overwriting meets the DoD 5200.28-M (ADP Security Manual) Eraser Procedures specification. The NIST Guidelines for Media Sanitization (NIST SP 800-88) also reference this algorithm.

Name fruadm – prints and updates customer data associated with FRUs

Synopsis

```
/usr/platform/sun4u/sbin/fruadm

/usr/platform/sun4u/sbin/fruadm -l

/usr/platform/sun4u/sbin/fruadm [-r] path [text]
```

Description `fruadm` prints or sets the customer data for Field-Replaceable Units (FRUs).

Without arguments, `fruadm` prints the paths of all FRU ID-capable FRUs (containers) in the system, along with the contents of the customer data record, if present, for each such FRU; for FRUs without customer data, `fruadm` prints only the container's path.

Only a privileged user can create or update data in containers. The privileges required to perform these write operations are hardware dependent. Typically, a default system configuration restricts write operations to the superuser or to the platform-administrator user.

Options The following options are supported:

`-l` List the system's frutree paths.

`-r` Recursively display or update the data for all containers rooted at the argument *path*.

Operands The following operands are supported:

path A full or partial system frutree path for or under which to print or set the customer data. The first field of each line of output of `fruadm -l` gives the valid full `frutree` paths for the system.

Paths can include shell meta-characters; such paths should be quoted appropriately for the user's shell. For partial paths, the first matching full path is selected for display or update. Without the `-r` option, the path must be that of a container; with the `-r` option, all containers (if any) under *path* will be selected.

text Up to 80 characters of text set as the customer data. If the text contains white space or shell metacharacters, it should be quoted appropriately for the user's shell.

Examples **EXAMPLE 1** Displaying All Customer Data

The following example prints all customer data available from FRUs on the system. For containers with no customer data, only the containers' paths will be listed.

```
example% fruadm
```

EXAMPLE 2 Displaying Customer Data For a Single FRU

The following command prints the customer data, if present, for the specified FRU:

```
example% fruadm /frutree/chassis/system-board
```

EXAMPLE 3 Displaying Customer Data For a Single FRU

The following command prints the customer data, if present, for the first `mem-module` found:

```
example% fruadm mem-module
```

EXAMPLE 4 Setting Customer Data

The following example sets the customer data for a FRU:

```
example# fruadm system-board 'Asset Tag 123456'
```

EXAMPLE 5 Setting Customer Data

The following command sets the customer data for all FRUs under `chassis`:

```
example# fruadm -r /frutree/chassis "Property of XYZ, Inc."
```

Exit Status

The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/fru-id
Interface Stability	Uncommitted

See Also

`prtfru(1M)`, `attributes(5)`

Name fsck – check and repair file systems

Synopsis

```
fsck [-F FSType] [-m] [-V] [-v] [special]...

fsck [-F FSType] [-n | N | y | Y] [-V] [-v]
     [-o FSType-specific-options] [special]...
```

Description `fsck` audits and interactively repairs inconsistent file system conditions. If the file system is inconsistent the default action for each correction is to wait for the user to respond `yes` or `no`. If the user does not have write permission `fsck` defaults to a `no` action. Some corrective actions will result in loss of data. The amount and severity of data loss can be determined from the diagnostic output.

FSType-specific-options are options specified in a comma-separated (with no intervening spaces) list of options or keyword-attribute pairs for interpretation by the *FSType*-specific module of the command.

special represents the character special device on which the file system resides, for example, `/dev/rdsk/c1t0d0s7`. Note: the character special device, not the block special device, should be used. `fsck` will not work if the block device is mounted.

If no *special* device is specified `fsck` checks the file systems listed in `/etc/vfstab`. Those entries in `/etc/vfstab` which have a character special device entry in the `fsckdev` field and have a non-zero numeric entry in the `fsckpass` field will be checked. Specifying `-F` *FSType* limits the file systems to be checked to those of the type indicated.

If *special* is specified, but `-F` is not, the file system type will be determined by looking for a matching entry in `/etc/vfstab`. If no entry is found, the default local file system type specified in `/etc/default/fs` will be used.

If a file system type supports parallel checking, for example, `ufs`, some file systems eligible for checking may be checked in parallel. Consult the file system-specific man page (for example, `fsck_ufs(1M)`) for more information.

Options The following generic options are supported:

`-F` *FSType*
Specify the file system type on which to operate.

`-m`
Check but do not repair. This option checks that the file system is suitable for mounting, returning the appropriate exit status. If the file system is ready for mounting, `fsck` displays a message such as:

```
ufs fsck: sanity check: /dev/rdsk/c0t3d0s1 okay
```

`-n` | `-N`
Assume a `no` response to all questions asked by `fsck`; do not open the file system for writing.

`-V`
Echo the expanded command line but do not execute the command. This option may be used to verify and to validate the command line.

`-v`
Enables verbose output. Might not be supported by all filesystem-specific `fsck` implementations.

`-y | Y`
Assume a `yes` response to all questions asked by `fsck`.

`-o` *specific-options*
These *specific-options* can be any combination of the following separated by commas (with no intervening spaces).

`b=`*n*
Use block *n* as the super block for the file system. Block 32 is always one of the alternate super blocks. Determine the location of other super blocks by running `newfs(1M)` with the `-Nv` options specified.

`c`
If the file system is in the old (static table) format, convert it to the new (dynamic table) format. If the file system is in the new format, convert it to the old format provided the old format can support the file system configuration. In interactive mode, `fsck` will list the direction the conversion is to be made and ask whether the conversion should be done. If a negative answer is given, no further operations are done on the file system. In preen mode, the direction of the conversion is listed and done if possible without user interaction. Conversion in preen mode is best used when all the file systems are being converted at once. The format of a file system can be determined from the first line of output from `fstyp(1M)`. Note: the `c` option is seldom used and is included only for compatibility with pre-4.1 releases. There is no guarantee that this option will be included in future releases.

`f`
Force checking of file systems regardless of the state of their super block clean flag.

`p`
Check and fix the file system non-interactively ("preen"). Exit immediately if there is a problem requiring intervention. This option is required to enable parallel file system checking.

`w`
Check writable file systems only.

Exit Status `0`
file system is unmounted and OK

`1`
erroneous parameters are specified

32
file system is unmounted and needs checking (`fsck -m` only)

33
file system is already mounted

34
cannot stat device

35
a filesystem that is mounted read/write was modified – reboot

36
uncorrectable errors detected - terminate normally

37
a signal was caught during processing

39
uncorrectable errors detected - terminate immediately

40
file system is mounted read-only and is OK

Usage The `fsck` command is *large file aware* for UFS file systems, per the `largefile(5)` man page.

Files `/etc/default/fs`
default local file system type. Default values can be set for the following flags in `/etc/default/fs`. For example: `LOCAL=ufs`.

`LOCAL`
The default partition for a command if no FSType is specified.

`/etc/vfstab`
list of default parameters for each file system

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os
Interface Stability	Committed

See Also `clri(1M)`, `fsck_ufs(1M)`, `fsdb_ufs(1M)`, `fsirand(1M)`, `fstyp(1M)`, `mkfs(1M)`, `mkfs_ufs(1M)`, `mountall(1M)`, `newfs(1M)`, `reboot(1M)`, `vfstab(4)`, `attributes(5)`, `largefile(5)`, `ufs(7FS)`

Warnings The operating system buffers file system data. Running `fsck` on a mounted file system can cause the operating system's buffers to become out of date with respect to the disk. For this reason, the file system should be *unmounted* when `fsck` is used. If this is not possible, care should be taken that the system is quiescent and that it is rebooted immediately after `fsck` is run. Quite often, however, this will not be sufficient. A panic will probably occur if running `fsck` on a file system modifies the file system.

Notes This command may not be supported for all *FSTypes*.

Starting with Solaris 9, `fsck` manages extended attribute data on the disk. (See `fsattr(5)` for a description of extended file attributes.) A file system with extended attributes can be mounted on versions of Solaris that are not attribute-aware (versions prior to Solaris 9), but the attributes will not be accessible and `fsck` will strip them from the files and place them in `lost+found`. Once the attributes have been stripped, the file system is completely stable on versions of Solaris that are not attribute-aware, but would be considered corrupted on attribute-aware versions. In the latter circumstance, run the attribute-aware `fsck` to stabilize the file system before using it in an attribute-aware environment.

Name fsck_pcfs – file system consistency check and interactive repair

Synopsis

```
fsck -F pcfs [generic_options] special

fsck -F pcfs [generic_options] [-o specific_options] special
```

Description The `fsck` utility audits and interactively repairs inconsistent conditions on file systems. *special* represents the character special device on which the file system resides, for example `/dev/disk`. The character special device, not the block special device, should be used.

In the case of correcting serious inconsistencies, by default, `fsck` asks for confirmation before making a repair and waits for the operator to respond either `yes` or `no`. If the operator does not have write permission on the file system, `fsck` defaults to a `-n` (no corrections) action. See `fsck(1M)`.

Repairing some file system inconsistencies may result in loss of data. The amount and severity of data loss may be determined from the diagnostic output.

When executed with the verify option (`-o v`), `fsck_pcfs` automatically scans the entire file system to verify that all of its allocation units are accessible. If it finds any units inaccessible, it updates the file allocation table (FAT) appropriately. It also updates any effected directory entries to reflect the problem. This directory update includes truncating the file at the point in its allocation chain where the file data is no longer accessible. Any remaining accessible allocation units become orphaned.

Orphaned chains of accessible allocation units are, with the operator's concurrence, linked back into the file system as files in the root directory. These files are assigned names of the form file*NNNN*.chk, where the *N*s are digits in the integral range from `0` through `9`.

After successfully scanning and correcting any errors in the file system, `fsck` displays a summary of information about the file system. This summary includes the size of the file system in bytes, the number of bytes used in directories and individual files, and the number of available allocation units remaining in the file system.

Options

generic_options The following generic options are supported:

`-m` Check but do not repair. This option checks that the file system is suitable for mounting, returning the appropriate exit status. If the file system is ready for mounting, `fsck` displays a message such as:

```
pcfs fsck: sanity check:
/dev/disk okay
```

`-n` | `-N` Assume a no response to all questions asked by `fsck`; do not open the file system for writing.

`-V` Echo the expanded command line, but do not execute the command. This option may be used to verify and to validate the command line.

`-y` | `-Y` Assume a `yes` response to all questions asked by `fsck`.

`-o` *specific_options* Specify `pcfs` file system specific options in a comma-separated list, in any combination, with no intervening spaces.

- v Verify all allocation units are accessible prior to correcting inconsistencies in the metadata.
- p Check and fix the file system non-interactively (`preen`). Exit immediately if there is a problem requiring intervention.
- w Check writable file systems only.

Files *special* The device which contains the pcfs. A hard disk device or high-capacity removable device name much be qualified with a suffix to indicate the proper `FDISK` partition.

For example, in the names: `/dev/rdsk/c0t0d0p0:c` and `/dev/rdsk/c0t4d0s2:c`, the `:c` suffix indicates the first partition on the disk contains the `pcfs`.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/file-system/pcfs
Interface Stability	Committed

See Also `fsck(1M)`, `fstyp(1M)`, `fdisk(1M)`, `mkfs(1M)`, `mkfs_pcfs(1M)`, `mountall(1M)`, `attributes(5)`, `pcfs(7FS)`,

Warnings The operating system buffers file system data. Running `fsck` on a mounted file system can cause the operating system's buffers to become out of date with respect to the disk. For this reason, the file system should be unmounted when `fsck` is used. If this is not possible, care should be taken that the system is quiescent and that it is rebooted immediately after `fsck` is run. Quite often, however, this is not sufficient. A panic will probably occur if running `fsck` on a file system modifies the file system.

Name fsck_udfs – file system consistency check and interactive repair

Synopsis

```
fsck -F udfs [generic_options] [special . . .]

fsck -F udfs [generic_options] [-o specific_options]
     [special . . .]
```

Description `fsck` audits and interactively repairs inconsistent conditions on file systems. A file system to be checked can be specified by giving the name of the block or character special device or by giving the name of its mount point if a matching entry exists in `/etc/vfstab`.

special represents the character special device, for example, `/dev/rdsk/c0t2d0s0`, on which the file system resides. The character special device, not the block special device should be used. `fsck` does not work on a mounted block device.

If no special device is specified, all `udfs` file systems specified in the `vfstab` file with a `fsckdev` entry are checked. If the `-p` (`preen`) option is specified, `udfs` file systems with an `fsckpass` number greater than `1` are checked in parallel. See `fsck`(1M).

In the case of correcting serious inconsistencies, by default, `fsck` asks for confirmation before making a repair and waits for the operator to respond with either `yes` or `no`. If the operator does not have write permission on the file system, `fsck` defaults to the `-n` (no corrections) option. See `fsck`(1M).

Repairing some file system inconsistencies can result in loss of data. The amount and severity of data loss can be determined from the diagnostic output.

`fsck` automatically corrects innocuous inconsistencies. It displays a message for each corrected inconsistency that identifies the nature of the correction which took place on the file system. After successfully correcting a file system, `fsck` prints the number of files on that file system and the number of used and free blocks.

Inconsistencies checked are as follows:

- Blocks claimed by more than one file or the free list
- Blocks claimed by a file or the free list outside the range of the file system
- Incorrect link counts in file entries
- Incorrect directory sizes
- Bad file entry format
- Blocks not accounted for anywhere
- Directory checks, file pointing to unallocated file entry and absence of a parent directory entry
- Descriptor checks, more blocks for files than there are in the file system
- Bad free block list format
- Total free block count incorrect

Options The following options are supported:

generic_options The following *generic_options* are supported:

`-m` Check but do not repair. This option checks to be sure that the file system is suitable for mounting, and returns the appropriate exit status. If the file system is ready for mounting, `fsck` displays a message such as:

```
udfs fsck: sanity check: /dev/rdsk/c0t2d0s0 okay
```

`-n` | `-N` Assume a `no` response to all questions asked by `fsck`; do not open the file system for writing.

`-V` Echo the expanded command line, but do not execute the command. This option can be used to verify and to validate the command line.

`-y` | `-Y` Assume a `yes` response to all questions asked by `fsck`.

`-o` *specific_options* Specify `udfs` file system specific options in a comma-separated list with no intervening spaces. The following *specific_options* are available:

`f` Force checking of file systems regardless of the state of their logical volume integrity state.

`p` Check and fix the file system non-interactively (`preen`). Exit immediately if there is a problem that requires intervention. This option is required to enable parallel file system checking.

`w` Check writable file systems only.

Files `/etc/vtstab` List of default parameters for each file system.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/file-system/udfs

See Also `fsck(1M)`, `fsdb_udfs(1M)`, `fstyp(1M)`, `mkfs(1M)`, `mkfs_udfs(1M)`, `mountall(1M)`, `reboot(1M)`, `vfstab(4)`, `attributes(5)`

Warnings The operating system buffers file system data. Running `fsck` on a mounted file system can cause the operating system's buffers to become out of date with respect to the disk. For this reason, use `fsck` only when the file system is unmounted. If this is not possible, take care that the system is quiescent and that it is rebooted immediately after running `fsck`. A panic will probably occur if running `fsck` on a file system that modifies the file system while it is mounted.

If an unmount of the file system is not done before the system is shut down, the file system might become corrupted. In this case, a file system check needs to be completed before the next mount operation.

Diagnostics

not writable
: You cannot write to the device.

Currently Mounted on
: The device is already mounted and cannot run `fsck`.

FILE SYSTEM WAS MODIFIED
: File system has been modified to bring it to a consistent state.

Can't read allocation extent
: Cannot read the block containing allocation extent.

Bad tag on alloc extent
: Invalid tag detected when expecting an allocation extent.

Volume sequence tag error
: Invalid tag detected in the volume sequence.

Space bitmap tag error
: Invalid tag detected in the space bitmap.

UNEXPECTED INCONSISTENCY; RUN fsck MANUALLY
: Use `fsck` in interactive mode.

Name fsck_ufs – file system consistency check and interactive repair

Synopsis

```
fsck -F ufs [generic-options] [special]...

fsck -F ufs [generic-options] [-o specific-options]
     [special]...
```

Description The `fsck` utility audits and interactively repairs inconsistent conditions on file systems. A file system to be checked may be specified by giving the name of the block or character *special* device or by giving the name of its mount point if a matching entry exists in `/etc/vfstab`.

The *special* parameter represents the character special device, for example, `/dev/rdsk/c1t0d0s7`, on which the file system resides. The character special device, not the block special device should be used. The `fsck` utility will not work if the block device is mounted, unless the file system is error-locked.

If no *special* device is specified, all `ufs` file systems specified in the `vfstab` with a `fsckdev` entry will be checked. If the `-p` ("preen") option is specified, `ufs` file systems with an `fsckpass` number greater than `1` are checked in parallel. See `fsck`(1M).

In the case of correcting serious inconsistencies, by default, `fsck` asks for confirmation before making a repair and waits for the operator to respond either `yes` or `no`. If the operator does not have write permission on the file system, `fsck` will default to a `-n` (no corrections) action. See `fsck`(1M).

Repairing some file system inconsistencies can result in loss of data. The amount and severity of data loss can be determined from the diagnostic output.

The `fsck` utility automatically corrects innocuous inconsistencies such as unreferenced inodes, too-large link counts in inodes, missing blocks in the free list, blocks appearing in the free list and also in files, or incorrect counts in the super block. It displays a message for each inconsistency corrected that identifies the nature of the correction on the file system which took place. After successfully correcting a file system, `fsck` prints the number of files on that file system, the number of used and free blocks, and the percentage of fragmentation.

Inconsistencies checked include:

- Blocks claimed by more than one inode or the free list.
- Blocks claimed by an inode or the free list outside the range of the file system.
- Incorrect link counts.
- Incorrect directory sizes.
- Bad inode format.
- Blocks not accounted for anywhere.
- Directory checks, file pointing to unallocated inode, inode number out of range, and absence of '.' and '. .' as the first two entries in each directory.
- Super Block checks: more blocks for inodes than there are in the file system.

- Bad free block list format.
- Total free block and/or free inode count incorrect.

Orphaned files and directories (allocated but unreferenced) are, with the operator's concurrence, reconnected by placing them in the `lost+found` directory. The name assigned is the inode number. If the `lost+found` directory does not exist, it is created. If there is insufficient space in the `lost+found` directory, its size is increased.

An attempt to mount a `ufs` file system with the `-o nolargefiles` option will fail if the file system has ever contained a large file (a file whose size is greater than or equal to 2 Gbyte). Invoking `fsck` resets the file system state if no large files are present in the file system. A successful mount of the file system after invoking `fsck` indicates the absence of large files in the file system. An unsuccessful mount attempt indicates the presence of at least one large file. See `mount_ufs(1M)`.

Options The *generic-options* consist of the following options:

`-m` Check but do not repair. This option checks that the file system is suitable for mounting, returning the appropriate exit status. If the file system is ready for mounting, `fsck` displays a message such as:

```
ufs fsck: sanity check: /dev/rdsk/c0t3d0s1 okay
```

`-n` | `N` Assume a `no` response to all questions asked by `fsck`; do not open the file system for writing.

`-V` Echo the expanded command line, but do not execute the command. This option may be used to verify and to validate the command line.

`-v` Enables verbose output. Might not be supported by all filesystem-specific `fsck` implementations.

`-y` | `Y` Assume a `yes` response to all questions asked by `fsck`.

See generic `fsck(1M)` for the details for specifying *special*.

`-o` *specific-options* Specify `ufs` file system specific options. These options can be any combination of the following separated by commas (with no intervening spaces).

`b=`*n* Use block *n* as the super block for the file system. Block 32 is always one of the alternate super blocks. Determine the location of other super blocks by running `newfs(1M)` with the `-Nv` options specified.

`f` Force checking of file systems regardless of the state of their super block clean flag.

p Check and fix the file system non-interactively ("preen"). Exit immediately if there is a problem requiring intervention. This option is required to enable parallel file system checking.

w Check writable file systems only.

Files `/etc/vfstab` list of default parameters for each file system

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also `clri(1M)`, `fsck(1M)`, `fsdb_ufs(1M)`, `fsirand(1M)`, `fstyp(1M)`, `mkfs(1M)`, `mkfs_ufs(1M)`, `mount_ufs(1M)`, `mountall(1M)`, `newfs(1M)`, `reboot(1M)`, `vfstab(4)`, `attributes(5)`, `largefile(5)`, `ufs(7FS)`

Warnings The operating system buffers file system data. Running `fsck` on a mounted file system can cause the operating system's buffers to become out of date with respect to the disk. For this reason, the file system should be *unmounted* when `fsck` is used. If this is not possible, care should be taken that the system is quiescent and that it is rebooted immediately after `fsck` is run. Quite often, however, this will not be sufficient. A panic will probably occur if running `fsck` on a file system modifies the file system.

Notes It is usually faster to check the character special device than the block special device.

Running `fsck` on file systems larger than 2 Gb fails if the user chooses to use the block interface to the device:

```
fsck /dev/dsk/c?t?d?s?
```

rather than the raw (character special) device:

```
fsck /dev/rdsk/c?t?d?s?
```

Name fsdb – file system debugger

Synopsis `fsdb [-F` *FSType*`] [-V] [-o` *FSType-specific_options*`]` *special*

Description `fsdb` is a file system debugger that allows for the manual repair of a file system after a crash. *special* is a special device used to indicate the file system to be debugged. `fsdb` is intended for experienced users only. *FSType* is the file system type to be debugged. Since different *FSTypes* have different structures and hence different debugging capabilities, the manual pages for the *FSType*-specific `fsdb` should be consulted for a more detailed description of the debugging capabilities.

Options

`-F` Specify the *FSType* on which to operate. The *FSType* should either be specified here or be determinable from `/etc/vfstab` by matching the *special* with an entry in the table, or by consulting `/etc/default/fs`.

`-V` Echo the complete command line, but do not execute the command. The command line is generated by using the options and arguments provided by the user and adding to them information derived from `/etc/vfstab`. This option may be used to verify and validate the command line.

`-o` Specify *FSType*-specific options.

Usage See `largefile(5)` for the description of the behavior of `fsdb` when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

Files

`/etc/default/fs` default local file system type. Default values can be set for the following flags in `/etc/default/fs`. For example: LOCAL=ufs

`LOCAL:` The default partition for a command if no FSType is specified.

`/etc/vfstab` list of default parameters for each file system

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also `vfstab(4)`, `attributes(5)`, `largefile(5)` Manual pages for the *FSType*-specific modules of `fsdb`.

Notes This command may not be supported for all *FSTypes*.

Name fsdb_udfs – udfs file system debugger

Synopsis `fsdb [-F] udfs [`*generic_option*`] [-o` *specific_option*`] special`

Description The `fsdb_udfs` command is an interactive tool that can be used to patch up a damaged `udfs` file system. `fsdb_udfs` has conversions to translate block and i-numbers into their corresponding disk addresses. Mnemonic offsets to access different parts of an inode are also included. Mnemonic offsets greatly simplify the process of correcting control block entries or descending the file system tree.

`fsdb` contains several error-checking routines to verify inode and block addresses. These can be disabled if necessary by invoking `fsdb` with the `-o` option or by using the `o` command.

`fsdb` reads one block at a time, and therefore works with raw as well as block I/O devices. A buffer management routine is used to retain commonly used blocks of data in order to reduce the number of read system calls. All assignment operations result in an immediate write-through of the corresponding block. In order to modify any portion of the disk, `fsdb` must be invoked with the `-w` option.

Wherever possible, `adb`-like syntax has been adopted to promote the use of `fsdb` through familiarity.

Options The following options are supported:

`-o` *specific_option* Specify `udfs` file system specific options in a comma-separated list with no intervening spaces. The following specific options are supported:

o	Override some error conditions.
p=*string*	Set prompt to *string*.
w	Open for write.
?	Display usage.

Usage Numbers are considered hexadecimal by default. The user has control over how data is to be displayed or accepted. The `base` command displays or sets the input and output base. Once set, all input defaults to this base and all output displays in this base. The base can be overriden temporarily for input by preceding hexadecimal numbers by `0x`, preceding decimal numbers with a `0t`, or octal numbers with a `0`. Hexadecimal numbers beginning with `a-f` or `A-F` must be preceded with a `0x` to distinguish them from commands.

Disk addressing by `fsdb` is at the byte level. However, `fsdb` offers many commands to convert a desired inode, directory entry, block, and so forth, to a byte address. After the address has been calculated, `fsdb` records the result in the current address (`dot`).

Several global values are maintained by `fsdb`:

- Current base (referred to as `base`)
- Current address (referred to as `dot`)

- Current inode (referred to as `inode`)
- Current count (referred to as `count`)
- Current type (referred to as `type`)

Most commands use the preset value of `dot` in their execution. For example,

```
> 2:inode
```

first sets the value of dot (`.`) to `2`, colon (`:`), signifies the start of a command, and the `inode` command sets `inode` to `2`. A count is specified after a comma (`,`). Once set, count remains at this value until a new command is encountered that resets the value back to `1` (the default).

So, if

```
> 2000,400/X
```

is entered, `400` hex longs are listed from `2000`, and when completed, the value of dot is `2000 + 400 * sizeof` (long). If a RETURN is then entered, the output routine uses the current values of `dot`, `count`, and `type` and displays `400` more hex longs. An asterisk (`*`) causes the entire block to be displayed. An example showing several commands and the use of RETURN would be:

```
> 2:ino; 0:dir?d
```

or

```
> 2:ino; 0:db:block?d
```

The two examples are synonymous for getting to the first directory entry of the root of the file system. Once there, subsequently entering a RETURN, plus (`+`), or minus (`-`) advances to subsequent entries. Notice that

```
> 2:inode; :ls
```

or

```
> :ls /
```

is again synonymous.

Expressions The following symbols are recognized by `fsdb`:

RETURN Update the value of dot by the current value of *type* and *display* using the current value of *count*.

\# Update the value of dot by specifying a numeric expression. Specify numeric expressions using addition, subtraction, mulitiplication, and division operators (`+`, `-`, `*`, and `%`). Numeric expressions are evaluated from left to right and can use parentheses. After evaluation, the value of dot is updated.

, *count*	Update the count indicator. The global value of *count* is updated to *count*. The value of *count* remains until a new command is run. A *count* specifier of * attempts to show a blocks's worth of information. The default for *count* is `1`.
?*f*	Display in structured style with format specifier `f`. See `Formatted Output`.
/*f*	Display in unstructured style with format specifier `f`. See `Formatted Output`.
.	Display the value of dot.
+*e*	Increment the value of dot by the expression *e*. The amount actually incremented is dependent on the size of type: `dot = dot + e * sizeof (type)` The default for *e* is `1`.
−*e*	Decrement the value of dot by the expression *e*. See `+`.
**e*	Multiply the value of dot by the expression *e*. Multiplication and division don't use *type*. In the above calculation of dot, consider the `sizeof (type)` to be `1`.
%*e*	Divide the value of dot by the expression *e*. See *.
< *name*	Restore an address saved in register *name*. *name* must be a single letter or digit.
> *name*	Save an address in register *name*. *name* must be a single letter or digit.
=*f*	Display indicator. If *f* is a legitimate format specifier (see `Formatted Output`), then the value of dot is displayed using format specifier *f*. Otherwise, assignment is assumed. See `= [s] [e]`.
= [*s*] [*e*]	Change the value of dot using an assignment indicator. The address pointed to by dot has its contents changed to the value of the expression *e* or to the ASCII representation of the quoted (") string *s*. This can be useful for changing directory names or ASCII file information.
=+ *e*	Change the value of dot using an incremental assignment. The address pointed to by dot has its contents incremented by expression *e*.
=- e	Change the value of dot using a decremental assignment. Decrement the contents of the address pointed to by dot by expression *e*.

Commands

A command must be prefixed by a colon (`:`). Only enough letters of the command to uniquely distinguish it are needed. Multiple commands can be entered on one line by separating them by a SPACE, TAB, or semicolon (`;`).

To view a potentially unmounted disk in a reasonable manner, `fsdb` supports the `cd`, `pwd`, `ls`, and `find` commands. The functionality of each of these commands basically matches that of its UNIX counterpart. See `cd(1)`, `pwd(1)`,`ls(1)`, and`find(1)` for details. The *, ,, ?, and - wildcard characters are also supported.

The following commands are supported:

base[=*b*]	Display or set the base. All input and output is governed by the current base. Without the = *b*, displays the current base. Otherwise, sets the current base to *b*. Base is interpreted using the old value of base, so to ensure correctness use the `0`, `0t`, or `0x` prefix when changing the base. The default for base is hexadecimal.
block	Convert the value of dot to a block address.
cd [*dir*]	Change the current directory to directory *dir*. The current values of inode and dot are also updated. If `dir` is not specified, changes directories to inode 2, root (`/`).
directory	If the current inode is a directory, converts the value of dot to a directory slot offset in that directory, and dot now points to this entry.
file	Set the value of dot as a relative block count from the beginning of the file. The value of dot is updated to the first byte of this block.
find *dir* [`-name` *n*] | [`-inum` *i*]	Find files by name or i-number. Recursively searches directory *dir* and below for file names whose i-number matches `i` or whose name matches pattern *n*. Only one of the two options (`-name` or `-inum`) can be used at one time. The find `-print` is not necessary or accepted.
fill=*p*	Fill an area of disk with pattern *p*. The area of disk is delimited by dot and count.
inode	Convert the value of dot to an inode address. If successful, the current value of inode is updated as well as the value of dot. As a convenient shorthand, if `:inode` appears at the beginning of the line, the value of dot is set to the current inode and that inode is displayed in inode format.
ls [`-R`] [`-l`] *pat1 pat2...*	List directories or files. If no file is specified, the current directory is assumed. Either or both of the options can be used (but, if used, must be specified before the filename specifiers). Wild card characters are available and multiple arguments are acceptable. The long listing shows only the i-number and the name; use the inode command with `?i` to get more information.
override	Toggle the value of override. Some error conditions might be overridden if override is toggled to `on`.
prompt "*p*"	Change the `fsdb` prompt to *p*. *p* must be enclosed in quotes.

pwd	Display the current working directory.
quit	Quit `fsdb`.
tag	Convert the value of dot and if this is a valid tag, print the volume structure according to the tag.
!	Escape to the shell.

Inode Commands

In addition to the above commands, several other commands deal with inode fields and operate directly on the current inode (they still require the colon (`:`). They can be used to more easily display or change the particular fields. The value of dot is only used by the `:db` and `:ib` commands. Upon completion of the command, the value of dot is changed so that it points to that particular field. For example,

```
> :ln=+1
```

increments the link count of the current inode and sets the value of dot to the address of the link count field.

The following inode commands are supported:

at	Access time
bs	Block size
ct	Creation time
gid	Group id
ln	Link number
mt	Modification time
md	Mode
maj	Major device number
min	Minor device number
nm	This command actually operates on the directory name field. Once poised at the desired directory entry (using the `directory` command), this command allows you to change or display the directory name. For example, `> 7:dir:nm="foo"` gets the 7th directory entry of the current inode and changes its name to `foo`. Directory names cannot be made larger than the field allows. If an attempt is made to make a directory name larger than the field allows,, the string is truncated to fit and a warning message is displayed.
sz	File size

uid User ID

uniq Unique ID

Formatted Output

Formatted output comes in two styles and many format types. The two styles of formatted output are: structured and unstructured. Structured output is used to display inodes, directories, and so forth. Unstructured output displays raw data.

Format specifiers are preceded by the slash (`/`) or question mark (`?`) character. *type* is updated as necessary upon completion.

The following format specifiers are preceded by the `?` character:

i Display as inodes in the current base.

d Display as directories in the current base.

The following format specifiers are preceded by the `/` character:

b Display as bytes in the current base.

c Display as characters.

o | O Display as octal shorts or longs.

d | D Display as decimal shorts or longs.

x | X Display as hexadecimal shorts or longs.

Examples

EXAMPLE 1 Using fsdb as a calculator for complex arithmetic

The following command displays `2010` in decimal format, and is an example of using `fsdb` as a calculator for complex arithmetic.

```
> 2000+400%(20+20)=D
```

EXAMPLE 2 Using fsdb to display an i-number in idode fomat

The following command displays the i-number `386` in inode format.`386` becomes the current inode.

```
> 386:ino?i
```

EXAMPLE 3 Using fsdb to change the link count

The following command changes the link count for the current inode to `4`.

```
> :ln=4
```

EXAMPLE 4 Using fsdb to increment the link count

The following command increments the link count by 1.

```
> :ln=+1
```

EXAMPLE 5 Using fsdb to display the creation time as a hexadecimal long

The following command displays the creation time as a hexadecimal long.

```
> :ct=X
```

EXAMPLE 6 Using fsdb to display the modification time in time format

The following command displays the modification time in time format.

```
> :mt=t
```

EXAMPLE 7 Using fsdb to display in ASCII

The following command displays, in ASCII, block 0 of the file associated with the current inode.

```
> 0:file/c
```

EXAMPLE 8 Using fsdb to display the directory entries for the root inode

The following command displays the first block's directory entries for the root inode of this file system. This command stops prematurely if the EOF is reached.

```
> 2:ino,*?d
```

EXAMPLE 9 Using fsdb to change the current inode

The following command changes the current inode to that associated with the 5th directory entry (numbered from 0) of the current inode. The first logical block of the file is then displayed in ASCII.

```
> 5:dir:inode; 0:file,*/c
```

EXAMPLE 10 Using fsdb to change the i-number

The following command changes the i-number for the 7th directory slot in the root directory to 3.

```
> 2:inode; 7:dir=3
```

EXAMPLE 11 Using fsdb to change the name field

The following command changes the *name* field in the directory slot to name.

```
 > 7:dir:nm="name"
```

EXAMPLE 12 Using fsdb to display the a block

The following command displays the 3rd block of the current inode as directory entries.

EXAMPLE 13 Using fsdb to set the contents of address

The following command sets the contents of address `2050` to `0xffffffff`. `0xffffffff` can be truncated, depending on the current type.

```
> 2050=0xffff
```

EXAMPLE 14 Using fsdb to place an ASCII string at an address

The following command places the ASCII string `this is some text` at address `1c92434`.

```
> 1c92434="this is some text"
```

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/file-system/udfs

See Also

`clri(1M)`, `fsck_udfs(1M)`, `dir(4)`, `attributes(5)`

Name fsdb_ufs – ufs file system debugger

Synopsis `fsdb -F ufs [`*generic_options*`] [`*specific_options*`] `*special*

Description The `fsdb_ufs` command is an interactive tool that can be used to patch up a damaged UFS file system. It has conversions to translate block and i-numbers into their corresponding disk addresses. Also included are mnemonic offsets to access different parts of an inode. These greatly simplify the process of correcting control block entries or descending the file system tree.

`fsdb` contains several error-checking routines to verify inode and block addresses. These can be disabled if necessary by invoking `fsdb` with the `-o` option or by the use of the `o` command.

`fsdb` reads a block at a time and will therefore work with raw as well as block I/O devices. A buffer management routine is used to retain commonly used blocks of data in order to reduce the number of read system calls. All assignment operations result in an immediate write-through of the corresponding block. Note that in order to modify any portion of the disk, `fsdb` must be invoked with the `w` option.

Wherever possible, `adb`-like syntax was adopted to promote the use of `fsdb` through familiarity.

Options The following option is supported:

`-o` Specify UFS file system specific options. These options can be any combination of the following separated by commas (with no intervening spaces). The options available are:

`?`	Display usage
`o`	Override some error conditions
`p='string'`	set prompt to string
`w`	open for write

Usage Numbers are considered hexadecimal by default. However, the user has control over how data is to be displayed or accepted. The `base` command will display or set the input/output base. Once set, all input will default to this base and all output will be shown in this base. The base can be overridden temporarily for input by preceding hexadecimal numbers with '`0x`', preceding decimal numbers with '`0t`', or octal numbers with '`0`'. Hexadecimal numbers beginning with `a-f` or `A-F` must be preceded with '`0x`' to distinguish them from commands.

Disk addressing by `fsdb` is at the byte level. However, `fsdb` offers many commands to convert a desired inode, directory entry, block, superblock and so forth to a byte address. Once the address has been calculated, `fsdb` will record the result in dot (`.`).

Several global values are maintained by `fsdb`:

- the current base (referred to as `base`),

- the current address (referred to as `dot`),
- the current inode (referred to as `inode`),
- the current count (referred to as `count`),
- and the current type (referred to as `type`).

Most commands use the preset value of `dot` in their execution. For example,

```
> 2:inode
```

will first set the value of `dot` to 2, ':', will alert the start of a command, and the `inode` command will set `inode` to 2. A count is specified after a ','. Once set, `count` will remain at this value until a new command is encountered which will then reset the value back to 1 (the default). So, if

```
> 2000,400/X
```

is typed, 400 hex longs are listed from 2000, and when completed, the value of `dot` will be `2000 + 400 * sizeof (long)`. If a RETURN is then typed, the output routine will use the current values of `dot`, `count`, and `type` and display 400 more hex longs. A '*' will cause the entire block to be displayed.

End of fragment, block and file are maintained by `fsdb`. When displaying data as fragments or blocks, an error message will be displayed when the end of fragment or block is reached. When displaying data using the `db`, `ib`, `directory`, or `file` commands an error message is displayed if the end of file is reached. This is mainly needed to avoid passing the end of a directory or file and getting unknown and unwanted results.

An example showing several commands and the use of RETURN would be:

```
> 2:ino; 0:dir?d
      or
> 2:ino; 0:db:block?d
```

The two examples are synonymous for getting to the first directory entry of the root of the file system. Once there, any subsequent RETURN (or +, -) will advance to subsequent entries. Note that

```
> 2:inode; :ls
      or
> :ls /
```

is again synonymous.

Expressions The symbols recognized by `fsdb` are:

RETURN update the value of `dot` by the current value of `type` and display using the current value of `count`.

`#`	numeric expressions may be composed of +, -, *, and % operators (evaluated left to right) and may use parentheses. Once evaluated, the value of `dot` is updated.
`,` *count*	count indicator. The global value of `count` will be updated to `count`. The value of `count` will remain until a new command is run. A count specifier of '*' will attempt to show a *blocks's* worth of information. The default for `count` is 1.
`?`*f*	display in structured style with format specifier *f*. See `Formatted`Output`.`
`/`*f*	display in unstructured style with format specifier *f* See `Formatted`Output`.`
`.`	the value of `dot`.
`+`*e*	increment the value of `dot` by the expression *e*. The amount actually incremented is dependent on the size of `type`: `dot = dot + e * sizeof (type)` The default for *e* is `1`.
`-`*e*	decrement the value of `dot` by the expression *e*. See `+`.
`*`*e*	multiply the value of `dot` by the expression *e*. Multiplication and division don't use `type`. In the above calculation of `dot`, consider the `sizeof(type)` to be `1`.
`%`*e*	divide the value of `dot` by the expression *e*. See `*`.
`<` *name*	restore an address saved in register *name*. *name* must be a single letter or digit.
`>` *name*	save an address in register *name*. *name* must be a single letter or digit.
`=`*f*	display indicator. If *f* is a legitimate format specifier. then the value of `dot` is displayed using the format specifier *f*. See `Formatted`Output. Otherwise, assignment is assumed See `=`.
`= [`*s*`] [`*e*`]`	assignment indicator. The address pointed to by `dot` has its contents changed to the value of the expression *e* or to the ASCII representation of the quoted ('') string *s*. This may be useful for changing directory names or ASCII file information.
`=+` *e*	incremental assignment. The address pointed to by `dot` has its contents incremented by expression *e*.
`=-` *e*	decremental assignment. The address pointed to by `dot` has its contents decremented by expression *e*.

Commands A command must be prefixed by a ':' character. Only enough letters of the command to uniquely distinguish it are needed. Multiple commands may be entered on one line by separating them by a SPACE, TAB or ';'.

In order to view a potentially unmounted disk in a reasonable manner, `fsdb` offers the `cd`, `pwd`, `ls` and `find` commands. The functionality of these commands substantially matches those of its UNIX counterparts. See individual commands for details. The '*', '?', and '[-]' wild card characters are available.

`base=b`	display or set base. As stated above, all input and output is governed by the current `base`. If the `=b` is omitted, the current `base` is displayed. Otherwise, the current `base` is set to *b*. Note that this is interpreted using the old value of `base`, so to ensure correctness use the '0', '0t', or '0x' prefix when changing the `base`. The default for `base` is hexadecimal.
`block`	convert the value of `dot` to a block address.
`cd` *dir*	change the current directory to directory *dir*. The current values of `inode` and `dot` are also updated. If no *dir* is specified, then change directories to inode `2` ("/").
`cg`	convert the value of `dot` to a cylinder group.
`directory`	If the current `inode` is a directory, then the value of `dot` is converted to a directory slot offset in that directory and `dot` now points to this entry.
`file`	the value of `dot` is taken as a relative block count from the beginning of the file. The value of `dot` is updated to the first byte of this block.
`find` *dir* [`-name` *n*] [`-inum` *i*]	find files by name or i-number. `find` recursively searches directory `dir` and below for filenames whose i-number matches *i* or whose name matches pattern *n*. Note that only one of the two options (-name or -inum) may be used at one time. Also, the -print is not needed or accepted.
`fill=`*p*	fill an area of disk with pattern *p*. The area of disk is delimited by `dot` and `count`.
`fragment`	convert the value of *dot* to a fragment address. The only difference between the `fragment` command and the `block` command is the amount that is able to be displayed.
`inode`	convert the value of *dot* to an inode address. If successful, the current value of `inode` will be updated as well as the value of *dot*. As a convenient shorthand, if ':inode' appears at the beginning of the line, the value of *dot* is set to the current `inode` and that inode is displayed in inode format.
`log_chk`	run through the valid log entries without printing any information and verify the layout.

`log_delta`	count the number of deltas into the log, using the value of dot as an offset into the log. No checking is done to make sure that offset is within the head/tail offsets.
`log_head`	display the header information about the file system logging. This shows the block allocation for the log and the data structures on the disk.
`log_otodb`	return the physical disk block number, using the value of dot as an offset into the log.
`log_show`	display all deltas between the beginning of the log (BOL) and the end of the log (EOL).
`ls`	[`-R`] [`-l`] *pat1 pat2* . . . list directories or files. If no file is specified, the current directory is assumed. Either or both of the options may be used (but, if used, *must* be specified before the filename specifiers). Also, as stated above, wild card characters are available and multiple arguments may be given. The long listing shows only the i-number and the name; use the `inode` command with '?i' to get more information.
`override`	toggle the value of override. Some error conditions may be overriden if override is toggled on.
`prompt` *p*	change the `fsdb` prompt to *p*. *p* must be surrounded by (")s.
`pwd`	display the current working directory.
`quit`	quit `fsdb`.
`sb`	the value of *dot* is taken as a cylinder group number and then converted to the address of the superblock in that cylinder group. As a shorthand, ':sb' at the beginning of a line will set the value of *dot* to *the* superblock and display it in superblock format.
`shadow`	if the current inode is a shadow inode, then the value of *dot* is set to the beginning of the shadow inode data.
`!`	escape to shell

Inode Commands

In addition to the above commands, there are several commands that deal with inode fields and operate directly on the current `inode` (they still require the ':'). They may be used to more easily display or change the particular fields. The value of *dot* is only used by the '`:db`' and '`:ib`' commands. Upon completion of the command, the value of *dot* is changed to point to that particular field. For example,

```
> :ln=+1
```

would increment the link count of the current `inode` and set the value of *dot* to the address of the link count field.

`at` access time.

`bs` block size.

`ct` creation time.

`db` use the current value of *dot* as a direct block index, where direct blocks number from 0 - 11. In order to display the block itself, you need to 'pipe' this result into the `block` or `fragment` command. For example,

```
> 1:db:block,20/X
```

would get the contents of data block field 1 from the inode and convert it to a block address. 20 longs are then displayed in hexadecimal. See `Formatted`Output.

`gid` group id.

`ib` use the current value of *dot* as an indirect block index where indirect blocks number from 0 - 2. This will only get the indirect block itself (the block containing the pointers to the actual blocks). Use the `file` command and start at block 12 to get to the actual blocks.

`ln` link count.

`mt` modification time.

`md` mode.

`maj` major device number.

`min` minor device number.

`nm` although listed here, this command actually operates on the directory name field. Once poised at the desired directory entry (using the *directory* command), this command will allow you to change or display the directory name. For example,

```
> 7:dir:nm="foo"
```

will get the 7th directory entry of the current `inode` and change its name to foo. Note that names cannot be made larger than the field is set up for. If an attempt is made, the string is truncated to fit and a warning message to this effect is displayed.

`si` shadow inode.

`sz` file size.

`uid` user id.

Formatted Output There are two styles and many format types. The two styles are structured and unstructured. Structured output is used to display inodes, directories, superblocks and the like. Unstructured displays raw data. The following shows the different ways of displaying:

`?`

- `c` display as cylinder groups
- `i` display as inodes
- `d` display as directories
- `s` display as superblocks
- `S` display as shadow inode data

`/`

- `b` display as bytes
- `c` display as characters
- `o O` display as octal shorts or longs
- `d D` display as decimal shorts or longs
- `x X` display as hexadecimal shorts or longs

The format specifier immediately follows the '/' or '?' character. The values displayed by '/b' and all '?' formats are displayed in the current `base`. Also, `type` is appropriately updated upon completion.

Examples

EXAMPLE 1 Displaying in Decimal

The following command displays `2010` in decimal (use of `fsdb` as a calculator for complex arithmetic):

```
> 2000+400%(20+20)=D
```

EXAMPLE 2 Displaying an i-number in Inode Format

The following command displays i-number `386` in an inode format. This now becomes the current `inode`:

```
> 386:ino?i
```

EXAMPLE 3 Changing the Link Count

The following command changes the link count for the current `inode` to `4`:

```
> :ln=4
```

EXAMPLE 4 Incrementing the Link Count

The following command increments the link count by 1:

```
> :ln=+1
```

EXAMPLE 5 Displaying the Creation Time

The following command displays the creation time as a hexadecimal long:

```
> :ct=X
```

EXAMPLE 6 Displaying the Modification Time

The following command displays the modification time in time format:

```
> :mt=t
```

EXAMPLE 7 Displaying in ASCII

The following command displays in ASCII, block zero of the file associated with the current `inode`:

```
> 0:file/c
```

EXAMPLE 8 Displaying the First Block's Worth of Directorty Entries

The following command displays the first block's worth of directory entries for the root inode of this file system. It will stop prematurely if the EOF is reached:

```
> 2:ino,*?d
```

EXAMPLE 9 Displaying Changes to the Current Inode

The following command displays changes the current inode to that associated with the 5th directory entry (numbered from zero) of the current `inode`. The first logical block of the file is then displayed in ASCII:

```
> 5:dir:inode; 0:file,*/c
```

EXAMPLE 10 Displaying the Superblock

The following command displays the superblock of this file system:

```
> :sb
```

EXAMPLE 11 Displaying the Cylinder Group

The following command displays cylinder group information and summary for cylinder group 1:

```
> 1:cg?c
```

EXAMPLE 12 Changing the i-number

The following command changes the i-number for the seventh directory slot in the root directory to 3:

```
> 2:inode; 7:dir=3
```

EXAMPLE 13 Displaying as Directory Entries

The following command displays the third block of the current `inode` as directory entries:

```
> 2:db:block,*?d
```

EXAMPLE 14 Changing the Name Field

The following command changes the name field in the directory slot to *name*:

```
> 7:dir:nm="name"
```

EXAMPLE 15 Getting and Filling Elements

The following command gets fragment `3c3` and fill `20` `type` elements with `0x20`:

```
> 3c3:fragment,20:fill=0x20
```

EXAMPLE 16 Setting the Contents of an Address

The following command sets the contents of address `2050` to `0xffffffff`. `0xffffffff` may be truncated depending on the current `type`:

```
> 2050=0xffff
```

EXAMPLE 17 Placing ASCII

The following command places the ASCII for the string at `1c92434`:

```
> 1c92434="this is some text"
```

EXAMPLE 18 Displaying Shadow Inode Data

The following command displays all of the shadow inode data in the shadow inode associated with the root inode of this file system:

```
> 2:ino:si:ino;0:shadow,*?S
```

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also clri(1M), fsck_ufs(1M), dir_ufs(4), attributes(5), ufs(7FS)

Warnings Since `fsdb` reads the disk raw, extreme caution is advised in determining its availability of `fsdb` on the system. Suggested permissions are 600 and owned by bin.

Notes The old command line syntax for clearing i-nodes using the ufs-specific `'-z i-number'` option is still supported by the new debugger, though it is obsolete and will be removed in a future release. Use of this flag will result in correct operation, but an error message will be printed warning of the impending obsolesence of this option to the command. The equivalent functionality is available using the more flexible clri(1M) command.

Name fsflush, kmem_task, pageout, sched, vmtasks – process not intended for user interaction

Synopsis *process_name arguments*

Description The Oracle Solaris operating system relies on a number of "behind the scenes" processes that are Private interfaces (see `attributes(5)`) and that are not intended for customer interaction. The user has no responsibility for starting, stopping, or, in any other way, manipulating these processes. Using tools such as `ps(1)`, these processes are visible to users. There presence or absence might be of consequence to Oracle service personnel. Their presence in `ps` output is of no consequence to a user.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	Various packages
Interface Stability	Private

See Also `ps(1)`, `attributes(5)`

Name fsirand – install random inode generation numbers

Synopsis

```
fsirand [-p] special
```

Description `fsirand` installs random inode generation numbers on all the inodes on device *special*, and also installs a file system ID in the superblock. This helps increase the security of file systems exported by NFS.

`fsirand` must be used only on an unmounted file system that has been checked with `fsck(1M)` The only exception is that it can be used on the root file system in single-user mode, if the system is immediately re-booted afterwards.

Options

`-p` Print out the generation numbers for all the inodes, but do not change the generation numbers.

Usage See `largefile(5)` for the description of the behavior of `fsirand` when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also `fsck(1M)`, `attributes(5)`, `largefile(5)`

Name fssnap – create temporary snapshots of a file system

Synopsis

```
fssnap [-F FSType] [-V] -o special_options /mount/point

fssnap  -d [-F  FSType] [-V] /mount/point | dev

fssnap -i [-F FSType] [-V] [-o special_options]
     [/mount/point | dev]
```

Description The `fssnap` command creates a stable, read-only snapshot of a file system when given either an active mount point or a special device containing a mounted file system, as in the first form of the synopsis. A snapshot is a temporary image of a file system intended for backup operations.

While the snapshot file system is stable and consistent, an application updating files when the snapshot is created might leave these files in an internally inconsistent, truncated, or otherwise unusable state. In such a case, the snapshot will contain these partially written or corrupted files. It is a good idea to ensure active applications are suspended or checkpointed and their associated files are also consistent during snapshot creation.

File access times are not updated while the snapshot is being created.

A path to the virtual device that contains this snapshot is printed to standard output when a snapshot is created.

Options The following options are supported:

`-d`	Deletes the snapshot associated with the given file system.
`-F` *FSType*	Specifies the file system type to be used. The `FSType` should either be specified here or be determined by matching the block special device with an entry in the `/etc/vfstab` table, or by consulting `/etc/default/fs`.
`-i`	Displays the state of any given `FSType` snapshot. If a mount-point or device is not given, a list of all snapshots on the system is displayed. When a mount-point or device is specified, detailed information is provided for the specified file system snapshot by default. The format and meaning of this information is file-system dependent. See the `FSType`-specific `fssnap` man page for details.
`-o` *special_options*	See the `FSType`-specific man page for `fssnap`.
`-V`	Echoes the complete command line, but does not execute the command.

Operands The following operands are supported:

/mount/point	The directory where the file system resides.

Examples See FSType-specific man pages for examples.

Exit Status The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

Files

`/etc/vfstab` Specifies file system type.

`/etc/default/fs` Specifies the default local file system type.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also `fssnap_ufs(1M)`, `attributes(5)`

Notes This command might not be supported for all FSTypes.

Name fssnap_ufs – create a temporary snapshot of a UFS file system

Synopsis

```
fssnap [-F ufs] [-V] -o backing-store=path,
     [specific-options] /mount/point

fssnap -d [-F ufs] [-V] /mount/point | dev

fssnap -i [-F ufs] [-V] [-o specific-options] /mount/point | dev
```

Description The `fssnap` command queries, creates, or deletes a temporary snapshot of a UFS file system. A snapshot is a point-in-time image of a file system that provides a stable and unchanging device interface for backups.

When creating a file system snapshot, you must specify the file system to be captured and the backing-store file. The backing-store file(s) are where the snapshot subsystem saves old file system data before it is overwritten. Beyond the first backing-store file, `fssnap` automatically creates additional backing-store files on an as-needed basis.

The number and size of the backing store files varies with the amount of activity in the file system. The destination path must have enough free space to hold the backing-store file(s). This location must be different from the file system that is being captured in a snapshot. The backing-store file(s) can reside on any type of file system, including another UFS file system or an NFS–mounted file system.

Options The following options are supported:

`-d`
Deletes the snapshot associated with the given file system.

`-i`
Displays the state of one or all UFS snapshots. If a mount-point or device is not specified, a list of all snapshots on the system is displayed. When a mount-point or device is specified, detailed information is provided for the specified file system snapshot by default.

Use the `-o` options with the `-i` option to specify what snapshot information is displayed. Since this feature is provided primarily for use in scripts and on the command line, no labels are displayed for the data. Sizes are all in bytes, and the output is not internationalized or localized. The information is displayed on one line per option. Unrecognized options display a single `?` on the line. One line per option guarantees that there are the same number of lines as options specified and there is a one–to-one correspondence between an output line and an option.

The following `-o` options display specific information for a given snapshot. See the EXAMPLES section for examples of how to use these options.

`snapnumber`
Display the snapshot number.

`blockdevname`
Display the block device path.

`rawdevname`
Display the raw device path.

`mountpoint`
Display the mount point of the master file system.

`state`
Display the state of the snapshot device.

`backing-store`
Display the location of the first backing-store file for this snapshot. If there are multiple backing-store files, subsequent files have the same name as the first file, with the suffixes `.2`, `.3`, and so forth.

`backing-store-len`
Display the sum of the sizes of the backing-store files.

`maxsize`
Display the `maxsize` value specified for the backing-store file(s).

`createtime`
Display the time that the snapshot was created.

`chunksize`
Display the copy-on-write granularity.

`-o` *specific-options*
Without `-d` or `-i`, the default action is to create a snapshot. Specify the following options when creating a snapshot. All of these options are discretionary, except for the backing-store file, which is required.

`backing-store=`*path*
Uses *path* in the creation of the backing-store file(s). *path* must not reside on the file system that is being captured in a snapshot and must not be the name of an existing file. If *path* is a directory, then a backing-store file is created within it using a name that is generated automatically. If *path* is not a directory and does not already exist, then a backing-store file with that name is created. If more than one backing-store file is required, `fssnap` creates subsequent files automatically. The second and subsequent files have the same name as the first file, with suffixes of `.2`, `.3`, and so forth.

This option can be abbreviated as `bf=`*path* or `bs=`*path*.

`unlink`
Unlinks the backing-store file after the snapshot is created. This option specifies that the backing-store file does not need to be removed manually when the snapshot is deleted. This might make administration more difficult since the file is not visible in the file system. If this option is not specified, the backing-store files should be removed manually after the snapshot is deleted.

`chunksize=`*n* [`k`,`m`,`g`]
Uses *n* for the chunk size. Chunk size is the granularity of the data that is sent to the backing store.

Specify `chunksize` in the following units: `k` for kilobytes, `m` for megabytes, or `g` for gigabytes. By default, chunk size is four times the block size of the file system (typically `32k`).

`maxsize=`*n*[`k`,`m`,`g`]
Does not allow the sum of the sizes of the backing-store file(s) to exceed *n*, where *n* is the unit specified. The snapshot is deleted automatically when the sum of the sizes of the backing-store file(s) exceeds `maxsize`.

Specify `maxsize` in the following units: `k` for kilobytes, `m` for megabytes, or `g` for gigabytes.

`raw`
Displays to standard output the name of the raw device instead of the block device when a snapshot is created. The block device is printed by default (when `raw` is not specified). This option makes it easier to embed `fssnap` commands in the command line for commands that require the raw device instead. Both devices are always created. This option affects only the output.

Operands The following operands are supported:

mount-point
The directory where the file system resides.

special
The physical device for the file system, such as `/dev/dsk/c0t0d0s7`.

Examples **EXAMPLE 1** Creating a Snapshot of a File System

The following example creates a snapshot of a file system. The block special device created for the snapshot is `/dev/fssnap/0`.

```
# fssnap -F ufs -o backing-store=/var/tmp /export/home
/dev/fssnap/0
```

EXAMPLE 2 Backing Up a File System Snapshot Without Having To Unmount the File System

The following example backs up a file system snapshot without having to unmount the file system. Since `ufsdump` requires the path to a raw device, the `raw` option is used. The `/export/home` file system snapshot is removed in the second command.

```
# ufsdump 0uf /dev/rmt/0 ‘fssnap -F ufs
      -o raw,bs=/export/snap /export/home‘
<output from ufsdump>
# fssnap -F ufs -d /export/home
```

EXAMPLE 3 Backing Up a File System

When backing up a file system, do not let the backing-store file(s) exceed `400` Mbytes. The second command removes the `/export/home` file system snapshot.

```
# ufsdump 0uf /dev/rmt/0 ‘fssnap -F ufs
      -o maxsize=400m,backing-store=/export/snap,raw
      /export/home‘
# fssnap -F ufs -d /export/home
```

EXAMPLE 4 Performing an Incremental Dump of a Snapshot

The following example uses `ufsdump` to back up a snapshot of `/var`. Note the use of the `N` option to `ufsdump`, which writes the name of the device being dumped, rather than the name of the snapshot device, to `/etc/dumpdates` file. See `ufsdump(1M)` for details on the `N` flag.

```
# ufsdump lfNu /dev/rmt/0 /dev/rdsk/c0t3d0s2 ‘fssnap -F ufs
-o raw,bs=/export/scratch,unlink /var‘
```

EXAMPLE 5 Finding Out What Snapshots Currently Exist

The following command displays the currently existing snapshots.

```
# fssnap -i
0  /src
1  /export/home
<output continues>
```

EXAMPLE 6 Mounting a File System Snapshot

The following example creates a file system snapshot. After you create a file system snapshot, mount it on `/tmp/mount` for temporary read-only access.

```
# fssnap -F ufs -o backing-store=/nfs/server/scratch /export/home
/dev/fssnap/1
# mkdir /tmp/mount
# mount -F ufs -o ro /dev/fssnap/1 /tmp/mount
```

EXAMPLE 7 Creating a File System Snapshot and Unlinking the Backing-store File

The following example creates a file system snapshot and unlinks the backing-store file. After creating a file system snapshot and unlinking the backing-store file, check the state of the snapshot.

```
# fssnap -o bs=/scratch,unlink /src
/dev/fssnap/0
# fssnap -i /src
Snapshot number              : 0
Block Device                 : /dev/fssnap/0
Raw Device                   : /dev/rfssnap/0
Mount point                  : /src
Device state                 : active
```

EXAMPLE 7 Creating a File System Snapshot and Unlinking the Backing-store File *(Continued)*

```
Backing store path            : /scratch/snapshot2 <UNLINKED>
Backing store size            : 192 KB
Maximum backing store size    : Unlimited
Snapshot create time          : Sat May 06 10:55:11 2000
Copy-on-write granularity     : 32 KB
```

EXAMPLE 8 Displaying the Size and Location of the Backing-store File(s) and the Creation Time for the Snapshot

The following example displays the size of the backing-store file(s) in bytes, the location of the backing store, and the creation time for the snapshot of the `/test` file system.

```
# fssnap -i -o backing-store-len,backing-store,createtime /test
196608
/snapshot2
Sat May 6 10:55:11 2000
```

Note that if there are multiple backing-store files stored in `/snapshot2`, they will have names of the form *file* (for the first file), *file*.1, *file*.2, and so forth.

Exit Status

The following exit values are returned:

0
Successful completion.

>**0**
An error occurred.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

The script-readable output mode is a stable interface that can be added to, but will not change. All other interfaces are subject to change.

See Also

`mlock(3C)`, `attributes(5)`

See the `ntpd` man page, delivered in the `SUNWntpu` package (not a SunOS man page).

Notes

The `fssnap` device files should be treated like a regular disk block or character device.

The association between a file system and the snapshot is lost when the snapshot is deleted or the system reboots. Snapshot persistence across reboots is not currently supported.

To avoid unnecessary performance impacts, perform the snapshot and system backup when the system is least active.

It is not possible to perform a snapshot of a file system if any of the following conditions are true:

- The file system is in use by system accounting
- The file system contains a local swap file
- The file system is used as backing store by an application that uses `mlock(3C)` to lock its pages. Typically, these are real time applications, such as `ntpd` (delivered in the `SUNWntpu` package).

These conditions result in `fssnap` being unable to write lock the file system prior to performing the snapshot.

Name fsstat – report file system statistics

Synopsis

```
fsstat [-A|a|f|i|n|v|Z] [-T d|u] [[-z zone]...]
     {-F | {fstype|path}...} [interval [count]]
```

Description `fsstat` reports kernel file operation activity by the file system type (*fstype*) or by the path name, which is converted to a mount point. The first set of lines of output reports all activity since:

- The file system module was loaded (in the case of *fstype*)
- The file system was mounted (in the case of mount point)

Statistics are gathered at the file system independent layer at both the *fstype* and the mount point levels. However, not all file system types are represented in the gathering of statistics. (See the NOTES section of this man page.)

The output of `fsstat` is dependent on the mode (option) requested. All statistic fields are displayed using "smart numbers" which automatically scale the units in a human readable form that fits in a maximum of 5 characters. For example:

100	is displayed as 100
2048	is displayed as 2K
3000000	is displayed as 2.86M

The unit modifiers are: K (Kbyte), M (Mbyte), G (Gbyte), T (terabyte), P (petabyte), and E (exabyte).

During the execution of `fsstat`, the state of the system can change. If relevant, a state change message is included in the `fsstat` output in one of the following forms:

```
<<mount point no longer available: {path}>>
<<file system module no longer loaded: {fstype}>>
<<zone no longer active: {zonename}>>
<<zone now active: {zonename}>>
```

After the state change messages are displayed, `fsstat` continues to display the statistics as directed. If all of the *fstypes* and mount points that `fsstat` was reporting on are no longer available, then `fsstat` exits.

The user is required to specify the `-F` option (all available file system types) or a list of one or more *fstypes* and/or mount points.

The default report shows general file system activity. This display combines similar operations into general categories as follows:

new file	Number of creation operations for file system objects (for example, files, directories, symlinks, etc.)
name remov	Number of name removal operations

name chng	Number of name change operations
attr get	Number of object attribute retrieval operations
attr set	Number of object attribute change operations
lookup ops	Number of object lookup operations
rddir ops	Number of read directory operations
read ops	Number of data read operations
read bytes	Bytes transferred by data read operations
write ops	Number of data write operations
write bytes	Bytes transferred by data write operations

The entity being reported on (*fstype* or mount point) is displayed in the last column.

Options The following options are supported:

`-A` Report aggregate activity for the specified `fstypes` across all zones. This is the default behavior if neither `-z` nor `-Z` are specified.

When used in conjunction with `-z` or `-Z`, the `-A` option will additionally report on a separate line the aggregate for the specified `fstypes` across all zones.

`-a` Report the activity for kernel attribute operations. The following statistics are reported:

getattr	Number of file attribute retrieval calls
setattr	Number of file attribute modification calls
getsec	Number of file security attribute retrieval calls
setsec	Number of file security attribute modification calls

The entity being reported on (*fstype* or mount point) is displayed in the last column.

`-F` Report on all available file system types.

`-f` Report the full activity for all kernel file operations. Each file operation is listed in the left column. The following statistics are reported for each operation:

#ops	Number of calls for this operation
bytes	Average transfer size in bytes (only applies to read, write, readdir)

The entity being reported on (*fstype* or mount point) is displayed in the first row.

`-i` Reports the activity for kernel I/O operations. The following statistics are reported:

read ops	Number of data read calls
read bytes	Number of bytes read
write ops	Number of data write calls
write bytes	Number of bytes written
rddir ops	Number of read directory calls
rddir bytes	Number of bytes read by reading directories
rwlock ops	Number of internal file system lock operations
rwulock ops	Number of internal file system unlock operations

The entity being reported on (*fstype* or mount point) is displayed in the last column.

`-n` Reports the activity for kernel naming operations. The following statistics are reported:

lookup	Number of file name retrieval calls
creat	Number of file creation calls
remov	Number of file remove calls
link	Number of link calls
renam	Number of file renaming calls
mkdir	Number of directory creation calls
rmdir	Number of directory removal calls
rddir	Number of directory read calls
symlink	Number of symlink creation calls
rdlink	Number of symlink read calls

The entity being reported on (*fstype* or mount point) is displayed in the last column.

`-v` Reports the activity for calls to the virtual memory operations. The following statistics are reported.

map	Number of calls mapping a file

addmap	Number of calls setting additional mapping to a mapped file
delmap	Number of calls deleting mapping to a file
getpag	Number of calls retrieving a page of data from a file
putpag	Number of calls writing a page of data to a file
pagio	Number of calls to transfer pages in file system swap files

The entity being reported on (fstype or mount point) is displayed in the last column.

`-T` *u*|*d* — Display a time stamp.

Specify *u* for a printed representation of the internal representation of time (see `time(2)`) Specify *d* for the standard date format. (See `date(1)`). The time stamp is only used when an interval is set.

`-Z` — Report per-zone activity for each zone.

Specifying `-Z` has no effect if used in conjunction with `-z`.

`-z` [*zonename*] — Report on activity in the specified zone. Multiple `-z` options can be specified to monitor multiple zones. If `-z` is specified, the user is notified of state changes only for explicitly specified zones.

Operands The following operands are supported:

count	Display only *count* reports.
fstype	Explicitly specify the file system type(s) to be reported. The file system module must be loaded.
interval	Report once each *interval* seconds.
path	Specify the path(s) of the mount point(s) to be reported. If path is not a mount point, the mount point containing path will be determined and displayed in the output.

If no *interval* and no *count* are specified, a single report is printed and `fsstat` exits. If an *interval* is specified but no *count* is specified, `fsstat` prints reports every *interval* seconds indefinitely until the command is interrupted.

Examples In some examples, `fsstat` output exceeds the width of 80–character displays.

EXAMPLE 1 Displaying General Activity

The following example shows general activity for all file system types.

```
$ fsstat -F
 new  name   name  attr   attr lookup rddir  read read  write write
```

EXAMPLE 1 Displaying General Activity *(Continued)*

```
 file remov  chng   get    set    ops   ops   ops bytes   ops bytes
  313K  214K 38.5K 2.16M 56.2K  8.36M 52.8K 19.7M 39.9G 18.8M 39.1G ufs
     0     0     0 2.95K     0  3.81K   282 2.52K  466K     0     0 proc
     0     0     0     0     0      0     0     0     0     0     0 nfs
    10     8     2    86     9     98    15   413  103M 8.43K 1.05G zfs
    13    14     4    98    16    125    10 1.01K  258M 15.9K  127M lofs
8.73K 3.29K 5.25K 55.3K    37  1.20M    44 37.9K 38.3M 47.2K 35.9M tmpfs
     0     0     0 4.93K     0      0     0 1.08K  913K     0     0 mntfs
     3     2     1   503     3    897    13   122 25.8K   128  272K nfs3
    10     8     0   615    10  10.1K    18    61 45.6K   292 2.26M nfs4
```

EXAMPLE 2 Displaying Naming Activity

The following example shows the naming activity for ufs, nfs, nfs3, nfs4, and tmpfs:

```
$ fsstat -n ufs nfs nfs3 nfs4 tmpfs
lookup creat remov  link renam mkdir rmdir rddir symlnk rdlnk
3.57M  3.10K   586     6    24   115   100 30.2K      5  330K ufs
    0      0     0     0     0     0     0     0      0     0 nfs
18.3K      3     5     0     0     0     0 1.03K      2   346 nfs3
  535      0     0     0     0     0     0    46      0     4 nfs4
  146     24    15     0     0     4     0     4      0     0 tmpfs
```

EXAMPLE 3 Displaying Attribute Activity

The following example shows the attribute activity for the FS type ufs and the mounted file systems "/" and "/export/home" every three seconds for every third iteration:

```
# fsstat -a ufs / /export/home 3 3
getattr setattr getsec setsec
  378K    91.9K  11.8K      0 ufs
  367K    82.3K  11.6K      0 /
 11.3K     9.6K    198      0 /export/home
 4.97K    2.27K    163      0 ufs
 3.94K    1.36K    162      0 /
 1.03K      927      1      0 /export/home
 2.30K    1.06K     73      0 ufs
 1.95K      766     71      0 /
   361      317      2      0 /export/home
 2.33K    1.06K     78      0 ufs
 1.64K      451     77      0 /
   711      631      1      0 /export/home
```

EXAMPLE 4 Displaying File Operation Statistics

The following example shows the statistics for each file operation for "/" (using the `-f` option):

EXAMPLE 4 Displaying File Operation Statistics *(Continued)*

```
$ fsstat -f /
Mountpoint: /
 operation  #ops  bytes
      open 8.54K
     close  9.8K
      read 43.6K  65.9M
     write 1.57K  2.99M
     ioctl 2.06K
     setfl     4
   getattr 40.3K
   setattr    38
    access 9.19K
    lookup  203K
    create   595
    remove    56
      link     0
    rename     9
     mkdir    19
     rmdir     0
   readdir 2.02K  2.27M
   symlink     4
  readlink 8.31K
     fsync   199
  inactive 2.96K
       fid     0
    rwlock 47.2K
  rwunlock 47.2K
      seek 29.1K
       cmp 42.9K
    frlock 4.45K
     space     8
    realvp 3.25K
   getpage  104K
   putpage 2.69K
       map 13.2K
    addmap 34.4K
    delmap 33.4K
      poll   287
      dump     0
  pathconf    54
    pageio     0
   dumpctl     0
   dispose 23.8K
getsecattr   697
setsecattr     0
```

EXAMPLE 4 Displaying File Operation Statistics *(Continued)*

```
   shrlock     0
   vnevent     0
```

EXAMPLE 5 Displaying per-Zone Statistics for All Zones

The following example shows per-zone statistics for each zone on the system, as well as a system-wide aggregate for `fstypes` `tmpfs` and `zfs`.

```
$ fsstat -A -Z tmpfs zfs
new  name   name  attr  attr lookup rddir  read read  write write
file remov  chng   get   set    ops   ops   ops bytes   ops bytes
125K  116K 8.92K  846K 1.25K  1.36M   252 1013K  913M 1.52M 1.55G tmpfs
98.9K 89.8K 8.87K 600K 1.19K  1.33M   226  394K  253M 1.04M 1.07G tmpfs:global
2.49K 2.42K 32    20.5K  45   3.82K    26 56.8K 85.8M 43.9K 69.5M tmpfs:zone1
23.3K 23.3K 13    226K   13   24.1K     0  562K  574M  452K  425M tmpfs:zone2
82.7K 232K  77.6K 4.72M 73.6K 22.7M  464K 2.88M 6.17G  828K 8.19G zfs
82.1K 231K  77.3K 4.46M 73.5K 21.8M  444K 2.53M 5.71G  809K 8.12G zfs:global
102    88   28    83.3K    68  326K 3.16K  238K  307M 10.5K 54.2M zfs:zone1
499   204   255   179K     34  599K 17.4K  125K  163M 8.85K 21.8M zfs:zone2
```

EXAMPLE 6 Displaying per-Zone Statistics for Specific Zones

The following example shows per-zone statistics for zones `zone1` and `zone2`, as well as a system-wide aggregate, for `fstypes` `tmpfs` and `zfs`.

```
$ fsstat -A -Z zone1 -z zone2 tmpfs zfs
new  name   name  attr  attr lookup rddir  read read  write write
file remov  chng   get   set    ops   ops   ops bytes   ops bytes
125K  116K 8.92K  846K 1.25K  1.36M   252 1013K  913M 1.52M 1.55G tmpfs
2.49K 2.42K  32  20.5K   45   3.82K    26 56.8K 85.8M 43.9K 69.5M tmpfs:zone1
23.3K 23.3K  13   226K    13  24.1K     0  562K  574M  452K  425M tmpfs:zone2
82.7K 232K 77.6K 4.72M 73.6K  22.7M  464K 2.88M 6.17G  828K 8.19G zfs
102    88    28  83.3K   68    326K 3.16K  238K  307M 10.5K 54.2M zfs:zone1
499   204   255  179K    34    599K 17.4K  125K  163M 8.85K 21.8M zfs:zone2
```

EXAMPLE 7 Example 7 Displaying Global Zone-Only Statistics

The following example shows only the global zone's activity for all `fstypes`.

```
$ fsstat -z global -F
new  name   name  attr  attr lookup rddir  read read  write write
file remov  chng   get   set    ops   ops   ops bytes   ops bytes
0     0     0     0     0      0     0     0     0     0     0 ufs:global
0     0     0     171K  0   289K   2.05K  628K  262M  230K 10.6M proc:global
0     0     0     0     0      0     0     0     0     0     0 nfs:global
82.1K 231K 77.3K 4.46M 73.5K  21.8M 444K 2.53M 5.71G  809K 8.12G zfs:global
0     0     0     357K  0      0     0     0     0     0     0 lofs:global
98.9K 89.8K 8.87K 600K 1.19K  1.33M 226  394K  253M 1.04M 1.07G tmpfs:global
```

EXAMPLE 7 Example 7 Displaying Global Zone-Only Statistics *(Continued)*

```
0     0     0 15.4K     0      0     0   7.44K 1.03M   0     0 mntfs:global
0     0     0 9.34K     0  9.24K     0     0     0     0     0 autofs:global
0     0     0     0     0      0     0     0     0     0     0 nfs3:global
2     3     0 9.16K    12   104K     4    46   179K   12 43.0K nfs4:global
```

Environment Variables See `environ(5)` for descriptions of the following environment variables that affect the execution of `fsstat`: `LANG`, `LC_ALL`, `LC_CTYPE`, `LC_MESSAGES`, `LC_TIME`, and `NLSPATH`.

Exit Status The following exit values are returned:

`0` Successful completion.

`1` A fatal error occurred. A fatal error could be a failed system call or another internal error.

`2` Invalid command-line options were specified.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os
CSI	Enabled
Interface Stability	See below.

The command-line options are Uncommitted. The human-readable output is not considered an interface.

See Also `date(1)`, `time(2)`, `attributes(5)`

Notes All display options (`-a`, `-f`, `-i`, `-n`, `-v`) are mutually exclusive. Entering more than one of these options will result in an error.

The *fstype* and *path* operands must appear after the option, but before the *interval* or *count* on the command line. For example, "`fsstat -a` *fstype interval*". Preference is given to *fstype* so that if a user wishes to see the statistics for a directory that has the same name as an *fstype* (for example, ufs), then the path must be specified unambiguously (for example, ./ufs). Similarly, in order to define a file with a numeric name (for example, "10") from an interval or count operand, the name should be prefixed accordingly (for example, ./10).

When an interval is used, headers repeat after more than 12 lines of statistics have been displayed and the set of lines to be displayed in the current interval have completed.

Statistics are not displayed for all pseudo-filesystems. The output displayed with the `-F` option shows which of the loaded filesystem types are supported.

Unbundled file systems may not be recognized by `fsstat`.

The command-line options are classified as Uncommitted and could change. The output is not considered to be an interface. The construction of higher level software tools depend on either the command-line options or the output of `fsstat` is not recommended.

Name fstyp – determine file system type

Synopsis

```
fstyp [-a | -v] special [:logical-drive]
```

Description `fstyp` allows the user to determine the file system type of unmounted file systems using heuristic programs.

An `fstyp` module for each file system type to be checked is executed; each of these modules applies an appropriate heuristic to determine whether the supplied *special* file is of the type for which it checks. If it is, the program prints on standard output the usual file system identifier for that type (for example, "ufs") and exits with a return code of `0`; if none of the modules succeed, the error message `unknown_fstyp (no matches)` is returned and the exit status is `1`. If more than one module succeeds, the error message `unknown_fstyp (multiple matches)` is returned and the exit status is `2`. Other errors are printed in the same format.

This command is unreliable and its results should not be used to make any decisions about subsequent use of a storage device or disk partition.

Options

`-a` Output all available file system attributes. If a file system has been successfully identified, and this option is specified, the *FSType* identifier is followed by one or more "name-value" pairs, one per line, in the format:

```
name: value
```

The following conventions are recognized for the file system attributes:

- String values are put in single quotes.
- Nested "name-value" list increases the indentation of its values by four whitespaces.
- For an array of "name-value" pairs, one array entry is printed per line, with the index following the name in square brackets.

For instance, in the following example, "*top_list*" is a "name-value" list, consisting of a string array "*string_array*" and a "name-value" list array "*list_array*". The second "list_array"element is an integer array "*int_array*" containing three elements.

```
top_string: 'String'
top_list:
    string_array[0]: 'S0'
    string_array[1]: 'S1'
    list_array[0]:
        int_one: 1
        string_two: 'S2'
    list_array[1]:
        int_array[0]: 1
        int_array[1]: 2
        int_array[2]: 3
```

In addition to the *FSType*-specific attributes, the following generic attributes may be present:

gen_clean Allowable values are “true” or “false”. If the value is “false”, the file system is damaged or was not unmounted cleanly and the `fsck(1M)` command must be run before this file system can be mounted.

gen_guid Globally unique identifier. This string uniquely identifies the file system.

gen_version A string describing the file system version.

gen_volume_label Volume label, a human-readable string used to either describe or identify the file system.

`-v` Produce verbose output. This is usually information about the file systems superblock and varies across different *FSTypes*. See `ufs(7FS)`, `mkfs_ufs(1M)`, and `tunefs(1M)` for details.

Usage See `largefile(5)` for the description of the behavior of `fstyp` when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also `fsck(1M)`, `mkfs_ufs(1M)`, `tunefs(1M)`, `attributes(5)`, `libfstyp(3LIB)`, `largefile(5)`, `hsfs(7FS)`, `ufs(7FS)`, `pcfs(7FS)`

Notes The use of heuristics implies that the result of `fstyp` is not guaranteed to be accurate.

This command is unreliable and its results should not be used to make any decisions about subsequent use of a storage device or disk partition.

Name fuser – identify users of files and devices

Synopsis

```
/usr/sbin/fuser [-c | -d | -f] [-nu] [-k | -s sig] files
     [ [-] [-c | -d | -f] [-nu] [-k | -s sig] files] ...
```

Description The `fuser` utility displays the process IDs of the processes that are using the *files* specified as arguments.

Each process ID is followed by a letter code. These letter codes are interpreted as follows. If the process is using the file as

`c` Indicates that the process is using the file as its current directory.

`m` Indicates that the process is using a file mapped with `mmap(2)`. See `mmap(2)` for details.

`n` Indicates that the process is holding a non-blocking mandatory lock on the file.

`o` Indicates that the process is using the file as an open file.

`r` Indicates that the process is using the file as its root directory.

`t` Indicates that the process is using the file as its text file.

`y` Indicates that the process is using the file as its controlling terminal.

For block special devices with mounted file systems, all processes using any file on that device are listed. For all types of files (text files, executables, directories, devices, and so forth), only the processes using that file are reported.

For all types of devices, `fuser` also displays any known kernel consumers that have the device open. Kernel consumers are displayed in one of the following formats:

[*module_name*]
[*module_name*,dev_path=*path*]
[*module_name*,dev=(*major*,*minor*)]
[*module_name*,dev=(*major*,*minor*),dev_path=*path*]

If more than one group of files are specified, the options may be respecified for each additional group of files. A lone dash cancels the options currently in force.

The process IDs are printed as a single line on the standard output, separated by spaces. All other output, including the single terminating newline, is written on standard error.

Any user can run fuser, but only the superuser can terminate another user's process.

Options The following options are supported:

`-c` Reports on files that are mount points for file systems, and any files within that mounted file system.

`-d` Report device usage information for all minor nodes bound to the same device node as the specified minor node. This option does not report file usage for files within a mounted file system.

`-f` Prints a report for the named file, not for files within a mounted file system.

`-k` Sends the `SIGKILL` signal to each process. Since this option spawns kills for each process, the kill messages may not show up immediately (see `kill(2)`). No signals will be sent to kernel file consumers.

`-n` Lists only processes with non-blocking mandatory locks on a file.

`-s` *sig* Sends a signal to each process. The *sig* option argument specifies one of the symbolic names defined in the `<signal.h>` header, or a decimal integer signal number. If *sig* is a symbolic name, it is recognized in a case-independent fashion, without the `SIG` prefix. The `-k` option is equivalent to `-s KILL` or `-s 9`. No signals will be sent to kernel file consumers.

`-u` Displays the user login name in parentheses following the process ID.

Examples

EXAMPLE 1 Reporting on the Mount Point and Files

The following example reports on the mount point and files within the mounted file system.

```
example% fuser -c /export/foo
```

EXAMPLE 2 Restricting Output when Reporting on the Mount Point and Files

The following example reports on the mount point and files within the mounted file system, but the output is restricted to processes that hold non-blocking mandatory locks.

```
example% fuser -cn /export/foo
```

EXAMPLE 3 Sending SIGTERM to Processes Holding a Non-blocking Mandatory Lock

The following command sends `SIGTERM` to any processes that hold a non-blocking mandatory lock on file `/export/foo/my_file`.

```
example% fuser -fn -s term /export/foo/my_file
```

Environment Variables

See `environ(5)` for descriptions of the following environment variables that affect the execution of `fuser`: `LANG`, `LC_ALL` `LC_CTYPE`, `LC_MESSAGES`, and `NLSPATH`.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os
Interface Stability	Committed

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Standard	See standards(5).

See Also ps(1), mount(1M), kill(2), mmap(2), signal(3C), attributes(5), environ(5), standards(5)

Notes Because fuser works with a snapshot of the system image, it may miss processes that begin using a file while fuser is running. Also, processes reported as using a file may have stopped using it while fuser was running. These factors should discourage the use of the -k option.

Name fwflash – firmware query and update utility

Synopsis

```
/usr/sbin/fwflash [-l [-c device_class | ALL ]]
     | [-v] | [-h]

fwflash [-f file1,file2,file3,... | -r file]
     [-y] [-d device_path]
```

Description The `fwflash` command writes a binary image file to supported flashable devices attached to a Solaris host. It also provides the ability to read firmware to a file if supported by the device. Because changing the firmware in a device can have significant impact on the stability of a system, only users with the privilege `All` are allowed to execute this command. Users authorized to run `fwflash` can be granted the "Firmware Flash Update" Rights Profile.

The first form of the command, above, provides information about devices. It lists all devices currently available on the system that are supported by `fwflash` for firmware upgrade. You can filter the list operation, to display only specified classes of devices. The second form of the command provides the operations to read or write the firmware images to specific devices.

Options The following options are supported:

`-c` *device_class*
: An optional parameter, valid only when used with the `-l` option. This option causes the command to list only devices of a specific class type. No other device classes are enumerated. Currently supported classes are `IB`, `enclosure`, `disk`, or `ALL`. If `-c` is not specified for the `-l` option, the class defaults to `ALL`.

  This option limits search to a specific class. Use `IB` for InfiniBand, `enclosure` for SCSI enclosures, and `disk` for SCSI/SATA/SAS/FC disks.

`-d` *dev_path*
: The *dev_path* is absolute path name of the device that the user wants to modify with the `-f`or `-r` operation. If the device cannot be found, the command fails. If the `-d` option is specified, then either `-f` or `-r` must also be specified.

`-f` *file1*,*file2*,*file3*,...
: Specify the path to one or more binary firmware image files you want to write to the device. `fwflash` will verify that each file is a valid firmware image for the specified device. If it is not, the command fails with an appropriate error message.

  If multiple firmware image files are specified, each image is verified and flashed to the device in the order given on the command line. If any of the specified files cannot be successfully flashed, then an appropriate message is displayed.

  After a new firmware image is flashed to a device, a reboot is required to correctly activate the new firmware.

`-h`
: Display the command line usage message for `fwflash`.

`-l`
List the devices on a system available for firmware upgrade and display information specific to each device or device class.

For InfiniBand (IB) devices, the list operation displays the `guids` (Globally Unique Identifier) currently set for the HCA, as well as the current firmware revision installed. There are four separate `guids` on the HCA; two of them can be set with the same value.

For SCSI Enclosure Services (`ses` or `sgen`) devices, an identifying target-port worldwide name is displayed, if available.

`-r` *file*
Specify the path to a file to create when reading the firmware from the device. The `-f` and `-r` options are mutually exclusive.

Not all flashable devices support reading firmware images back from the device. At present, only InfiniBand (IB) devices are supported for this operation. A message will be displayed if the selected device does not support this operation.

`-v`
Display `fwflash` version information and exit.

`-y`
Valid only when a flash read (`-r`) or write (`-f`) operation is specified. This option causes `fwflash` not to prompt for confirmation during operation and operate non-interactively. Note that there is no option that allows you to forcibly flash an incompatible firmware image onto a device.

Examples

EXAMPLE 1 Entering Command Without Arguments

The following command shows `fwflash` when the command is entered without arguments.

```
example# fwflash
Usage:
Usage:
      fwflash [-l [-c device_class | ALL]] | [-v] | [-h]
      fwflash [-f file1,file2,file3,... | -r file] [-y] -d device_path

      -l              list flashable devices in this system
      -c device_class limit search to a specific class
                      eg IB for InfiniBand, ses for SCSI Enclosures
      -v              print version number of fwflash utility
      -h              print this usage message

      -f file1,file2,file3,...
                      firmware image file list to flash
      -r file         file to dump device firmware to
      -y              answer Yes/Y/y to prompts
      -d device_path  pathname of device to be flashed
```

EXAMPLE 1 Entering Command Without Arguments *(Continued)*

```
      If -d device_path is specified, then one of -f <files>
      or -r <file> must also be specified

      If multiple firmware images are required to be flashed
      they must be listed together, separated by commas. The
      images will be flashed in the order specified.
```

EXAMPLE 2 Listing Devices Available to Flash

The following command lists the devices available to be flashed.

```
example# fwflash -l
List of available devices:
Device[0],  /devices/pci@0,0/pci8086,3595@2/pci8086,32a@0,2/\
            pci15b3,5a46@c/pci15b3,5a44@0:devctl
    Class [IB]
        GUID: System Image - 0002c901081e33b3
              Node         - 0000000000003446
              Port 1       - 0002c901081e33b1
              Port 2       - 0002c901081e33b2
        Firmware revision: 2.7.8100
               Product           : 375-3606-03
               PSID              : SUN0150000009
Device[1],  /devices/pci@0,0/pci8086,3597@4/pci15b3,6278@0:devctl
    Class [IB]
        GUID: System Image - 0002c9010a99e3b3
              Node         - 0002c9010a99e3b0
              Port 1       - 0002c9010a99e3b1
              Port 2       - 0002c9010a99e3b2
        Firmware revision: 2.7.8100
               Product           : 375-3606-03
               PSID              : SUN0150000009
```

Alternatively, for a SAS Expander presented as a SCSI Enclosure Services device, we might see output such as this:

```
example# fwflash -l
List of available devices:
Device[0] /devices/pci@0/pci@0/pci@2/scsi@0/ses@3,0:ses
  Class [sgen]
        Target port WWN  : 500605b00002453d
        Vendor           : SUN
        Product          : 16Disk Backplane
        Firmware revision: 5021
```

EXAMPLE 3 Flash Upgrading an IB HCA Device

The following command flash upgrades an IB HCA device.

```
example# fwflash -f ./version.3.2.0000 \
   -d /devices/pci@0,0/pci8086,3597@4/pci15b3,6278@0:devctl
About to update firmware on:
  /devices/pci@0,0/pci8086,3597@4/pci15b3,6278@0:devctl
Continue (Y/N): Y

Updating . . . . . .
Done.  New image will be active after the system is rebooted.
```

Note that you are prompted before the upgrading proceeds and that it is mandatory that you reboot your host to activate the new firmware image.

The following command adds the `-y` option to the command.

```
example# fwflash -y -f ./version.3.2.0000 \
   -d /devices/pci@0,0/pci8086,3597@4/pci15b3,6278@0:devctl
About to update firmware on:
  /devices/pci@0,0/pci8086,3597@4/pci15b3,6278@0:devctl

Updating . . . . . .
Done.  New image will be active after the system is rebooted.
```

EXAMPLE 4 Reading Device Firmware to File

The command shown below reads the device firmware to a file. The command uses the `-y` option so that read occurs without prompting.

```
example# fwflash -y -r /firmware.bin \
   -d /devices/pci@1d,700000/pci@1/pci15b3,5a44@0:devctl
About to read firmware on:
         /devices/pci@1d,700000/pci@1/pci15b3,5a44@0:devctl
to filename: /firmware.bin

Reading . . .
Done.
```

EXAMPLE 5 When No Flashable Devices Are Found

The command output shown below informs the user that there are no supported flashable devices found in the system:

```
example# fwflash -l
fwflash: No flashable devices attached with the ses driver in this system
fwflash: No flashable devices attached with the sgen driver in this system
fwflash: No flashable devices attached with the hermon driver in this system
fwflash: No flashable devices in this system
```

EXAMPLE 5 When No Flashable Devices Are Found *(Continued)*

Each plugin found in `/usr/lib/fwflash/identify` is loaded in turn, and walks the system device tree, determining whether any currently-attached devices can be flashed. For the list of device types and drivers that are currently supported, please see the `NOTES` section below.

Return Values

The `fwflash` command returns the following values:

`0`
Success

`1`
Failure

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/flash/fwflash
Interface Stability	Committed

See Also

`attributes(5)`, `hermon(7D)`, `ses(7D)`

The InfiniBand Trade Association website, `http://www.infinibandta.org`

The SCSI Storage Interfaces committee website, `http://www.t10.org`

SCSI Primary Commands-4, SPC4

SCSI Enclosure Services-2, SES2

Serial Attached SCSI-2, SAS2

Notes

The `fwflash` command supports:

- InfiniBand Host Channel Adapters (IB HCAs) containing either the AMD or the Intel parallel flash parts.
- SCSI Enclosure Services devices such as SAS Expanders, attached with `ses(7D)` drivers.

Name fwtmp, wtmpfix – manipulate connect accounting records

Synopsis

```
/usr/lib/acct/fwtmp [-ic]

/usr/lib/acct/wtmpfix [file]...
```

Description `fwtmp` reads from the standard input and writes to the standard output, converting binary records of the type found in `/var/adm/wtmpx` to formatted ASCII records. The ASCII version is useful when it is necessary to edit bad records.

`wtmpfix` examines the standard input or named files in `utmpx` format, corrects the time/date stamps to make the entries consistent, and writes to the standard output. A hyphen (–) can be used in place of *file* to indicate the standard input. If time/date corrections are not performed, `acctcon(1M)` will fault when it encounters certain date-change records.

Each time the date is set, a pair of date change records are written to `/var/adm/wtmpx`. The first record is the old date denoted by the string "`old time`" placed in the `line` field and the flag `OLD_TIME` placed in the `type` field of the `utmpx` structure. The second record specifies the new date and is denoted by the string `new time` placed in the `line` field and the flag `NEW_TIME` placed in the `type` field. `wtmpfix` uses these records to synchronize all time stamps in the file.

In addition to correcting time/date stamps, `wtmpfix` will check the validity of the `name` field to ensure that it consists solely of alphanumeric characters or spaces. If it encounters a name that is considered invalid, it will change the login name to `INVALID` and write a diagnostic to the standard error. In this way, `wtmpfix` reduces the chance that `acctcon` will fail when processing connect accounting records.

Options `-ic` Denotes that input is in ASCII form, and output is to be written in binary form.

Files `/var/adm/wtmpx` history of user access and administration information

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/accounting/legacy-accounting

See Also `acctcom(1)`, `ed(1)`, `acct(1M)`, `acctcms(1M)`, `acctcon(1M)`, `acctmerg(1M)`, `acctprc(1M)`, `acctsh(1M)`, `runacct(1M)`, `acct(2)`, `acct.h(3HEAD)`, `utmpx(4)`, `attributes(5)`

Oracle Solaris Administration: Common Tasks

Name getdevpolicy – inspect the system's device policy

Synopsis

```
/usr/sbin/getdevpolicy [device...]
```

Description Without arguments, `getdevpolicy` outputs the device policy in effect to standard output.

With arguments, each argument is treated as a pathname to a device and the device policy in effect for that specific device is printed preceded by the supplied pathname.

Usage The device policy adds access restrictions over and above the file permissions.

Exit Status The following exit values are returned:

`0` Successful completion.

`non-zero` An error occurred.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os
Interface Stability	See below

The invocation is evolving. The output is unstable.

See Also `add_drv(1M)`, `rem_drv(1M)`, `update_drv(1M)`, `attributes(5)`, `privileges(5)`, `devfs(7FS)`

Name getent – get entries from administrative database

Synopsis `getent` *database* [*key*]...

Description `getent` gets a list of entries from the administrative database specified by *database*. The information generally comes from one or more of the sources that are specified for the *database* in `/etc/nsswitch.conf`.

database is the name of the database to be examined. This can be `passwd`, `group`, `hosts`, `ipnodes`, `services`, `protocols`, `auth_attr` or `exec_attr`. For each of these databases, `getent` uses the appropriate library routines described in `getpwnam`(3C), `getgrnam`(3C), `gethostbyaddr`(3NSL), `gethostbyname`(3NSL), `getipnodebyaddr`(3SOCKET), `getipnodebyname`(3SOCKET), `getservbyname`(3SOCKET), `getprotobyname`(3SOCKET), `getprofattr`(3C), `getauthattr`(3C), and `getexecattr`(3C), respectively.

Each *key* must be in a format appropriate for searching on the respective database. For example, it can be a *username* or *numeric-uid* for `passwd`; *hostname* or *IP address* for `hosts`; or *service*, *service/protocol*, *port*, or *port/proto* for `services`.

`getent` prints out the database entries that match each of the supplied keys, one per line, in the format of the matching administrative file: `passwd`(4), `group`(4), `project`(4), `networks`(4), `netmasks`(4), `user_attr`(4), `prof_attr`(4), `auth_attr`(4), or `exec_attr`(4). The key for `exec_attr`(4) is a profile name. If no key is given, all entries returned by the corresponding enumeration library routine, for example, `getpwent()` or `gethostent()`, are printed. Enumeration is not supported on `ipnodes`, `ethers`, and `netmasks`.

Key Interpretation for passwd and group Databases

When `getent` is invoked with database set to `passwd`, each key value is processed as follows:

- If the key value consists only of numeric characters, `getent` assumes that the key value is a numeric user ID and searches the user database for a matching user ID.
- If the user ID is not found in the user database or if the key value contains any non-numeric characters, `getent` assumes the key value is a user name and searches the user database for a matching user name.

Similarly, when `getent` is invoked with database set to `group`, each key value is processed as follows:

- If the key value consists only of numeric characters, `getent` assumes that the key value is a numeric group ID and searches the group database for a matching group ID.
- If the group ID is not found in the `group` database or if the key value contains any non-numeric characters, `getent` assumes the key value is a group name and searches the `group` database for a matching group name.

When `getent` is invoked with database set to `user_attr`(4), each key value is processed as follows:

- If the key value consists only of numeric characters, `getent` assumes that the key value is a numeric user ID and searches the user database for a matching user ID.

- If the key value contains any non-numeric characters, `getent` assumes the key value is a user name and searches the user database for a matching user name.

Exit Status The following exit values are returned:

`0` Successful completion.

`1` Command syntax was incorrect, an invalid option was used, or an internal error occurred.

`2` At least one of the specified entry names was not found in the database.

`3` There is no support for enumeration on this database.

Files

`/etc/nsswitch.conf`	name service switch configuration file
`/etc/passwd`	password file
`/etc/group`	group file
`/etc/inet/hosts`	IPv4 and IPv6 host name database
`/etc/services`	Internet services and aliases
`/etc/project`	project file
`/etc/protocols`	protocol name database
`/etc/ethers`	Ethernet address to hostname database or domain
`/etc/networks`	network name database
`/etc/netmasks`	network mask database
`/etc/user_attr`	Extended user attributes
`/etc/security/prof_attr`	Profile description database
`/etc/security/auth_attr`	Authorization description
`/etc/security/exec_attr`	Execution profiles database

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os
Interface Stability	Committed

See Also `ethers(3SOCKET)`, `getgrnam(3C)`, `gethostbyaddr(3NSL)`, `gethostbyname(3NSL)`, `gethostent(3NSL)`, `getipnodebyaddr(3SOCKET)`, `getipnodebyname(3SOCKET)`, `getnetbyname(3SOCKET)`, `getprojbyname(3PROJECT)`, `getprotobyname(3SOCKET)`, `getpwnam(3C)`, `getservbyname(3SOCKET)`, `getauthattr(3C)`, `getexecattr(3C)`,

getprofattr(3C), getuserattr(3C), ethers(4), group(4), hosts(4), netmasks(4), networks(4), nsswitch.conf(4), auth_attr(4), exec_attr(4), passwd(4), prof_attr(4), project(4), protocols(4), services(4), user_attr(4), attributes(5)

Name gettable – get DoD Internet format host table from a host

Synopsis `/usr/sbin/gettable` *host*

Description `gettable` is a simple program used to obtain the DoD Internet host table from a "hostname" server. The specified *host* is queried for the table. The table is placed in the file `hosts.txt`.

`gettable` operates by opening a TCP connection to the port indicated in the service specification for "hostname". A request is then made for all names and the resultant information is placed in the output file.

`gettable` is best used in conjunction with the `htable(1M)` program which converts the DoD Internet host table format to that used by the network library lookup routines.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/network/nis

See Also `htable(1M)`, `attributes(5)` Harrenstien, Ken, Mary Stahl, and Elizabeth Feinler, HOSTNAME *Server*, RFC 953, Network Information Center, SRI International, Menlo Park, California, October 1985.

Notes Should allow requests for only part of the database.

Name getty – set terminal type, modes, speed, and line discipline

Synopsis

```
/usr/lib/saf/ttymon [-h] [-t timeout] line
     [speed [type [linedisc]]]

/usr/lib/saf/ttymon -c file
```

Description `getty` sets terminal type, modes, speed, and line discipline. `getty` is a symbolic link to `/usr/lib/saf/ttymon`. It is included for compatibility with previous releases for the few applications that still call `getty` directly.

`getty` can only be executed by the super-user, (a process with the user ID `root`). Initially `getty` prints the login prompt, waits for the user's login name, and then invokes the `login` command. `getty` attempts to adapt the system to the terminal speed by using the options and arguments specified on the command line.

Without optional arguments, `getty` specifies the following: The *speed* of the interface is set to 300 baud, either parity is allowed, NEWLINE characters are converted to carriage return-line feed, and tab expansion is performed on the standard output. `getty` types the login prompt before reading the user's name a character at a time. If a null character (or framing error) is received, it is assumed to be the result of the user pressing the BREAK key. This will cause `getty` to attempt the next *speed* in the series. The series that `getty` tries is determined by what it finds in `/etc/ttydefs` .

Options The following options are supported:

`-h` If the `-h` flag is not set, a hangup will be forced by setting the speed to zero before setting the speed to the default or a specified speed.

`-t` *timeout* Specifies that `getty` should exit if the open on the line succeeds and no one types anything in *timeout* seconds.

`-c` *file* The `-c` option is no longer supported. Instead use `/usr/sbin/sttydefs -l` to list the contents of the `/etc/ttydefs` file and perform a validity check on the file.

Operands The following operands are supported:

line The name of a TTY line in `/dev` to which `getty` is to attach itself. `getty` uses this string as the name of a file in the `/dev` directory to open for reading and writing.

speed The *speed* argument is a label to a speed and TTY definition in the file `/etc/ttydefs`. This definition tells `getty` at what speed to run initially, what the initial TTY settings are, and what speed to try next, (should the user press the BREAK key to indicate that the speed is inappropriate). The default *speed* is 300 baud.

type and *linedisc* These options are obsolete and will be ignored.

Files `/etc/ttydefs`

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also `ct(1C)`, `login(1)`, `sttydefs(1M)`, `ttymon(1M)`, `ioctl(2)`, `attributes(5)`, `tty(7D)`

Name gkadmin – Kerberos principals and policies administration GUI

Synopsis `/usr/sbin/gkadmin`

Description `gkadmin` is an interactive graphical user interface (GUI) that enables you to maintain Kerberos principals and policies. `gkadmin` provides much the same functionality as the `kadmin(1M)` command.

`gkadmin` does not support the management of keytabs. You must use `kadmin` for keytabs management. `gkadmin` uses Kerberos authentication and an encrypted RPC to operate securely from anywhere on the network.

When `gkadmin` is invoked, the login window is populated with default values. For the principal name, `gkadmin` determines your user name from the `USER` environment variable. It appends `/admin` to the name (*username*`/admin`) to create a default user instance in the same manner as `kadmin`. It also selects appropriate defaults for realm and master KDC (*admin_server*) from the `/etc/krb5/krb5.conf` file.

You can change these defaults on the login window. When you enter your password, a session is started with `kadmind`. Operations performed are subject to permissions that are granted or denied to the chosen user instance by the Kerberos ACL file. See `kadm5.acl(4)`.

After the session is started, a tabbed folder is displayed that contains a principal list and a policy list. The functionality is mainly the same as `kadmin`, with addition, deletion, and modification of principal and policy data available.

`gkadmin` also includes an interface to specify principal key encryption types when modifying or creating principal records. The default set of encryption types is used if they are not selected through this interface. The default set of encryption types can be found in `krb5.conf(4)` under the `default_tkt_enctypes` section.

In addition, `gkadmin` provides the following features:

- New principal or policy records can be added either from default values or from the settings of an existing principal.
- A comment field is available for principals.
- Default values are saved in `$HOME/.gkadmin`.
- A logout option permits you to log back in as another user instance without exiting the tool.
- Principal and policy lists and attributes can be printed or saved to a file.
- Online context-sensitive help and general help is available in the `Help` menu.

Files

`/etc/krb5/krb5.conf` Kerberos configuration information on a Kerberos client. Used to search for default realm and master KDC (`admin_server`), including a port number for the master KDC.

`$HOME/.gkadmin`	Default parameters used to initialize new principals created during the session.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/security/kerberos-5
Interface Stability	Committed

See Also `kpasswd(1)`, `kadmin(1M)`, `kadmind(1M)`, `kadmin.local(1M)`, `kdb5_util(1M)`, `kadm5.acl(4)`, `kdc.conf(4)`, `krb5.conf(4)`, `attributes(5)`, `kerberos(5)`

Diagnostics The `gkadmin` interface is currently incompatible with the MIT `kadmind` daemon interface, so you cannot use this interface to administer an MIT-based Kerberos database. However, clients running the Solaris implementation of Kerberos can still use an MIT-based KDC.

Name groupadd – add (create) a new group definition on the system

Synopsis

```
/usr/sbin/groupadd [-g gid [-o]] [-S repository]
     [-U user1[,user2..] ] group
```

Description The `groupadd` command creates a new group definition on the system by adding the appropriate entry to the group database in the `files` and `ldap` repositories.

An administrator must be granted the User Management rights profile or have `solaris.group.manage` authorization to be able to add a group. Once the group is successfully added, the administrator is granted the authorization to modify and delete the group. See `groupmod(1M)` and `groupdel(1M)`. An administrator who is assigned the solaris.group.assign authorization, typically the root account, can modify the authorization assignment with `usermod(1M)`.

Options The following options are supported:

`-g` *gid*
Assigns the group id *gid* for the new group. This group id must be a non-negative decimal integer below `MAXUID` as defined in `/usr/include/sys/param.h`. The group ID defaults to the next available (unique) number above the highest number currently assigned. For example, if groups `100`, `105`, and `200` are assigned as groups, the next default group number is `201`. (Group IDs from `0`–`99` are reserved by SunOS for future applications.)

`-o`
Allows the *gid* to be duplicated (non-unique). An administrator must have `solaris.group.assign` authorization to use this option.

`-S` *repository*
The valid repositories are `files` and `ldap`. The repository specifies which name service will be updated. When *repository* is not specified, the `files` repository is used. When the repository is `files`, the user name and other items can be present in other name service repositories and can be assigned to a group in the `files` repository. When the repository is `ldap`, all the assignable attributes must be present in the `ldap` repository.

`-U` *user1*[,*user2*]
Adds a list of users *user1*, *user2* to the group.

Operands The following operands are supported:

group A string consisting of characters from the set of lower case alphabetic characters and numeric characters. A warning message is written if the string exceeds `MAXGLEN-1`, which is usually eight characters. The *group* field must contain at least one character; it accepts lower case or numeric characters or a combination of both, and must not contain a colon (`:`) or NEWLINE.

Exit Status The following exit values are returned:

`0` Successful completion.

`2` Invalid command syntax. A usage message for the `groupadd` command is displayed.

`3` An invalid argument was provided to an option.

`4` The *gid* is not unique (when `-o` option is not used).

`9` The *group* is not unique.

`10` The group database cannot be updated.

Files

- `/etc/group`
- `/usr/include/userdefs.h`

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also `users(1B)`, `groupdel(1M)`, `groupmod(1M)`, `grpck(1M)`, `logins(1M)`, `pwck(1M)`, `useradd(1M)`, `userdel(1M)`, `usermod(1M)`, `group(4)`, `attributes(5)`

Notes `groupadd` adds a group definition to the system. If a network name service is being used to supplement the local `/etc/group` file with additional entries, `groupadd` verifies the uniqueness of a specified group name and group ID against the external name service and uses the entries in the `files` repository.

If the number of characters in a `group` entry exceeds 2047, group maintenance commands, such as `groupdel(1M)` and `groupmod(1M)`, fail.

Name groupdel – delete a group definition from the system

Synopsis `/usr/sbin/groupdel [-S` *repository*`] group`

Description The `groupdel` utility deletes a group definition from the system. It deletes the appropriate entry from the `/etc/group` file.

In addition to `solaris.group.manage` authorization, an administrator must have either `solaris.group.assign` or a matching authorization of the form `solaris.group.assign/`*groupname* to delete a group. The authorization `solaris.group.assign/`*groupname* is automatically assigned to the administrator who created the group.

Options `groupdel` supports the following option.

`-S` *repository*
The valid repositories are `files` and `ldap`. The repository specifies which name service will be updated. When *repository* is not specified, `groupdel` consults `nsswitch.conf(4)`.

Operands *group* An existing group name to be deleted.

Exit Status The following exit values are returned:

`0` Success.

`2` Invalid command syntax. A usage message for the `groupdel` command is displayed.

`6` *group* does not exist.

`10` Cannot update the `/etc/group` file.

Files `/etc/group` system file containing group definitions

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also `users(1B)`, `groupadd(1M)`, `groupmod(1M)`, `logins(1M)`, `useradd(1M)`, `userdel(1M)`, `usermod(1M)`, `nsswitch.conf(4)`, `attributes(5)`

Notes The `groupdel` utility deletes a group definition that is in the group database in the repository.

`groupdel` fails if a group entry (a single line in `/etc/group`) exceeds 2047 characters.

Name groupmod – modify a group definition on the system

Synopsis

```
/usr/sbin/groupmod [-S repository] [-g gid [-o]] [-n name]
     [-U [+|-]user1[,user2]...] group
```

Description The `groupmod` command modifies the definition of the specified group by modifying the appropriate entry in the group database in the repository.

An administrator can modify any group for which it has a matching authorization of the form `solaris.group.assign`/*groupname*. This authorization is automatically assigned to the administrator who created the group. An administrator must have `solaris.group.assign` authorization to modify all other groups.

Options The following options are supported:

`-g` *gid*
Specify the new group ID for the group. This group ID must be a non-negative decimal integer less than `MAXUID`, as defined in <param.h>. The group ID defaults to the next available (unique) number above 99. (Group IDs from 0-99 are reserved by SunOS for future applications.)

`-n` *name*
Specify the new name for the group. The *name* argument is a string of no more than eight bytes consisting of characters from the set of lower case alphabetic characters and numeric characters. A warning message will be written if these restrictions are not met. A future Solaris release may refuse to accept group fields that do not meet these requirements. The *name* argument must contain at least one character and must not include a colon (`:`) or NEWLINE (`\n`).

`-o`
Allow the *gid* to be duplicated (non-unique). An administrator must have solaris.group.assign authorization to use this option.

`-S` *repository*
The valid repositories are `files` and `ldap`. The repository specifies which name service will be updated. When *repository* is not specified, `groupmod` consults `nsswitch.conf(4)`. When the repository is `files`, the user name and other items can be present in other name service repositories and can be assigned to a group in the `files` repository. When the repository is `ldap`, all the assignable attributes must be present in the `ldap` repository.

`-U` [+|-]*user1*[,*user2*]
Updates the list of users for the group as follows:

- A prefix + before the list adds that list to existing users list.
- A prefix - before the list removes each user in the list from the existing users list.
- With no prefix before the list, replaces the existing users list with the new list of users specified.

Operands The following operands are supported:

group An existing group name to be modified.

Exit Status The `groupmod` utility exits with one of the following values:

`0` Success.

`2` Invalid command syntax. A usage message for the `groupmod` command is displayed.

`3` An invalid argument was provided to an option.

`4` *gid* is not unique (when the `-o` option is not used).

`6` *group* does not exist.

`9` *name* already exists as a group name.

`10` Cannot update the `/etc/group` file.

Files `/etc/group` group file

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also `users(1B)`, `groupadd(1M)`, `groupdel(1M)`, `logins(1M)`, `useradd(1M)`, `userdel(1M)`, `usermod(1M)`, `group(4)`, `nsswitch.conf(4)`, `attributes(5)`

Notes The `groupmod` utility only modifies group definitions in the group database in the repository. If a network name service such as NIS is being used to supplement the local `/etc/group` file with additional entries, `groupmod` cannot change information supplied by the network name service. `groupmod` verifies the uniqueness of group name and group ID against the external name service and uses the entries in the `files` repository.

`groupmod` fails if a group entry (a single line in `/etc/group`) exceeds 2047 characters.

Name growfs – non-destructively expand a UFS file system

Synopsis
```
/usr/sbin/growfs [-M mount-point] [newfs-options]
      [raw-device]
```

Description `growfs` non-destructively expands a mounted or unmounted UNIX file system (UFS) to the size of the file system's slice(s).

Typically, disk space is expanded by first adding a slice to a metadevice, then running the `growfs` command. When adding space to a mirror, you expand each submirror before expanding the file system.

`growfs` will "write-lock" (see `lockfs(1M)`) a mounted file system when expanding. The length of time the file system is write-locked can be shortened by expanding the file system in stages. For instance, to expand a 1 Gbyte file system to 2 Gbytes, the file system can be grown in 16 Mbyte stages using the `-s` option to specify the total size of the new file system at each stage. The argument for `-s` is the number of sectors, and must be a multiple of the cylinder size. Note: The file system cannot be grown if a cylinder size of less than 2 is specified. Refer to the `newfs(1M)` man page for information on the options available when growing a file system.

`growfs` displays the same information as `mkfs` during the expansion of the file system.

If `growfs` is aborted, recover any lost free space by unmounting the file system and running the `fsck` command, or run the `growfs` command again.

Note: If `growfs` is aborted and the file system is used before `fsck` is run on it, UFS metadata might be left in an incomplete state, with the result that the file system would be corrupted. In such a circumstance, you would have to restore the file system from backups.

Options Root privileges are required for all of the following options.

`-M` *mount-point*	The file system to be expanded is mounted on *mount-point*. File system locking (`lockfs`) will be used.
newfs-options	The options are documented in the `newfs` man page.
raw-device	Specifies the name of a raw metadevice or raw special device, residing in `/dev/md/rdsk`, or `/dev/rdsk`, respectively, including the disk slice, where you want the file system to be grown.

Examples **EXAMPLE 1** Expanding nonmetadevice slice for `/export` file system

The following example expands a nonmetadevice slice for the `/export` file system. In this example, the existing slice, `/dev/dsk/c1t0d0s3`, is converted to a metadevice so additional slices can be concatenated.

```
# metainit -f d8 2 1 c1t0d0s3 1 c2t0d0s3
# umount /export
```

EXAMPLE 2 Associate /export with new metadevice

Edit the /etc/vfstab file to change the entry for /export to the newly defined metadevice, d8.

```
# mount /export
# growfs -M /export /dev/md/rdsk/d8
```

The first example starts by running the metainit command with the -f option to force the creation of a new concatenated metadevice d8, which consists of the existing slice /dev/dsk/c1t0d0s3 and a new slice /dev/dsk/c2t0d0s3. Next, the file system on /export must be unmounted. The /etc/vfstab file is edited to change the entry for /export to the newly defined metadevice name, rather than the slice name. After the file system is remounted, the growfs command is run to expand the file system. The file system will span the entire metadevice when growfs completes. The -M option enables the growfs command to expand a mounted file system. During the expansion, write access for /export is suspended until growfs unlocks the file system. Read access is not affected, though access times are not kept when the lock is in effect.

EXAMPLE 3 Dynamic Expansion of /export file system

The following example picks up from the previous one. Here, the /export file system mounted on metadevice d8 is dynamically expanded.

```
# metattach d8 c0t1d0s2
# growfs -M /export /dev/md/rdsk/d8
```

This example begins by using the metattach command to dynamically concatenate a new slice, /dev/dsk/c0t1d0s2, to the end of an existing metadevice, d8. Next, the growfs command specifies that the mount-point is /export and that it is to be expanded onto the raw metadevice /dev/md/rdsk/d8. The file system will span the entire metadevice when growfs completes. During the expansion, write access for /export is suspended until growfs unlocks the file system. Read access is not affected, though access times are not kept when the lock is in effect.

EXAMPLE 4 Expanding mounted file system to existing mirror

The following example expands a mounted file system /files, to an existing mirror, d80, which contains two submirrors, d9 and d10.

```
# metattach d9 c0t2d0s5
# metattach d10 c0t3d0s5
# growfs -M /files /dev/md/rdsk/d80
```

In this example, the metattach command dynamically concatenates the new slices to each submirror. The metattach command must be run for each submirror. The mirror will automatically grow when the last submirror is dynamically concatenated. The mirror will grow to the size of the smallest submirror. The growfs command then expands the file system. The growfs command specifies that the mount-point is /files and that it is to be expanded

EXAMPLE 4 Expanding mounted file system to existing mirror *(Continued)*

onto the raw metadevice `/dev/md/rdsk/d80`. The file system will span the entire mirror when the `growfs` command completes. During the expansion, write access for the file system is suspended until `growfs` unlocks the file system. Read access is not affected, though access times are not kept when the lock is in effect.

Exit Status The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	storage/svm

See Also `fsck(1M)`, `lockfs(1M)`, `mkfs(1M)`, `metattach(1M)`, `newfs(1M)`, `attributes(5)`

Solaris Volume Manager Administration Guide

Limitations Only UFS file systems (either mounted or unmounted) can be expanded using the `growfs` command. Once a file system is expanded, it cannot be decreased in size. The following conditions prevent you from expanding file systems: When `acct` is activated and the accounting file is on the target device. When C2 security is activated and the logging file is on the target file system. When there is a local `swap` file in the target file system. When the file system is root (`/`), `/usr`, or `swap`.

Name gsscred – add, remove, and list gsscred table entries

Synopsis

```
gsscred [-n user [-o oid] [-u uid]] [-c comment] -m mech -a

gsscred [-n user [-o oid]] [-u uid] [-m mech] -r

gsscred [-n user [-o oid]] [-u uid] [-m mech] -l
```

Description The `gsscred` utility is used to create and maintain a mapping between a security principal name and a local UNIX *uid*. The format of the user name is assumed to be `GSS_C_NT_USER_NAME`. You can use the `-o` option to specify the object identifier of the *name* type. The `OID` must be specified in dot-separated notation, for example: `1.2.3.45464.3.1`

The `gsscred` table is used on server machines to lookup the *uid* of incoming clients connected using `RPCSEC_GSS`.

When adding users, if no *user* name is specified, an entry is created in the table for each user from the `passwd` table. If no *comment* is specified, the `gsscred` utility inserts a comment that specifies the user name as an ASCII string and the GSS-APIsecurity mechanism that applies to it. The security mechanism will be in string representation as defined in the `/etc/gss/mech` file.

The parameters are interpreted the same way by the `gsscred` utility to delete users as they are to create users. At least one of the following options must be specified: `-n`, `-u`, or `-m`. If no security mechanism is specified, then all entries will be deleted for the user identified by either the *uid* or *user* name. If only the security mechanism is specified, then all *user* entries for that security mechanism will be deleted.

Again, the parameters are interpreted the same way by the `gsscred` utility to search for users as they are to create users. If no options are specified, then the entire table is returned. If the *user* name or *uid* is specified, then all entries for that *user* are returned. If a security mechanism is specified, then all *user* entries for that security mechanism are returned.

Options

`-a`	Add a table entry.
`-c` *comment*	Insert comment about this table entry.
`-l`	Search table for entry.
`-m` *mech*	Specify the mechanism for which this name is to be translated.
`-n` *user*	Specify the optional principal name.
`-o` *oid*	Specify the OID indicating the name type of the user.
`-r`	Remove the entry from the table.
`-u` *uid*	Specify the *uid* for the *user* if the *user* is not local.

Examples

EXAMPLE 1 Creating a `gsscred` Table for the Kerberos v5 Security Mechanism

The following shows how to create a `gsscred` table for the kerberos v5 security mechanism. `gsscred` obtains *user* names and *uid*'s from the `passwd` table to populate the table.

```
example% gsscred -m kerberos_v5 -a
```

EXAMPLE 2 Adding an Entry for `root/host1` for the Kerberos v5 Security Mechanism

The following shows how to add an entry for `root/host1` with a specified *uid* of `0` for the kerberos v5 security mechanism.

```
example% gsscred -m kerberos_v5 -n root/host1 -u 0 -a
```

EXAMPLE 3 Listing All User Mappings for the Kerberos v5 Security Mechanism

The following lists all user mappings for the kerberos v5 security mechanism.

```
example% gsscred -m kerberos_v5 -l
```

EXAMPLE 4 Listing All Mappings for All Security Mechanism for a Specified User

The following lists all mappings for all security mechanisms for the user `bsimpson`.

```
example% gsscred -n bsimpson -l
```

Exit Status

The following exit values are returned:

`0` Successful completion.

>`0` An error occurred.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/library/security/gss
Interface Stability	Committed

See Also

`gssd(1M)`, `gsscred.conf(4)`, `attributes(5)`

Notes

Some GSS mechanisms, such as `kerberos_v5`, provide their own authenticated-name-to-local-name (uid) mapping and thus do not usually have to be mapped using `gsscred`. See `gsscred.conf(4)` for more information.

Name gssd – generates and validates GSS-API tokens for kernel RPC

Synopsis

```
/usr/lib/gss/gssd
```

Description `gssd` is the user mode daemon that operates between the kernel rpc and the Generic Security Service Application Program Interface (GSS-API) to generate and validate GSS-API security tokens. In addition, `gssd` maps the GSS-API principal names to the local user and group ids. By default, all groups that the requested user belongs to will be included in the grouplist credential. `gssd` is invoked by the Internet daemon `inetd(1M)` the first time that the kernel RPC requests GSS-API services.

Exit Status The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/kernel/security/gss
Interface Stability	Committed

See Also `kill(1)`, `pkill(1)`, `svcs(1)`, `inetadm(1M)`, `inetd(1M)`, `gsscred(1M)`, `svcadm(1M)`, `gsscred.conf(4)`, `resolv.conf(4)`, `attributes(5)`, `smf(5)`

RFC 2078

Notes The following signal has the specified effect when sent to the server process using the `kill(1)` command:

`SIGHUP` `gssd` rereads the `gsscred.conf(4)` options.

When one of the mechanisms being used is Kerberos, then the `gssd` process must be restarted after adding or changing the `resolv.conf(4)` file.

The `gssd` service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/network/rpc/gss:default
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. Responsibility for initiating and restarting this service is delegated to `inetd(1M)`. Use `inetadm(1M)` to make configuration changes and to view configuration information for this service. The service's status can be queried using the `svcs(1)` command.

Name hald – daemon that supports hardware abstraction layer

Synopsis

```
/usr/lib/hal/hald [--daemon={yes | no}]] [--help] [--use-syslog]
 [--verbose={yes | no}] [--version]
```

Description The `hald` daemon supports the recognition of hardware changes for devices that conform to the Hardware Abstraction Layer (HAL) specification.

The enabling and disabling of `hald` can be performed through the service management facility (SMF) (see `smf(5)`). `hald` is managed using the fault management resource identifier (FMRI) `svc:/system/hal`.

Options The following options are supported:

`--daemon=yes|no` Run as a daemon.

`--help` Display usage information and exit.

`--use-syslog` Display debug messages to `syslog` instead of `stderr`. Use this option to record debug messages if HAL runs as daemon.

`--verbose=yes|no` Display debug information.

`--version` Display version information and exit.

Files `/usr/lib/hal` HAL-related files

`/etc/hal/fdi` Device information files

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/hal
Interface Stability	Volatile

See Also `svccfg(1M)`, `attributes(5)`, `hal(5)`

Name hal-device – manage HAL devices

Synopsis

```
hal-device  [-h] [--a udi |  --r udi]
```

Description The Hardware Abstraction Layer (HAL) provides a view of the various hardware attached to a system. The `hal-device` command enables you to manage devices that conform to the HAL standard. Specifically, `hal-device` lets you add or remove a device to or from the HAL global device list. Device properties are read from stdin in `lshal(1M)` syntax.

Options The following options are supported:

`-a`, `--add` *udi* Add device specified by Universal Device Identifier *udi* to HAL's global device list.

`-h` Display usage information.

`-r`, `--remove` *udi* Remove device specified by Universal Device Identifier *udi* from HAL's global device list.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/hal
Interface Stability	Volatile

See Also `hald(1M)`, `attributes(5)`, `hal(5)`

Name hal-fdi-validate – validate HAL device information files

Synopsis

```
hal-fdi-validate  [-f dtd] file [file]...
```

Description The `hal-fdi-validate` command validates one or more device information files. See `fdi(4)`. The standard DTD file will be used unless the `-f` option is used to specify a different file.

Options The following option is supported:

`-f` *dtd* Specify path to a DTD file.

Operands The `hal-fdi-validate` command accepts the following operand(s):

file [*file*...] One or more DTD files to be validated.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/hal
Interface Stability	Volatile

See Also `hald(1M)`, `fdi(4)`, `attributes(5)`, `hal(5)`

Name hal-find, hal-find-by-capability, hal-find-by-property – search HAL global device list

Synopsis

```
hal-find-by-capability --capability capability [--help]
 [--verbose] [--version]

hal-find-by-property  --key key --string value [--help]
 [--verbose] [--version]
```

Description The `hal-find` commands, `hal-find-by-capability` and `hal-find-by-property`, search the Hardware Abstraction Layer (HAL) device list by specified criteria and displays results on the standard output. `hal-find-by-capability` searches by capability, such as `volume` or `block`. `hal-find-by-property` searches by property, such as `block.is_volume` or `volume.disc.has_audio`.

Options The following options are supported:

`--capability` *capability*	HAL device capability to search for.
`--help`	Display list of options.
`--key` *key*	The *key* to the property that is the basis of the search.
`--string` *value*	The string *value* associated with the property that is the basis of the search.
`--verbose`	Verbose mode.
`--version`	Display version and exit.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/hal
Interface Stability	Volatile

See Also `hald(1M)`, `attributes(5)`, `hal(5)`

Name hal-get-property, hal-set-property – get and set HAL device properties

Synopsis

```
hal-get-property --udi udi --key key [--help] [--verbose]
 [--version]

hal-set-property  --udi udi --key key {--int value |  --uint64 value
 |  --string value |  --bool value |  --strlist-pre value
 |  --strlist-post value |  --strlist-rem value |  --double value
 |  --remove value} [--help] [--version]
```

Description The Hardware Abstraction Layer (HAL) provides a view of the various hardware attached to a system. This view is updated dynamically as hardware configuration changes by means of hotplug or other mechanisms. HAL represents a piece of hardware as a device object. A device object is identified by a unique identifer and carries a set of key/value pairs, referred to as device properties. Some properties are derived from the actual hardware, some are merged from device information files (`.fdi` files), and some are related to the actual device configuration.

The `hal-get-property` and `hal-set-property` commands allow you to get and set properties of hardware that conforms to HAL specifications.

Options The following options are supported:

`--udi` *udi*
Unique device ID.

`--key` *key*
Key of the property to set.

`--int`
Set value to an integer. Accepts decimal or hexadecimal value prefixed with `0x` or `x`.

`--uint64`
Set value to an integer. Accepts decimal or hexadecimal value prefixed with `0x` or `x`.

`--string` *value*
Set value to a string.

`--double` *value*
Set value to a floating point number.

`--boolean` *value*
Set value to a boolean, that is, true or false

`--strlist-pre` *value*
Prepend a string to a list.

`--strlist-post` *value*
Append a string to a list.

`--strlist-rem` *value*
Remove a string from a list.

`--remove` *value*
Indicates that the property should be removed.

`--version`
Display version and exit.

`--help`
Display list of options and exit

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/hal
Interface Stability	Volatile

See Also `hald(1M)`, `attributes(5)`, `hal(5)`

Name halt, poweroff – stop the processor

Synopsis

```
/usr/sbin/halt [-dlnqy]

/usr/sbin/poweroff [-dlnqy]
```

Description The `halt` and `poweroff` utilities write any pending information to the disks and then stop the processor. The `poweroff` utility has the machine remove power, if possible.

The `halt` and `poweroff` utilities normally log the system shutdown to the system log daemon, `syslogd(1M)`, and place a shutdown record in the login accounting file `/var/adm/wtmpx`. These actions are inhibited if the `-n` or `-q` options are present.

Options The following options are supported:

`-d` Force a system crash dump before rebooting. See `dumpadm(1M)` for information on configuring system crash dumps.

`-l` Suppress sending a message to the system log daemon, `syslogd(1M)`, about who executed `halt`.

`-n` Prevent the `sync(1M)` before stopping.

`-q` Quick halt. No graceful shutdown is attempted.

Files `/var/adm/wtmpx` History of user access and administration information.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also `dumpadm(1M)`, `init(1M)`, `reboot(1M)`, `shutdown(1M)`, `sync(1M)`, `syslogd(1M)`, `inittab(4)`, `attributes(5)`, `smf(5)`

Notes The `halt` and `poweroff` utilities do not cleanly shutdown `smf(5)` services, execute the scripts in `/etc/rc`*num*`.d`, or execute shutdown actions in `inittab(4)`. To ensure a complete shutdown of system services, use `shutdown(1M)` or `init(1M)` to reboot a Solaris system.

Name hextoalabel – convert an internal text label to its human readable equivalent

Synopsis

```
/usr/sbin/hextoalabel [internal-text-sensitivity-label]
/usr/sbin/hextoalabel -c [internal-text-clearance]
```

Description `hextoalabel` converts an internal text label into its human readable equivalent and writes the result to the standard output file. This internal form is often hexadecimal. If no option is supplied, the label is assumed to be a sensitivity label.

If no internal text label is specified, the label is read from the standard input file. The expected use of this command is emergency repair of labels that are stored in internal databases.

Options

`-c` Identifies the internal text label as a clearance.

Exit Status The following exit values are returned:

`0` On success.

`1` On failure, and writes diagnostics to the standard error file.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/trusted
Interface Stability	See below.

The command output is Committed for systems with the same `label_encodings` file. The command invocation is Committed for systems that implement the DIA MAC policy.

Files

`/etc/security/tsol/label_encodings`
The label encodings file contains the classification names, words, constraints, and values for the defined labels of this system.

See Also `atohexlabel(1M)`, `label_to_str(3TSOL)`, `str_to_label(3TSOL)`, `label_encodings(4)`, `attributes(5)`

Trusted Extensions Configuration and Administration

Notes The functionality described on this manual page is available only if the system is configured with Trusted Extensions.

This file is part of the Defense Intelligence Agency (DIA) Mandatory Access Control (MAC) policy. This file might not be applicable to other MAC policies that might be developed for future releases of Solaris Trusted Extensions software.

Name host – DNS lookup utility

Synopsis

```
host [-aCdilmrsTvw] [-c class] [-N ndots] [-R number]
     [-t type] [-W wait]  [-4 | -6] name [server]
```

Description The `host` utility performs simple DNS lookups. It is normally used to convert names to IP addresses and IP addresses to names. When no arguments or options are given, `host` prints a short summary of its command line arguments and options.

The *name* argument is the domain name that is to be looked up. It can also be a dotted-decimal IPv4 address or a colon-delimited IPv6 address, in which case `host` by default performs a reverse lookup for that address. The optional *server* argument is either the name or IP address of the name server that `host` should query instead of the server or servers listed in `/etc/resolv.conf`.

Options The following options are supported:

`-4`
Use only IPv4 transport. By default, both IPv4 and IPv6 transports can be used. Options `-4` and `-6` are mutually exclusive.

`-6`
Use only IPv6 transport. By default, both IPv4 and IPv6 transports can be used. Options `-4` and `-6` are mutually exclusive.

`-a`
Equivalent to setting the `-v` option and asking `host` to make a query of type `ANY`.

`-c` *class*
Make a DNS query of class *class*. This can be used to lookup Hesiod or Chaosnet class resource records. The default class is IN (Internet).

`-C`
Attempt to display the SOA records for zone *name* from all the listed authoritative name servers for that zone. The list of name servers is defined by the NS records that are found for the zone.

`-d`
Generate verbose output. This option is equivalent to `-v`. These two options are provided for backward compatibility. In previous versions, the `-d` option switched on debugging traces and `-v` enabled verbose output.

`-i`
Specifies that reverse lookups of IPv6 addresses should use the IP6.INT domain as defined in RFC 1886. The default is to use RFC 3152 domain IP6.ARPA.

`-l`
List mode. This option makes `host` perform a zone transfer for zone *name*, displaying the NS, PTR and address records (A/AAAA). If combined with `-a`, all records will be displayed. The argument is provided for compatibility with previous implementations. Options `-la` is equivalent to making a query of type `AXFR`.

`-m`
Sets the memory usage debugging flags: record, usage, and trace.

`-N` *ndots*
Set the number of dots that have to be in *name* for it to be considered absolute. The default value is that defined using the ndots statement in `/etc/resolv.conf`, or 1 if no ndots statement is present. Names with fewer dots are interpreted as relative names and will be searched for in the domains listed in the `search` or `domain` directive in `/etc/resolv.conf`.

`-r`
Make a non-recursive query. Setting this option clears the `RD` (recursion desired) bit in the query made by `host`. The name server receiving the query does not attempt to resolve *name*. The `-r` option enables `host` to mimic the behaviour of a name server by making non-recursive queries and expecting to receive answers to those queries that are usually referrals to other name servers.

`-R` *number*
Change the number of UDP retries for a lookup. The *number* argument indicates how many times `host` will repeat a query that does not get answered. The default number of retries is 1. If *number* is negative or zero, the number of retries will default to 1.

`-s`
Specifies that the host not send the query to the next name server if any server responds with a `SERVFAIL` response, which is the reverse of normal stub resolver behavior.

`-t` *type*
Select the query type. The *type* argument can be any recognised query type: `CNAME`, `NS`, `SOA`, `SIG`, `KEY`, and `AXFR`, among others. When no query type is specified, `host` automatically selects an appropriate query type. By default it looks for A, AAAA, and MX records, but if the `-C` option is specified, queries are made for SOA records. If *name* is a dotted-decimal IPv4 address or colon-delimited IPv6 address, `host` queries for PTR records.

If a query type of IXFR is chosen the starting serial number can be specified by appending an equal followed by the starting serial number (for example: `-t IXFR=12345678`).

`-T`
Use a TCP connection when querying the name server. TCP is automatically selected for queries that require it, such as zone transfer (`AXFR`) requests. By default `host` uses UDP when making queries.

`-v`
Generate verbose output. This option is equivalent to `-d`.

`-w`
Wait forever for a reply. The time to wait for a response will be set to the number of seconds given by the hardware's maximum value for an integer quantity.

`-W` *wait*
Wait for *wait* seconds for a reply. If *wait* is less than one, the wait interval is set to one second.

Files `/etc/resolv.conf`
Resolver configuration file

Attributes See for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	network/dns/bind
Interface Stability	Volatile

See Also `dig(1M)`, `named(1M)`, `attributes(5)`

RFC 1035, *RFC 1886*, *RFC 3152*

See the BIND 9 *Administrator's Reference Manual*. As of the date of publication of this man page, this document is available at `https://www.isc.org/software/bind/documentation`.

Name hostconfig – configure a system's host parameters

Synopsis

```
/usr/sbin/hostconfig -p protocol [-d] [ -h] [-n] [-v]
     [-i interface] [-f hostname]
```

Description The `hostconfig` program uses a network protocol to acquire a machine's *host parameters* and set these parameters on the system.

The program selects which protocol to use based on the argument to the required `-p` flag. Different protocols may set different host parameters. Currently, only one protocol (`bootparams`) is defined.

Options The following options are supported:

`-d` Enable `debug` output.

`-f` *hostname* Run the protocol as if this machine were named `hostname`.

`-h` Echo the received `hostname` to `stdout`, rather than setting `hostname` using the system name directly.

`-i` *interface* Use only the named network interface to run the protocol.

`-n` Run the network protocol, but do not set the acquired parameters into the system.

`-p` *protocol* Run `hostconfig` using *protocol*. Currently, only one protocol (`bootparams`) is available. This option is required.

Specifying the `-p bootparams` option uses the `whoami` call of the RPC `bootparams` protocol. This sets the system's `hostname`, `domainname`, and default IP router parameters.

`-v` Enable verbose output.

Examples **EXAMPLE 1** Configuring Host Parameters with Verbose Output

The following command configures a machine's host parameters using the `whoami` call of the RPC `bootparams` protocol with a verbose output.

```
example# hostconfig -p bootparams -v
```

EXAMPLE 2 Displaying Host Parameters

The following command displays the parameters that would be set using the `whoami` call of the RPC `bootparams` protocol.

```
example# hostconfig -p bootparams -n -v
```

EXAMPLE 3 Configuring Host Parameters Less the System Name

The following command configures a machine's host parameters, less the system name, using the `whoami` call of the RPC `bootparams` protocol.

```
example# hostconfig='hostconfig -p bootparams -h'
```

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also

`hostname(1)`, `domainname(1M)`, `route(1M)`, `attributes(5)`

Name hotplug – configure hotplug connectors and ports

Synopsis

```
hotplug list [-c] [-d] [-l] [-v] [path [connection]]

hotplug online path port

hotplug offline [-f] [-q] path port

hotplug enable [-f] [path] connector

hotplug disable [-f] [-q] [path] connector

hotplug poweron [path] connector

hotplug poweroff [-f] [-q] [path] connector

hotplug set -o options [path] connector

hotplug get -ooptions [path] connector

hotplug install path port

hotplug uninstall [-f] [-q] path port

hotplug -?
```

Description The `hotplug` command is used to manage hotplug connections. A connection can be a connector or port. A hotplug connector is a representation of a physical point in the system where components can be inserted or removed. A hotplug port is a representation of a logical point in the system device tree where the connection of a device to the system is managed.

The `hotplug` command only supports hotplug operations on hotplug connectors for PCI Express buses and PCI buses that implement the Standard PCI Hotplug feature. Hotplug ports on PCI Expresss and PCI buses in systems with PCI Express fabrics are also supported. Additional buses may be supported in the future.

The `hotplug` command operates on the following kinds of objects:

`path`
: Hotplug connectors and ports are integrated into the Solaris device tree. The names of connectors and ports are unique relative only to their bus controller. A device path is required to uniquely reference a connector and the parent path of the device is required to uniquely reference a port.

  The `hotplug` command can accept the case that a user specifies only a physical hotplug connector for the state-change operation subcommands. If no such connector exists, the command will fail. If multiple connectors exist with the same name in the system, the state change operations interact with the user to determine which connector needs to be operated upon. See the “Examples” section, below.

`connector`
: If a hardware component supports being physically inserted or removed, then a hotplug connector represents the location where this action may occur. When a connector exists, it has a hierarchy of ports and device nodes that depend upon it.

`port`
All device nodes can be virtually hotplugged, even if their hardware does not support physical hotplugging. A hotplug port exists between a device node and its parent node in the system device tree. It represents the location where the device node and its dependents can be managed.

`connection`
A hotplug connection is a generic term to refer to either a hotplug connector or a hotplug port.

Hotplug connectors and ports are managed according to a state model. The `hotplug` command can list information about the hotplug connections in a system, or it can initiate change of state operations on specific hotplug connections.

Hotplug connectors can be in the following states:

`empty`
A component is not physically inserted in the connector.

`present`
A component is physically inserted in the connector, but the component is powered off. The component is not in use.

`powered`
A component is physically inserted in the connector, and the component is powered on. The component is disabled and is not in use.

`enabled`
A component is physically inserted in the connector. The component is powered on and has been probed and tested. The component is enabled and devices that represent its functions can be used.

Hotplug ports can be in the following states:

`port-empty`
No device exists for the hotplug port.

`port-present`
A device exists for the hotplug port, but the device has not been probed and it has no attached device driver. The device is not in use.

`offline`
A device exists for the hotplug port, and the device has been probed. A device driver is not attached, and the device is not in use.

`online`
A device exists for the hotplug port, and its device driver is fully attached. The device is in use.

`maintenance`
A device exists for the hotplug port, and its device driver is fully attached. The device is in use, but not fully operational. A maintenance or fault management operation is affecting the device. The reason that caused the device to enter maintenance state may vary. It is described by a sub-state under maintenance state. Currently there is only one sub-state defined:

`maintenance-suspended`
The device is live suspended.

The `hotplug` command can also access bus private properties for each hotplug connector. The current values of bus private properties can be displayed. New values for each bus private property can be set directly.

Sub-commands The following subcommands are supported:

`list`
Show information for hotplug connectors, ports, and their associated devices. Hotplug connectors and hotplug ports are integrated into the Solaris device tree hierarchy. The `list` subcommand therefore displays the hierarchy of device nodes with additional information included to show the locations of hotplug connectors and hotplug ports. The names of hotplug connectors are enclosed in square brackets, and the names of hotplug ports are enclosed in angled brackets. The current state of each hotplug connection is displayed next to its name.

`online`
Change the state of a hotplug port to the `online` state.

`offline`
Change the state of a hotplug port to the `offline` state.

`enable`
Change the state of a hotplug connector to the `enabled` state. All of the hotplug connector's dependent ports will be automatically probed and initialized into the `online` state.

If hardware errors occur while probing a connected device, details of the errors are reported to the Solaris Fault Manager for diagnosis. Depending upon the errors, it may be possible to ignore them with a forced operation, using the `-f` option.

`disable`
Change the state of a hotplug connector from the `enabled` state to the `powered` state. All dependent ports that are in the `online` state will first be transitioned to the `port-present` state.

`poweron`
Change the state of a hotplug connector from the `present` state to the `powered` state.

`poweroff`
Change the state of a hotplug connector from the `powered` or `enabled` state to the `present` state. All dependent ports that are in the `online` state will first be transitioned to the `port-present` state, and will then be removed.

`set`
Set bus-specific properties for a hotplug connector. The specified option string is a bus specific string of name and value pairs, as could be parsed by `getsubopt(3C)`. The names and values will be passed directly to the bus controller that manages the specified hotplug connector to perform a bus-specific function.

`get`
Display the current values of bus specific properties for a hotplug connector. The specified option string is a bus specific string of named properties, as could be parsed by `getsubopt(3C)`. The names will be passed directly to the bus controller to specify which properties should be returned. The current values of each named property will then be displayed.

The `install` and `uninstall` subcommands install and uninstall services the drivers of the ports can support.

`install`
The `install` subcommand install services the drivers of the hotplug port's device can support.

For example, this subcommand can be applied to the port of physical function of PCIe IO virtualization devices. It upgrades the port to `ONLINE` state (if it is not yet in that state) and then installs the virtual functions that the physical function (driver) supports. New hotplug ports will be created to represent each virtual function as a dependent of the specified physical function. The newly created ports will be initiated to `OFFLINE` state.

`uninstall`
The `uninstall` subcommand uninstall services the drivers of the hotplug port's device can support.

This subcommand can be applied to the ports of physical functions of PCIe IO virtualization devices. If the specified hotplug port has any dependent ports of virtual functions, the dependent ports and corresponding virtual function nodes will be removed.

Options The following options are supported:

`-l`, `--list-path`
Show full paths to connections and device nodes. By default, the `list` subcommand shows hotplug connectors, ports, and devices in the format of a tree. This option enables the display of full paths to each connection and device node.

`-c`, `--connectors`
Display a table that summarizes the current status of all physical hotplug connectors. Device topologies and hierachical information are not included. In general, the names of

physical hotplug connectors should be unique. If multiple connectors of the same name exist in the system, `hotplug` displays a message specifying the same-named hotplug connectors and prompts for information to distinguish among those connectors. See "Examples," below.

`-d`, `--drivers`
Show the binding driver names and the instance number of the device nodes. By default, the `list` subcommand shows only hotplug connectors, ports, and devices. This option enables the display of the binding driver names and the instance numbers of the device nodes.

`-v`, `--verbose`
Show verbose usage details. By default, the `list` subcommand shows only hotplug connectors, ports, and devices. This option enables the display of more detailed information about how the devices are currently consumed. Examples include mounted filesystems or plumbed network interfaces associated with individual devices.

Note that the `-v` option does not display information for disks under ZFS control.

`-f`, `--force`
Force the operation. Some change state operations that impact resources currently in use will fail with a warning. A forced operation will attempt to ignore these warnings and proceed.

This option should be used with extreme caution.

`-n`, `--non-interactive`
Disable the interactive feature. If the flag is specified and an ambiguous input is encountered, the command will exit immediately, returning a unique exit status to indicate the problem.

`-q`, `--query`
Query the operation. Instead of actually performing a change state operation, perform a test to predict if the operation would succeed or fail. If it would fail, show the error messages that would be expected if the operation had really been attempted.

It is not possible to predict every failure. An operation that succeeds during a query could still fail for another reason when actually attempted.

This option will not actually change the state of the system.

`-o` *options*, `--=`*options*
Specify bus-specific properties for a `set` or `get` command. The options string conforms to the `getsubopt(3C)` syntax convention.

For the `get` subcommand, there are two special options that can be used. The special options value of `help` will display all supported properties and their possible values. The special options value of `all` will display the current value of all supported properties.

For the `set` subcommand, there is one special option that can be used. The special options value of `help` will display all supported properties which can be set and their possible values.

See "Notes" section for the properties supported by bus controllers.

`-?`, `--help`
Display a brief help message on proper use of the command.

Examples

EXAMPLE 1 Showing All Hotplug Connections

The following command shows all hotplug connections:

```
# hotplug list -v
pci@0,0
pci108e,534a@2,1 <pci.2,1> ONLINE
        [pci30] EMPTY
pci10de,5d@e <pci.e,0> ONLINE
        display@b <pci.b,0> ONLINE
        [NEM0] ENABLED
        pci108e,534a@a,0 <pci.a,0> ONLINE
                { Network interface nge0 }
                { nge0: hosts IP addresses: 10.0.0.1 }
        pci108e,534a@a,1 <pci.a,1> (MAINTENANCE)
                           [NEM1] (EMPTY)
pci108e,534a@c <pci.c,0> OFFLINE
pci108e,534a@d <pci.d,0> ONLINE
                         pci1028,40d@0 <pci.0,0> (MAINTENANCE-SUSPENDED,
"activities=dma+pio+intr,reason=resource-rebalance")
                                 { Network interface bge0 }
                                 { bge0: hosts IP addresses: 10.0.1.1 }
```

To show the full paths of hotplug connections and devices, enter the following command:

```
# hotplug list -l
/pci@0,0
/pci@0,0  <pci.0,0>  OFFLINE
/pci@0,0/pci108e,4341
/pci@0,0  <pci.1,0>  OFFLINE
/pci@0,0/pci8086,3408
/pci@0,0  <pci.3,0>  ONLINE
/pci@0,0/pci8086,340a@3
/pci@0,0/pci8086,340a@3 [Slot2]  EMPTY
/pci@0,0  <pci.5,0>  ONLINE
/pci@0,0/pci8086,340c@5
/pci@0,0/pci8086,340c@5 [Slot1]  ENABLED
/pci@0,0/pci8086,340c@5  <pci.0,0>  ONLINE
/pci@0,0/pci8086,340c@5/pci111d,8018@0
/pci@0,0  <pci.7,0>  ONLINE
/pci@0,0/pci8086,340e@7
```

EXAMPLE 1 Showing All Hotplug Connections *(Continued)*

```
/pci@0,0/pci8086,340e@7 [pcie4]  ENABLED
/pci@0,0  <pci.9,0>  ONLINE
/pci@0,0/pci8086,3410@9
/pci@0,0/pci8086,3410@9 [pcie3]  ENABLED
/pci@0,0  <pci.13,0>  OFFLINE
/pci@0,0/pci8086,342d
/pci@0,0  <pci.14,0>  OFFLINE
/pci@0,0/pci8086,342e
```

To show the binding driver names and instance numbers of the devices, enter the following command:

```
# hotplug list -d
pci@0,0  npe#0
    pci108e,4341  <pci.0,0>  OFFLINE  #0
    pci8086,3408  <pci.1,0>  OFFLINE  pcieb#0
    pci8086,340a@3  <pci.3,0>  ONLINE  pcieb#1
        [Slot2]  EMPTY
    pci8086,340c@5  <pci.5,0>  ONLINE  pcieb#2
        [Slot1]  ENABLED
        pci111d,8018@0  <pci.0,0>  ONLINE  pcieb#0
            pci111d,8018@2  <pci.2,0>  ONLINE  pcieb#6
                pci108e,f1bc@0  <pci.0,0>  ONLINE  e1000g#0
                pci108e,f1bc@0,1  <pci.0,1>  OFFLINE  e1000g#1
    pci8086,340e@7  <pci.7,0>  ONLINE  pcieb#3
        [pcie4]  ENABLED
    pci8086,3410@9  <pci.9,0>  ONLINE  pcieb#4
        [pcie3]  ENABLED
    pci8086,342d  <pci.13,0>  OFFLINE  #0
    pci8086,342e  <pci.14,0>  OFFLINE  #0
```

To show the status of all physical hotplug connectors. enter the following command:

```
# hotplug list -c
Connection             State         Description
-------------------------------------------------------
NEM0                   ENABLED       PCIe-Native
PCI-EM0                ENABLED       PCIe-Native
NEM1                   ENABLED       PCIe-Native
PCI-EM1                ENABLED       PCIe-Native
```

The table in the condition that multiple connectors with the same exist in the system:

```
# hotplug list -c
Connection   Note      State         Description
-----------------------------------------------------------
NEM0                   ENABLED       PCIe-Native
PCI-EM0      *1        ENABLED       PCIe-Native
```

EXAMPLE 1 Showing All Hotplug Connections *(Continued)*

```
PCI-EM0      *2        ENABLED       PCIe-Native
NEM1                   ENABLED       PCIe-Native

Note:
Multiple connectors with the same name exist:

  [1] PCI-EM0   /pci@0,0/pci108e,4341@1a,2
  [2] PCI-EM0   /pci@1,0/pci108e,4341@1a,2
```

EXAMPLE 2 Reporting Failure During State Change Operation

If a change of state operation fails, an explanation is displayed to describe the failure. An attempt to offline a hotplug port with dependent devices that are currently in use by the system might fail as follows:

```
# hotplug offline /pci@0,0/pci10de,5d@e pci.a,0
ERROR: devices or resources are busy.
pci108e,534a@a,0:
    { Network interface nge0 }
    { nge0: hosts IP addresses: 10.0.0.1 }
    { Plumbed IP Address }
```

EXAMPLE 3 Enabling User Interaction During State Change Operation

If there are multiple connectors with the same name in a system, the state change operations interact with the user to specify which connector needs to be operated upon.

```
# hotplug enable PCI-EM0
Multiple connectors with the same name exist:

   [1] PCI-EM0  /pci@0,0/pci108e,4341@1a,2
   [2] PCI-EM0  /pci@1,0/pci108e,4341@1a,2

Please select a connector, then press ENTER:
```

EXAMPLE 4 Displaying Bus-Specific Properties and Values

The following command displays all supported bus-specific properties and their possible values:

```
# hotplug get -o help /pci@0,0 pci.2,1
power_led=<on|off|blink>
fault_led=<on|off|blink>
active_led=<on|off|blink>
attn_led=<on|off|blink>
card_type=<type description>
board_type=<type description>
```

EXAMPLE 5 Displaying Bus-Specific Options

The following command displays the card type and the current state of the Power LED of a PCI hotplug connector:

```
# hotplug get -o card_type,power_led /pci@0,0 pci.2,1
card_type=fibre
power_led=on
```

EXAMPLE 6 Setting a Bus-Specific Property

The following command turns on the attention LED of a PCI hotplug connector:

```
# hotplug set -o attn_led=on /pci@0,0 pci.2,1
```

EXAMPLE 7 Installing Port Dependents

The following commands install dependent ports of an IOV physical function and then display the resulting IOV virtual functions that were probed.

```
# hotplug install /pci@400/pci@1/pci@0/pci@4 pci.0,1

# hotplug list -v /pci@400/pci@1/pci@0/pci@4 pci.0,1
<pci.0,1>  (ONLINE)
     { IOV physical function }
     { IOV virtual function 'pci.0,81' }
     { IOV virtual function 'pci.0,83' }
     { IOV virtual function 'pci.0,85' }
     { IOV virtual function 'pci.0,87' }
<pci.0,81>  (OFFLINE)
ethernet@0,81
<pci.0,83>  (OFFLINE)
ethernet@0,83
<pci.0,85>  (OFFLINE)
ethernet@0,85
<pci.0,87>  (OFFLINE)
ethernet@0,87
```

EXAMPLE 8 Uninstalling Port Dependents

The following command attempts to uninstall dependent ports of an IOV physical function, but fails when a dependent IOV virtual function is busy.

```
# hotplug uninstall /pci@400/pci@1/pci@0/pci@4 pci.0,0
ERROR: devices or resources are busy.
ethernet@0,81:
    { Network interface igvbf1 }
    { igvbf1: hosts IP addresses: 10.0.0.1 }
    { Plumbed IP Address }
```

Exit Status

`0`
Successful completion.

`1`
Invalid command line options were specified.

`2`
The specified path or connection does not exist.

`3`
A fatal error occurred. One or more error messages are displayed on standard error.

`4`
The hotplug service is not available.

`5`
The connector name specified refers to multiple ports.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os
Interface Stability	Committed

See Also `cfgadm(1M)`, `hotplugd(1M)`, `getsubopt(3C)`, `attributes(5)`

Diagnostics The following error message is displayed on systems that do not have any supported IO buses:

```
ERROR: there are no connections to display.
(See hotplug(1m) for more information.)
```

If this error message is seen, note that the system might still have other IO devices that support hotplugging, through the `cfgadm(1M)` command instead of `hotplug`.

The following error message is displayed if hardware errors occurred while processing an enable operation:

```
ERROR: hardware or driver specific failure.
```

If this error message is seen, note that additional details of the hardware errors may have been reported to the Solaris Fault Manager for diagnosis.

Notes The `hotplug` service (FMRI `svc:/system/hotplug`) must be enabled as a prerequisite for using the `hotplug` command. The service is disabled by default. See `hotplugd(1M)`.

The authorization `solaris.hotplug.modify` must be granted in order to perform change-of-state operations, or to install and uninstall dependent ports. Alternatively, the rights profile “Hotplug Management” can be granted, which includes that authorization.

Verbose usage information is gathered from the RCM framework. Its format and content is subject to change.

The following bus specific properties are supported in PCI bus controllers:

`power_led` | `fault_led` | `attn_led` | `active_led`
States of a specific LED of a slot. The value could be `on`, `off`, or `blink`.

They can all be used with `get` subcommand, but only property `attn_led` can be used with `set` subcommand.

`card_type` | `board_type`
Type of a card or board of a slot.

They can all be used with `get` subcommand, but neither can be used with `set` subcommand.

Name hotplugd – hotplug daemon

Synopsis

```
/usr/lib/hotplugd [-d]
```

Description The hotplug daemon, `hotplugd`, provides user-level services for the management of hotplug connections. It is a system daemon started by the Service Management Facility (see `smf(5)`). Its fault management resource identifier (FMRI) is:

```
svc:/system/hotplug:default
```

Note that `hotplugd` is a Consolidation Private interface. See `attributes(5)`.

The `hotplug(1M)` command and any other client program that uses the private `libhotplug` library to query information about hotplug connections or initiate hotplug commands depends upon this daemon. The hotplug daemon is a door server which services requests from all `libhotplug` clients. The door interface is private.

Client applications use the private `libhotplug` interface to administer hotplug connections. `libhotplug` uses the door interface to administer hotplug connections through the hotplug daemon service. The hotplug daemon acts as a central location to serialize all hotplug operations and coordinate activities with all other parts of the system.

Options The following option is supported:

`-d`, `--debug`
: Run the daemon in standalone debug mode. Messages will be displayed on the controlling terminal instead of to syslog. And increased verbosity will be enabled to display more details about the internal operations of the daemon.

Examples **EXAMPLE 1** Enabling the Hotplug Service

The following command enables the hotplug service:

```
# svcadm enable svc:/system/hotplug:default
```

EXAMPLE 2 Disabling the Hotplug Service

The following command disables the hotplug service:

```
# svcadm disable svc:/system/hotplug:default
```

Errors The hotplug daemon uses `syslog(3C)` to report status and error messages. All of the messages are logged with the `LOG_DAEMON` facility. Error messages are logged with the `LOG_ERR` and `LOG_NOTICE` priorities, and informational messages are logged with the `LOG_INFO` priority. The default entries in the `/etc/syslog.conf` file log all of the hotplug daemon error messages to the `/var/adm/messages` log.

Files

`/var/run/hotplugd_door`
: Hotplug daemon door

`/var/run/hotplugd_pid`
: Hotplug daemon lock file

`/usr/lib/hotplugd`
Hotplug daemon binary

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os
Interface Stability	Consolidation Private

See Also `svcs(1)`, `hotplug(1M)`, `svcadm(1M)`, `syslog(3C)`, `syslog.conf(4)`, `attributes(5)`, `smf(5)`

Notes The `hotplugd` service is managed by the service management facility, `smf(5)`, under the fault management resource identifier (FMRI):

```
svc:/system/hotplug:default
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. The service's status can be queried using the `svcs(1)` command. To administer the service, the authorization `solaris.smf.manage.hotplug` must be granted. Alternatively, the rights profile “Hotplug Management” can be granted.

The hotplug service must be enabled for the `hotplug(1M)` command and any other `libhotplug` client applications to function properly.

Name htable – convert DoD Internet format host table

Synopsis `/usr/sbin/htable` *filename*

Description `htable` converts a host table in the format specified by RFC 952 to the format used by the network library routines. Three files are created as a result of running `htable`: `hosts`, `networks`, and `gateways`. The `hosts` file is used by the `gethostbyname(3NSL)` routines in mapping host names to addresses. The `networks` file is used by the `getnetbyname(3SOCKET)` routines in mapping network names to numbers. The `gateways` file is used by the routing daemon to identify "passive" Internet gateways.

If any of the files `localhosts`, `localnetworks`, or `localgateways` are present in the current directory, the file's contents is prepended to the output file without interpretation. This allows sites to maintain local aliases and entries which are not normally present in the master database.

`htable` is best used in conjunction with the `gettable(1M)` program which retrieves the DoD Internet host table from a host.

Files `localhosts`

`localnetworks`

`localgateways`

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/network/nis

See Also `gettable(1M)`, `gethostbyname(3NSL)`, `getnetbyname(3SOCKET)`, `attributes(5)`
Harrenstien, Ken, Mary Stahl, and Elizabeth Feinler, *DoD Internet Host Table Specification*, RFC 952, Network Information Center, SRI International, Menlo Park, California, October 1985.

Notes `htable` does not properly calculate the `gateways` file.

Name ickey – install a client key for WAN boot

Synopsis

```
/usr/lib/inet/wanboot/ickey [-d] [-o type=3des]

/usr/lib/inet/wanboot/ickey [-d] [-o type=aes]

/usr/lib/inet/wanboot/ickey [-d] [-o type=sha1]
```

Description The `ickey` command is used to install WAN boot keys on a running UNIX system so that they can be used the next time the system is installed. You can store three different types of keys: 3DES and AES for encryption and an HMAC SHA-1 key for hashed verification.

`ickey` reads the key from standard input using `getpass(3C)` so that it does not appear on the command line. When installing keys on a remote system, you must take proper precautions to ensure that any keying materials are kept confidential. At a minimum, use `ssh(1)` to prevent interception of data in transit.

Keys are expected to be presented as strings of hexadecimal digits; they can (but need not) be preceded by a `0x` or `0X`.

The `ickey` command has a single option, described below. An argument of the type `-o type=`*keytype* is required.

Options The `ickey` command the following option.

`-d` Delete the key specified by the *keytype* argument.

Exit Status On success, `ickey` exits with status 0; if a problem occurs, a diagnostic message is printed and `ickey` exits with non-zero status.

Files `/dev/openprom` WAN boot key storage driver

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/boot/wanboot
Interface Stability	Uncommitted

See Also `ssh(1)`, `openprom(7D)`, `attributes(5)`

Name id – return user identity

Synopsis

```
/usr/bin/id [-p] [user]
/usr/bin/id -a [-p] [user]
/usr/bin/id -G [-n] [user]
/usr/bin/id -g [-nr] [user]
/usr/bin/id -u [-nr] [user]
/usr/xpg4/bin/id [-p] [user]
/usr/xpg4/bin/id -a [-p] [user]
/usr/xpg4/bin/id -G [-n] [user]
/usr/xpg4/bin/id -g [-nr] [user]
/usr/xpg4/bin/id -u [-nr] [user]
```

Description If no *user* operand is provided, the `id` utility writes the user and group IDs and the corresponding user and group names of the invoking process to standard output. If the effective and real IDs do not match, both are written. If multiple groups are supported by the underlying system, `/usr/xpg4/bin/id` also writes the supplementary group affiliations of the invoking process.

If a *user* operand is provided and the process has the appropriate privileges, the user and group IDs of the selected user are written. In this case, effective IDs are assumed to be identical to real IDs. If the selected user has more than one allowable group membership listed in the group database, `/usr/xpg4/bin/id` writes them in the same manner as the supplementary groups described in the preceding paragraph.

Formats The following formats are used when the `LC_MESSAGES` locale category specifies the "C" locale. In other locales, the strings `uid`, `gid`, `euid`, `egid`, and `groups` may be replaced with more appropriate strings corresponding to the locale.

"`uid=%u(%s) gid=%u(%s)\n`" <*real user ID*>, <*user-name*>,
<*real group ID*>, <*group-name*>

If the effective and real user IDs do not match, the following are inserted immediately before the `\n` character in the previous format:

" `euid=%u(%s)`"

with the following arguments added at the end of the argument list:

<*effective user ID*>, <*effective user-name*>

If the effective and real group IDs do not match, the following is inserted directly before the `\n` character in the format string (and after any addition resulting from the effective and real user IDs not matching):

" `egid=%u(%s)`"

with the following arguments added at the end of the argument list:

<*effectivegroup-ID*>, <*effectivegroupname*>

If the process has supplementary group affiliations or the selected user is allowed to belong to multiple groups, the first is added directly before the NEWLINE character in the format string:

" groups=%u(%s)"

with the following arguments added at the end of the argument list:

<*supplementary group ID*>, <*supplementary group name*>

and the necessary number of the following added after that for any remaining supplementary group IDs:

",%u(%s)"

and the necessary number of the following arguments added at the end of the argument list:

<*supplementary group ID*>, <*supplementary group name*>

If any of the user ID, group ID, effective user ID, effective group ID or supplementary/multiple group IDs cannot be mapped by the system into printable user or group names, the corresponding (%s) and name argument is omitted from the corresponding format string.

When any of the options are specified, the output format is as described under OPTIONS.

Options The following options are supported by both /usr/bin/id and /usr/xpg4/bin/id. The -p and -a options are invalid if specified with any of the -G, -g, or -u options.

-p Reports additionally the current project membership of the invoking process. The project is reported using the format:

"projid=%u(%s)"

which is inserted prior to the \n character of the default format described in the Formats section. The arguments

<*project ID*>,<*project name*>

are appended to the end of the argument list. If the project ID cannot be mapped by the system into a printable project name, the corresponding (%s) and name argument is omitted from the corresponding format string.

-a Reports user name, user ID and all the groups to which the user belongs.

-G Outputs all different group IDs (effective, real and supplementary) only, using the format "%u\n". If there is more than one distinct group affiliation, output each such affiliation, using the format " %u", before the NEWLINE character is output.

-g Outputs only the effective group ID, using the format "%u\n".

-n Outputs the name in the format "%s" instead of the numeric ID using the format "%u".

`-r` Outputs the real ID instead of the effective ID.

`-u` Outputs only the effective user ID, using the format "`%u\n`".

Operands The following operand is supported:

user The user (login) name for which information is to be written.

Environment Variables See `environ(5)` for descriptions of the following environment variables that affect the execution of `id`: `LANG`, `LC_ALL`, `LC_CTYPE`, `LC_MESSAGES`, and `NLSPATH`.

Exit Status The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

Attributes See `attributes(5)` for descriptions of the following attributes:

/usr/bin/id

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os, system/library/processor
Interface Stability	Committed

/usr/xpg4/bin/id

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/xopen/xcu4
Interface Stability	Standard

See Also `fold(1)`, `logname(1)`, `who(1)`, `getgid(2)`, `getgroups(2)`, `getprojid(2)`, `getuid(2)`, `attributes(5)`, `environ(5)`, `standards(5)`

Notes Output produced by the `-G` option and by the default case could potentially produce very long lines on systems that support large numbers of supplementary groups.

Name idmap – configure and manage the Native Identity Mapping service

Synopsis

```
idmap

idmap -f command-file

idmap add [-d] name1 name2

idmap dump [-n] [-v]

idmap export [-f file] format

idmap flush [-a]

idmap get-namemap name

idmap help

idmap import [-F] [-f file] format

idmap list

idmap remove [-t|-f] name

idmap remove -a

idmap remove [-d] name1 name2

idmap set-namemap [-a authenticationMethod] [-D bindDN]
     [-j passwdfile] name1 name2

idmap show [-c] [-v] [-V] identity [target-type]

idmap unset-namemap [-a authenticationMethod] [-D bindDN]
     [-j passwdfile] name [target-type]
```

Description The `idmap` utility is used to configure and manage the Native Identity Mapping service.

The Native Identity Mapping service supports the following types of mappings between Windows security identitier (SIDs) and POSIX user IDs and group IDs (UIDs and GIDs):

- **Name-based mapping.** An administrator maps Windows and UNIX users and groups by name.
- **Ephemeral ID mapping.** A UID or GID is dynamically allocated for every SID that is not already mapped by name.
- **Local-SID mapping.** A non-ephemeral UID or GID is mapped to an algorithmically generated local SID.

The `idmap` utility can be used to create and manage the name-based mappings and to monitor the mappings in effect.

If the `idmap` utility is invoked without a subcommand or option, it reads the subcommands from standard input. When standard input is a TTY, the `idmap` command prints the usage message and exits.

Mapping Mechanisms

The idmapd(1M) daemon maps Windows user and group SIDs to UNIX UIDs and GIDs as follows:

1. SIDs are mapped by name.

   This mapping uses the name-based mappings that are manually set up by the system administrator.

2. If no name-based mapping is found, the SID is mapped to a dynamically allocated ephemeral ID.

   This allocation uses the next available UID or GID from 2^{31} to 2^{32} - 2.

Local SID mappings are used to map from UNIX to Windows.

To prevent aliasing problems, all file systems, archive and backup formats, and protocols must store SIDs or map all UIDs and GIDs in the 2^{31} to 2^{32} - 2 range to the `nobody` user and group.

It is possible to create also diagonal mappings. They are the mappings between Windows groups and Solaris users and between Solaris groups and Windows users. They are needed when Windows uses a group identity as a file owner or vice versa.

Name-based Mappings

Name-based mappings establish name equivalence between Windows users and groups and their counterparts in the UNIX name service. These mappings persist across reboots. For example, the following command maps Windows users to UNIX users with the same name:

```
# idmap add "winuser:*@mywindomain.com" "unixuser:*"
```

If configured to use a directory service, idmapd(1M) will first try to use the mapping information that is stored in user or group objects in the Active Directory (AD) and/or the native LDAP directory service. For example, an AD object for a given Windows user or group can be augmented to include the corresponding Solaris user or group name or numeric id. Similarly, the native LDAP object for a given Solaris user or group can be augmented to include the corresponding Windows user or group name.

idmapd(1M) can be configured to use AD and/or native LDAP directory-based name mappings by setting the appropriate service management facility (SMF) properties of the `idmap` service. See "Service Properties," below, for more details.

If directory-based name mapping is not configured or if configured but not found, then idmapd(1M) will process locally stored name-based mapping rules.

`idmap` supports the mapping of Windows well-known names. A few of these are listed below:

```
Administrator
Guest
KRBTGT
Domain Admins
Domain Users
Domain Guest
Domain Computers
Domain Controllers
```

When `idmap` rules are added, these well-known names will be expanded to canonical form. That is, either the default domain name will be added (for names that are not well-known) or an appropriate built-in domain name will be added. Depending on the particular well-known name, this domain name might be null, `BUILTIN`, or the local host name.

The following sequence of `idmap` commands illustrate the treatment of the non-well-known name `fred` and the well-known names `administrator` and `guest`.

```
# idmap add winname:fred unixuser:fredf
add     winname:fred    unixuser:fredf

# idmap add winname:administrator unixuser:root
add     winname:administrator   unixuser:root

# idmap add winname:guest unixuser:nobody
add     winname:guest   unixuser:nobody

# idmap add wingroup:administrators sysadmin
add     wingroup:administrators unixgroup:sysadmin

# idmap list
add     winname:Administrator@examplehost  unixuser:root
add     winname:Guest@examplehost  unixuser:nobody
add     wingroup:Administrators@BUILTIN unixgroup:sysadmin
add     winname:fred@example.com       unixuser:fredf
```

Ephemeral Mappings

The `idmapd` daemon attempts to preserve ephemeral ID mappings across daemon restarts. However, when IDs cannot be preserved, the daemon maps each previously mapped SID to a new ephemeral UID or GID value. The daemon will never re-use ephemeral UIDs or GIDs. If the `idmapd` daemon runs out of ephemeral UIDs and GIDs, it returns an error as well as a default UID or GID for SIDs that cannot be mapped by name.

The dynamic ID mappings are not retained across reboots. So, any SIDs that are dynamically mapped to UNIX UIDs or GIDs are most likely mapped to different IDs after rebooting the system.

Local SID Mappings

If no name-based mapping is found, a non-ephemeral UID or GID is mapped to an algorithmically generated local SID. The mapping is generated as follows:

`local SID for UID =` <*machine SID*> `-` <*1000 + UID*>
`local SID for GID =` <*machine SID*> `-` <*2^31 + GID*>

<*machine SID*> is a unique SID generated by the `idmap` service for the host on which it runs.

Rule Lookup Order

When mapping a Windows name to a UNIX name, lookup for name-based mapping rules is performed in the following order:

1. *windows-name*@*domain* to ""
2. *windows-name*@*domain* to *unix-name*

3. *windows-name*@* to ""
4. *windows-name*@* to *unix-name*
5. *@*domain* to *
6. *@*domain* to ""
7. *@*domain* to *unix-name*
8. *@* to *
9. *@* to ""
10. *@* to *unix-name*

When mapping a UNIX name to a Windows name, lookup for name-based mapping rules is performed in the following order:

1. *unix-name* to ""
2. *unix-name* to *windows-name*@*domain*
3. * to *@*domain*
4. * to ""
5. * to *windows-name*@*domain*

Service Properties

The service properties determine the behavior of the `idmapd(1M)` daemon. These properties are stored in the SMF repository (see `smf(5)`) under property group `config`. They can be accessed and modified using `svccfg(1M)`, which requires `solaris.smf.value.idmap` authorization. The service properties for the `idmap` service are:

`config/ad_unixuser_attr`
: Specify the name of the AD attribute that contains the UNIX user name. There is no default.

`config/ad_unixgroup_attr`
: Specify the name of the AD attribute that contains the UNIX group name. There is no default.

`config/nldap_winname_attr`
: Specify the name of the Native LDAP attribute that contains the Windows user/group name. There is no default.

`config/directory_based_mapping`
: Controls support for identity mapping using data stored in a directory service.

  `none` disables directory-based mapping.

  `name` enables name-based mapping using the properties described above.

  `idmu` enables mapping using Microsoft's Identity Management for UNIX (IDMU). This Windows component allows the administrator to specify a UNIX user ID for each Windows user, mapping the Windows identity to the corresponding UNIX identity. Only IDMU data from the domain the Solaris system is a member of is used.

Changes to service properties do not affect a running `idmap` service. The service must be refreshed (with `svcadm(1M)`) for the changes to take effect.

Operands The `idmap` command uses the following operands:

format
Specifies the format in which user name mappings are described for the `export` and `import` subcommands. The Netapp `usermap.cfg` and Samba `smbusers` external formats are supported. These external formats are *only* for users, not groups.

- The `usermap.cfg` rule-mapping format is as follows:

  windows-username `[`*direction*`]` *unix-username*

  windows-username is a Windows user name in either the *domain**username* or *username*@*domain* format.

  unix-username is a UNIX user name.

  direction is one of the following:

  - == means a bidirectional mapping, which is the default.
  - => or <= means a unidirectional mapping.

  The IP qualifier is not supported.

- The `smbusers` rule-mapping format is as follows:

  unixname `=` *winname1 winname2* `...`

  If *winname* includes whitespace, escape the whitespace by enclosing the value in double quotes. For example, the following file shows how to specify whitespace in a valid format for the `idmap` command:

  ```
  $ cat myusermap
  terry="Terry Maddox"
  pat="Pat Flynn"
  cal=cbrown
  ```

  The mappings are imported as unidirectional mappings from Windows names to UNIX names.

  The format is based on the "username map" entry of the `smb.conf` man page, which is available on the `samba.org` web site. The use of an asterisk (*) for *windows-name* is supported. However, the `@group` directive and the chaining of mappings are not supported.

  By default, if no mapping entries are in the `smbusers` file, Samba maps a *windows-name* to the equivalent *unix-name*, if any. If you want to set up the same mapping as Samba does, use the following `idmap` command:

  ```
  idmap add -d "winuser:*@*" "unixuser:*"
  ```

identity
Specifies a user name, user ID, group name, or group ID. *identity* is specified as *type*:*value*. *type* is one of the following:

`usid`	Windows user SID in text format
`gsid`	Windows group SID in text format
`sid`	Windows group SID in text format that can belong either to a user or to a group
`uid`	Numeric POSIX UID
`gid`	Numeric POSIX GID
`unixuser`	UNIX user name
`unixgroup`	UNIX group name
`winuser`	Windows user name
`wingroup`	Windows group name
`winname`	Windows user or group name

value is a number or string that is appropriate to the specified *type*. For instance, `unixgroup:staff` specifies the UNIX group name, `staff`. The identity `gid:10` represents GID 10, which corresponds to the UNIX group `staff`.

name

Specifies a UNIX name (`unixuser`, `unixgroup`) or a Windows name (`winuser`, `wingroup`) that can be used for name-based mapping rules.

A Windows security entity name can be specified in one of these ways:

- *domain**name*
- *name*@*domain*
- *name*, which uses the default mapping domain

If *name* is the empty string (""), mapping is inhibited. Note that a name of "" should not be used to preclude logins by unmapped Windows users.

If *name* uses the wildcard (*), it matches all names that are not matched by other mappings. Similarly, if *name* is the wildcard Windows name (*@*), it matches all names in all domains that are not matched by other mappings.

If *name* uses the wildcard on both sides of the mapping rule, the name is the same for both Windows and Solaris users. For example, if the rule is "*@domain" == "*", the `jp@domain` Windows user name matches this rule and maps to the `jp` Solaris user name.

Specifying the type of *name* is optional if the type can be deduced from other arguments or types specified on the command line.

target-type

Used with the `show` and `unset-namemap` subcommands. For `show`, specifies the mapping type that should be shown. For example, if *target-type* is `sid`, `idmap show` returns the SID

mapped to the identity specified on the command line. For `unset-namemap`, identifies an attribute within the object specified by the *name* operand.

Options The `idmap` command supports one option and a set of subcommands. The subcommands also have options.

Command-Line Option

`-f` *command-file*
Reads and executes `idmap` subcommands from *command-file*. The `idmap -f -` command reads from standard input. This option is not used by any subcommands.

Subcommands

The following subcommands are supported:

`add` [`-d`] *name1 name2*
Adds a name-based mapping rule. By default, the name mapping is bidirectional. If the `-d` option is used, a unidirectional mapping is created from *name1* to *name2*.

Either *name1* or *name2* must be a Windows name, and the other must be a UNIX name. For the Windows name, the `winname` identity type must not be used. Instead, specify one of the `winuser` or `wingroup` types. See "Operands" for information about the *name* operand.

Note that two unidirectional mappings between the same two names in two opposite directions are equivalent to one bidirectional mapping.

This subcommand requires the `solaris.admin.idmap.rules` authorization.

`dump` [`-n`] [`-v`]
Dumps all the mappings cached since the last system boot. The `-n` option shows the names, as well. By default, only `sids`, `uids`, and `gids` are shown. The `-v` option shows how the mappings were generated.

`export` [`-f` *file*] *format*
Exports name-based mapping rules to standard output in the specified *format*. The `-f` *file* option writes the rules to the specified output file.

`flush` [`-a`]
Flushes the identity mapping cache so that future mapping requests will be fully processed based on the current rules and directory information. This is a non-disruptive operation. A rule change automatically flushes the cache; this manual operation can be used to force newly changed directory information to take effect.

With `-a`, the cache is wiped clean. This operation can potentially disrupt operations that are in process and so should be used only on a quiet system. It should not normally be necessary, but might be appropriate to use `-a` to set up "clean slate" test cases.

`get-namemap` *name*
Get the directory-based name mapping information from the AD or native LDAP user or group object represented by the specified name.

`help`
Displays the usage message.

`import` [`-F`] [`-f` *file*] *format*
Imports name-based mapping rules from standard input by using the specified *format*. The `-f` *file* option reads the rules from the specified file. The `-F` option flushes existing name-based mapping rules before adding new ones.

Regardless of the external format used, the imported rules are processed by using the semantics and order described in the section "Rule Lookup Order," above.

This subcommand requires the `solaris.admin.idmap.rules` authorization.

`list`
Lists all name-based mapping rules. Each rule appears in its `idmap add` form.

`remove` [`-t`|`-f`] *name*
Removes any name-based mapping rule that involves the specified name. *name* can be either a UNIX or Windows user name or group name.

The `-f` option removes rules that use *name* as the source. The `-t` option removes rules that use *name* as the destination. These options are mutually exclusive.

This subcommand requires the `solaris.admin.idmap.rules` authorization.

`remove -a`
Removes all name-based mapping rules.

This subcommand requires the `solaris.admin.idmap.rules` authorization.

`remove` [`-d`] *name1 name2*
Removes name-based mapping rules between *name1* and *name2*. If the `-d` option is specified, rules from *name1* to *name2* are removed.

Either *name1* or *name2* must be a Windows name, and the other must be a UNIX name.

This subcommand requires the `solaris.admin.idmap.rules` authorization.

`set-namemap` [`-a` *authenticationMethod*] [`-D` *bindDN*] [`-j` *passwdfile*] *name1 name2*
Sets name mapping information in the AD or native LDAP user or group object. Either *name1* or *name2* must be a Windows name, and the other must be a UNIX name.

If *name1* is a Windows name, then the UNIX name *name2* is added to the AD object represented by *name1*. Similarly, if *name1* is a UNIX name then the Windows name *name2* is added to the native LDAP entry represented by *name1*.

The following options are supported:

`-a` *authenticationMethod*
Specify authentication method when modifying native LDAP entry. See `ldapaddent(1M)` for details. Default value is `sasl/GSSAPI`.

`-D` *bindDN*
Uses the distinguished name *bindDN* to bind to the directory.

`-j` *passwdfile*
Specify a file containing the password for authentication to the directory.

`show` [`-c`] [`-v`] [`-V`] *name* [*target-type*]
Shows the identity of type, *target-type*, that the specified *name* maps to. If the optional *target-type* is omitted, the non-diagonal mapping is shown.

By default, this subcommand shows only mappings that have been established already. The `-c` option forces the evaluation of name-based mapping configurations or the dynamic allocation of IDs.

The `-v` option shows how the mapping was generated and also whether the mapping was just generated or was retrieved from the cache. The `-V` option details the exact steps taken to determine the mapping, including interim steps and approaches that were rejected.

`unset-namemap` [`-a` *authenticationMethod*] [`-D` *bindDN*] [`-j` *passwdfile*] *name* [*target-type*]
Unsets directory-based name mapping information from the AD or native LDAP user or group object represented by the specified name and optional target type.

See the `set-namemap` subcommand for options.

Examples

EXAMPLE 1 Using a Wildcard on Both Sides of a Name-Based Mapping Rule

The following command maps all Windows user names in the `xyz.com` domain to the UNIX users with the same names provided that one exists and is not otherwise mapped. If such a rule is matched but the UNIX user name does not exist, an ephemeral ID mapping is used.

```
# idmap add "winuser:*@xyz.com" "unixuser:*"
```

EXAMPLE 2 Using a Wildcard on One Side of a Name-Based Mapping Rule

The following command maps all unmapped Windows users in the `xyz.com` domain to the `guest` UNIX user. The `-d` option specifies a unidirectional mapping from `*@xyz.com` users to the `guest` user.

```
# idmap add -d "winuser:*@xyz.com" unixuser:guest
```

EXAMPLE 3 Adding a Bidirectional Name-Based Mapping Rule

The following command maps Windows user, `foobar@example.com`, to UNIX user, `foo`, and conversely:

```
# idmap add winuser:foobar@example.com unixuser:foo
```

This command shows how to remove the mapping added by the previous command:

```
# idmap remove winuser:foobar@example.com unixuser:foo
```

EXAMPLE 4 Showing a UID-to-SID Mapping

- The following command shows the SID that the specified UID, `uid:50000`, maps to:

EXAMPLE 4 Showing a UID-to-SID Mapping *(Continued)*

```
# idmap show uid:50000 sid
uid:50000 -> usid:S-1-5-21-3223191800-2000
```

- The following command shows the UNIX user name that the specified Windows user name, `joe@example.com`, maps to:

```
# idmap show joe@example.com unixuser
winuser:joe@example.com -> unixuser:joes
```

EXAMPLE 5 Listing the Cached SID-to-UID Mappings

The following command shows all of the SID-to-UID mappings that are in the cache:

```
# idmap dump | grep "uid:"
usid:S-1-5-21-3223191800-2000    ==     uid:50000
usid:S-1-5-21-3223191800-2001    ==     uid:50001
usid:S-1-5-21-3223191800-2006    ==     uid:50010
usid:S-1-5-21-3223191900-3000    ==     uid:2147491840
usid:S-1-5-21-3223191700-4000    =>     uid:60001
```

EXAMPLE 6 Batching `idmap` Requests

The following commands show how to batch `idmap` requests. This particular command sequence does the following:

- Removes any previous rules for `foobar@example.com`.
- Maps Windows user `foobar@example.com` to UNIX user `bar` and vice-versa.
- Maps Windows group `members` to UNIX group `staff` and vice-versa.

```
# idmap <<EOF
       remove winuser:foobar@example.com
       add winuser:foobar@example.com unixuser:bar
       add wingroup:members unixgroup:staff
EOF
```

EXAMPLE 7 Listing Name-Based Mapping Rules

The following command shows how to list the name-based mapping rules:

```
# idmap list
add winuser:foobar@example.com unixuser:bar
add wingroup:members unixgroup:staff
```

EXAMPLE 8 Importing Name-Based Mapping Rules From the `usermap.cfg` File

The `usermap.cfg` file can be used to configure name-based mapping rules. The following `usermap.cfg` file shows mapping rules that map Windows user `foo@example.com` to UNIX user `foo`, and that map `foobar@example.com` to the UNIX user `foo`.

EXAMPLE 8 Importing Name-Based Mapping Rules From the `usermap.cfg` File *(Continued)*

```
# cat usermap.cfg
foo@example.com == foo
foobar@example.com => foo
```

The following `idmap` command imports `usermap.cfg` information to the `idmapd` database:

```
# cat usermap.cfg | idmap import usermap.cfg
```

This command does the same as the previous command:

```
# idmap import -f usermap.cfg usermap.cfg
```

The following commands are equivalent to the previous `idmap import` commands:

```
# idmap <<EOF
       add winuser:foo@example.com unixuser:foo
       add -d winuser:foobar@example.com unixuser:foo
EOF
```

EXAMPLE 9 Using Name-Based and Ephemeral ID Mapping With Identity Function Mapping and Exceptions

The following commands map all users in the `example.com` Windows domain to UNIX user accounts of the same name. The command also specifies mappings for the following Windows users: `joe@example.com`, `jane.doe@example.com`, `administrator@example.com`. The `administrator` from all domains is mapped to `nobody`. Any Windows users without corresponding UNIX accounts are mapped dynamically to available ephemeral UIDs.

```
# idmap import usermap.cfg <<EOF
       joe@example.com == joes
       jane.doe@example.com == janed
       administrator@* => nobody
       *@example.com == *
       *@example.com => nobody
EOF
```

EXAMPLE 10 Adding Directory-based Name Mapping to AD User Object

The following command maps Windows user `joe@example.com` to UNIX user `joe` by adding the UNIX name to AD object for `joe@example.com`.

```
# idmap set-namemap winuser:joe@example.com joes
```

EXAMPLE 11 Adding Directory-based Name Mapping to Native LDAP User Object

The following command maps UNIX user `foo` to Windows user `foobar@example.com` by adding the Windows name to native LDAP object for `foo`.

```
# idmap set-namemap unixuser:foo foobar@example.com
```

EXAMPLE 12 Removing Directory-based Name Mapping from AD User Object

The following command removes the UNIX username `unixuser` from the AD object representing `joe@example.com`.

```
# idmap unset-namemap winuser:joe@example.com unixuser
```

Exit Status

`0` Successful completion.

`>0` An error occurred. A diagnostic message is written to standard error.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os
Interface Stability	Uncommitted

See Also

`svcs(1)`, `idmapd(1M)`, `ldapaddent(1M)`, `svcadm(1M)`, `svccfg(1M)`, `ad(5)`, `attributes(5)`, `smf(5)`

Notes

The `idmapd` service is managed by the service management facility, `smf(5)`. The service identifier for the `idmapd` service is `svc:/system/idmap`.

Use the `svcadm` command to perform administrative actions on this service, such as enabling, disabling, or restarting the service. These actions require the `solaris.smf.manage.idmap` authorization. Use the `svcs` command to query the service's status.

Windows user names are case-insensitive, while UNIX user names are case-sensitive. The case of Windows names as they appear in `idmap` name-rules and `idmap show` command lines is irrelevant.

Because common practice in UNIX environments is to use all-lowercase user names, wildcard name-rules map Windows names to UNIX user/group names as follows: first, the canonical Windows name (that is, in the case as it appears in the directory) is used as a UNIX user or group name. If there is no such UNIX entity, then the Windows name's case is folded to lowercase and the result is used as the UNIX user or group name.

As a result of this differing treatment of case, user names that appear to be alike might not be recognized as matches. You must create rules to handle such pairings of strings that differ only in case. For example, to map the Windows user `sam@example` to the Solaris user `Sam`, you must create the following rules:

```
# idmap add "winuser:*@example" "unixuser:*"
# idmap add winuser:sam@example unixuser:Sam
```

For guidance on modifying an Active Directory schema, consult the Microsoft document, *Step-by-Step Guide to Using Active Directory Schema and Display Specifiers*, which you can find at their `technet` web site, `http://technet.microsoft.com/`.

Name idmapd – Native Identity Mapping service daemon

Synopsis `/usr/lib/idmapd`

Description The `idmapd` daemon maps Windows Security Identifiers (SIDs) to POSIX Identifiers (UIDs/GIDs) and conversely.

The `idmap(1M)` utility provides a front end to the `idmapd` daemon.

Files `/var/idmap/idmap.db`
Database in which to store local name-based ID mapping rules. The contents of the database are private. The database should not be accessed or modified directly.

`/var/run/idmap/idmap.db`
Database in which to cache ID mappings that are generated by ephemeral ID mapping and by name-based mapping. The contents of the database are private. The database should not be accessed or modified directly.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os
Interface Stability	See below.

Interface stability for these components is as follows:

`svc:/system/idmap`
Committed

`/var/idmap/idmap.db`
Project Private

`/var/run/idmap/idmap.db`
Project Private

See Also `svcs(1)`, `idmap(1M)`, `svcadm(1M)`, `svccfg(1M)`, `defaultdomain(4)`, `attributes(5)`, `smf(5)`

Notes The `idmapd` service is managed by the service management facility (SMF). The service identifier for the `idmapd` service is `svc:/system/idmap`.

Use the `svcadm` command to perform administrative actions on this service, such as enabling, disabling, or restarting the service. These actions require the `solaris.smf.manage.idmap` authorization. Use the `svcs` command to query the service's status.

The functionality of this daemon might change in a future release of the Solaris operating system.

Name idsconfig – prepare a Directory Server Enterprise Edition (DSEE) to be populated with data and serve LDAP clients

Synopsis

```
/usr/lib/ldap/idsconfig [-v] [-i input_configfile]
     [-o output_configfile]
```

Description Use the `idsconfig` tool to set up a Directory Server Enterprise Edition (DSEE). You can specify the input configuration file with the `-i` option on the command line. Alternatively, the tool will prompt the user for configuration information. The input configuration file is created by `idsconfig` with the `-o` option on a previous run.

The first time a server is set up, the user is prompted for all the required information. Future installations on that machine can use the configuration file previously generated by `idsconfig` using the `-o` option.

The output configuration file contains the directory administrator's password in clear text. Thus, if you are creating an output configuration file, take appropriate security precautions.

You should back up the directory server's configuration and data prior to running this command.

Options The following options are supported:

`-i` *input_configfile*
: Specify the file name for `idsconfig` to use as a configuration file. This file will be read by `idsconfig`, and the values in the file will be used to configure the server. Do not manually edit *input_configfile*. The *input_configfile* is only partially validated, as `idsconfig` assumes that the file was created by a previous invocation of the command.

`-o` *output_configfile*
: Create a configuration file.

`-v`
: Verbose output.

Operands The following operands are supported:

input_configfile
: Name of configuration file for `idsconfig` to use.

output_configfile
: Configuration file created by `idsconfig`.

Examples **EXAMPLE 1** Prompting the User for Input

In the following example, the user is prompted for information to set up DSEE.

```
example# idsconfig
```

EXAMPLE 2 Creating an Output Configuration File

In the following example, the user is prompted for information to set up DSEE, and an output configuration file, `config.1`, is created when completed.

```
example# idsconfig -o config.1
```

EXAMPLE 3 Setting up DSEE Using the Specified Configuration File

In the following example, DSEE is set up by using the values specified in the configuration file, `config.1`. The verbose mode is specified, so detailed information will print to the screen.

```
example# idsconfig -v -i config.1
```

Exit Status

The following exit values are returned:

`0`
: Successful completion.

`>0`
: An error occurred.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/network/nis
Interface Stability	Committed

See Also

`ldapadd(1)`, `ldapdelete(1)`, `ldaplist(1)`, `ldapmodify(1)`, `ldapmodrdn(1)`, `ldapsearch(1)`, `ldap_cachemgr(1M)`, `ldapaddent(1M)`, `ldapclient(1M)`, `resolv.conf(4)`, `attributes(5)`,`ldap(5)`

Name ifconfig – configure network interface parameters

Synopsis

```
ifconfig interface [address_family] [address [/prefix_length]
 [dest_address]] [addif address [/prefix_length]]
 [removeif address [/prefix_length]] [arp | -arp]
 [auth_algs authentication algorithm] [encr_algs encryption algorithm]
 [encr_auth_algs authentication algorithm] [auto-revarp]
 [broadcast address] [deprecated | -deprecated]
 [preferred | -preferred] [destination dest_address]
 [ether [address]] [failover | -failover] [group
 [name | ""]] [index if_index] [ipmp] [metric n] [modlist]
 [modinsert mod_name@pos] [modremove mod_name@pos]
 [mtu n] [netmask mask] [plumb] [unplumb] [private
 | -private] [nud | -nud] [set [address] [/netmask]]
 [standby | -standby] [subnet subnet_address] [tdst
 tunnel_dest_address] [token address/prefix_length]
 [tsrc tunnel_src_address] [trailers | -trailers]
 [up] [down] [usesrc [name | none]] [xmit | -xmit]
 [encaplimit n | -encaplimit] [thoplimit n] [router
 | -router] [zone zonename | -zone | -all-zones]

ifconfig [address_family] interface {auto-dhcp | dhcp} [primary]
 [wait seconds] drop | extend | inform | ping
 | release | start | status
```

Description The command `ifconfig` is used to assign an address to a network interface and to configure network interface parameters. Network interfaces configured by the `ifconfig` command do not survive a reboot. The `ipadm(1M)` command must be used to configure network interfaces persistently. If no option is specified, `ifconfig` displays the current configuration for a network interface. If an address family is specified, `ifconfig` reports only the details specific to that address family. Only privileged users may modify the configuration of a network interface. Options appearing within braces (`{ }`) indicate that one of the options must be specified.

DHCP Configuration The forms of `ifconfig` that use the `auto-dhcp` or `dhcp` arguments are used to control the Dynamic Host Configuration Protocol ("DHCP") configuration of the interface. In this mode, `ifconfig` is used to control operation of `dhcpagent(1M)`, the DHCP client daemon. Once an interface is placed under DHCP control by using the `start` operand, `ifconfig` should not, in normal operation, be used to modify the address or characteristics of the interface. If the address of an interface under DHCP is changed, `dhcpagent` will remove the interface from its control.

Options The following options are supported:

`addif` *address*
Create the next unused logical interface on the specified physical interface.

`all-zones`
Make the interface available to every shared-IP zone on the system. The appropriate zone to which to deliver data is determined using the `tnzonecfg` database. This option is available only if the system is configured with the Solaris Trusted Extensions feature.

The `tnzonecfg` database is described in the `tnzonecfg(4)` man page, which is part of the *Solaris Trusted Extensions Reference Manual*.

`anycast`
Marks the logical interface as an anycast address by setting the `ANYCAST` flag. See "INTERFACE FLAGS," below, for more information on anycast.

`-anycast`
Marks the logical interface as not an anycast address by clearing the `ANYCAST` flag.

`arp`
Enable the use of the Address Resolution Protocol ("ARP") in mapping between network level addresses and link level addresses (default). This is currently implemented for mapping between IPv4 addresses and MAC addresses.

`-arp`
Disable the use of the ARP on a physical interface. ARP cannot be disabled on an IPMP IP interface.

`auth_algs` *authentication algorithm*
For a tunnel, enable IPsec AH with the authentication algorithm specified. The algorithm can be either a number or an algorithm name, including *any* to express no preference in algorithm. All IPsec tunnel properties must be specified on the same command line. To disable tunnel security, specify an `auth_alg` of `none`.

It is now preferable to use the `ipsecconf(1M)` command when configuring a tunnel's security properties. If `ipsecconf` was used to set a tunnel's security properties, this keyword will not affect the tunnel.

`auto-dhcp`
Use DHCP to automatically acquire an address for this interface. This option has a completely equivalent alias called `dhcp`.

For IPv6, the interface specified must be the zeroth logical interface (the physical interface name), which has the link-local address.

primary
Defines the interface as the `primary`. The interface is defined as the preferred one for the delivery of client-wide configuration data. Only one interface can be the primary at any given time. If another interface is subsequently selected as the primary, it replaces the previous one. Nominating an interface as the primary one will not have much significance once the client work station has booted, as many applications will already have started and been configured with data read from the previous primary interface.

wait *seconds*
The `ifconfig` command will wait until the operation either completes or for the interval specified, whichever is the sooner. If no wait interval is given, and the operation is one that cannot complete immediately, `ifconfig` will wait 30 seconds for the requested operation to complete. The symbolic value `forever` may be used as well, with obvious meaning.

drop
Remove the specified interface from DHCP control without notifying the DHCP server, and record the current lease for later use. Additionally, for IPv4, set the IP address to zero. For IPv6, unplumb all logical interfaces plumbed by `dhcpagent`.

extend
Attempt to extend the lease on the interface's IP address. This is not required, as the agent will automatically extend the lease well before it expires.

inform
Obtain network configuration parameters from DHCP without obtaining a lease on IP addresses. This is useful in situations where an IP address is obtained through mechanisms other than DHCP.

ping
Check whether the interface given is under DHCP control, which means that the interface is managed by the DHCP agent and is working properly. An exit status of `0` means success.

release
Relinquish the IP addresses on the interface by notifying the server and discard the current lease. For IPv4, set the IP address to zero. For IPv6, all logical interfaces plumbed by `dhcpagent` are unplumbed.

start
Start DHCP on the interface.

status
Display the DHCP configuration status of the interface.

`auto-revarp`
Use the Reverse Address Resolution Protocol (RARP) to automatically acquire an address for this interface. This will fail if the interface does not support RARP; for example, IPoIB (IP over InfiniBand), and on IPv6 interfaces.

`broadcast` *address*
For IPv4 only. Specify the address to use to represent broadcasts to the network. The default broadcast address is the address with a host part of all 1's. A “+” (plus sign) given for the broadcast value causes the broadcast address to be reset to a default appropriate for the (possibly new) address and netmask. The arguments of `ifconfig` are interpreted left to right. Therefore

```
example% ifconfig -a netmask + broadcast +
```

and

```
example% ifconfig -a broadcast + netmask +
```

may result in different values being assigned for the broadcast addresses of the interfaces.

`deprecated`
Marks the logical interface as deprecated. An address associated with a deprecated interface will not be used as source address for outbound packets unless either there are no other addresses available on the interface or the application has bound to this address explicitly. The status display shows `DEPRECATED` as part of flags. See INTERFACE FLAGS for information on the flags supported by `ifconfig`.

`-deprecated`
Marks a logical interface as not deprecated. An address associated with such an interface could be used as a source address for outbound packets.

`preferred`
Marks the logical interface as preferred. This option is only valid for IPv6 addresses. Addresses assigned to preferred logical interfaces are preferred as source addresses over all other addresses configured on the system, unless the address is of an inappropriate scope relative to the destination address. Preferred addresses are used as source addresses regardless of which physical interface they are assigned to. For example, you can configure a preferred source address on the loopback interface and advertise reachability of this address by using a routing protocol.

`-preferred`
Marks the logical interface as not preferred.

`destination` *dest_address*
Set the destination address for a point-to point interface.

`dhcp`
This option is an alias for option `auto-dhcp`

`down`
Mark a logical interface as "down". (That is, turn off the `IFF_UP` bit.) When a logical interface is marked "down," the system does not attempt to use the address assigned to that interface as a source address for outbound packets and will not recognize inbound packets destined to that address as being addressed to this host. Additionally, when all logical interfaces on a given physical interface are "down," the physical interface itself is disabled.

When a logical interface is down, all routes that specify that interface as the output (using the `-ifp` option in the `route(1M)` command or `RTA_IFP` in a `route(7P)` socket) are removed from the forwarding table. Routes marked with `RTF_STATIC` are returned to the table if the interface is brought back up, while routes not marked with `RTF_STATIC` are simply deleted.

When all logical interfaces that could possibly be used to reach a particular gateway address are brought down (specified without the interface option as in the previous paragraph), the

affected gateway routes are treated as though they had the RTF_BLACKHOLE flag set. All matching packets are discarded because the gateway is unreachable.

encaplimit *n*
Set the tunnel encapsulation limit for the interface to n. This option applies to IPv4–in-IPv6 and IPv6–in-IPv6 tunnels only, and it simply modifies the encaplimit link property of the underlying IPv6 tunnel link (see dladm(1M)). The tunnel encapsulation limit controls how many more tunnels a packet can enter before it leaves any tunnel, that is, the tunnel nesting level.

This option is obsolete, superseded by the dladm(1M) encaplimit link property.

-encaplimit
Disable generation of the tunnel encapsulation limit. This option applies only to IPv4–in-IPv6 and IPv6–in-IPv6 tunnels. This simply sets the encaplimit link property of the underlying IPv6 tunnel link to 0 (see dladm(1M) encaplimit).

This option is obsolete, superseded by the dladm(1M) encaplimit link property.

encr_auth_algs *authentication algorithm*
For a tunnel, enable IPsec ESP with the authentication algorithm specified. It can be either a number or an algorithm name, including any or none, to indicate no algorithm preference. If an ESP encryption algorithm is specified but the authentication algorithm is not, the default value for the ESP authentication algorithm will be any.

It is now preferable to use the ipsecconf(1M) command when configuring a tunnel's security properties. If ipsecconf was used to set a tunnel's security properties, this keyword will not affect the tunnel.

encr_algs *encryption algorithm*
For a tunnel, enable IPsec ESP with the encryption algorithm specified. It can be either a number or an algorithm name. Note that all IPsec tunnel properties must be specified on the same command line. To disable tunnel security, specify the value of encr_alg as none. If an ESP authentication algorithm is specified, but the encryption algorithm is not, the default value for the ESP encryption will be null.

It is now preferable to use the ipsecconf(1M) command when configuring a tunnel's security properties. If ipsecconf was used to set a tunnel's security properties, this keyword will not affect the tunnel.

ether [*address*]
If no address is given and the user is root or has sufficient privileges to open the underlying datalink, then display the current Ethernet address information.

Otherwise, if the user is root or has sufficient privileges, set the Ethernet address of the interfaces to *address*. The address is an Ethernet address represented as *x:x:x:x:x:x* where *x* is a hexadecimal number between 0 and FF. Similarly, for the IPoIB (IP over InfiniBand) interfaces, the address will be 20 bytes of colon-separated hex numbers between 0 and FF.

Some, though not all, Ethernet interface cards have their own addresses. To use cards that do not have their own addresses, refer to section 3.2.3(4) of the IEEE 802.3 specification for a definition of the locally administered address space. Note that all IP interfaces in an IPMP group must have unique hardware addresses; see `in.mpathd(1M)`.

`-failover`
Set `NOFAILOVER` on the logical interface. This makes the associated address available for use by `in.mpathd` to perform probe-based failure detection for the associated physical IP interface. As a side effect, `DEPRECATED` will also be set on the logical interface. This operation is not permitted on an IPMP IP interface.

`failover`
Clear `NOFAILOVER` on the logical interface. This is the default. These logical interfaces are subject to migration when brought up (see `IP MULTIPATHING GROUPS`).

`group` [*name* |`""`]
When applied to a physical interface, it places the interface into the named group. If the group does not exist, it will be created, along with one or more IPMP IP interfaces (for IPv4, IPv6, or both). Any `UP` addresses that are not also marked `NOFAILOVER` are subject to migration to the IPMP IP interface (see `IP MULTIPATHING GROUPS`). Specifying a group name of `""` removes the physical IP interface from the group.

When applied to a physical IPMP IP interface, it renames the IPMP group to have the new name. If the name already exists, or a name of `""` is specified, it fails. Renaming IPMP groups is discouraged. Instead, the IPMP IP interface should be given a meaningful name when it is created by means of the `ipmp` subcommand, which the system will also use as the IPMP group name.

`index` *n*
Change the interface index for the interface. The value of *n* must be an interface index (*if_index*) that is not used on another interface. *if_index* will be a non-zero positive number that uniquely identifies the network interface on the system.

`ipmp`
Create an IPMP IP interface with the specified name. An interface must be separately created for use by IPv4 and IPv6. The *address_family* parameter controls whether the command applies to IPv4 or IPv6 (IPv4 if unspecified). All IPMP IP interfaces have the `IPMP` flag set.

`metric` *n*
Set the routing metric of the interface to *n*; if no value is specified, the default is `0`. The routing metric is used by the routing protocol. Higher metrics have the effect of making a route less favorable. Metrics are counted as addition hops to the destination network or host.

`modinsert` *mod_name@pos*
Insert a module with name *mod_name* to the stream of the device at position *pos*. The position is relative to the stream head. Position `0` means directly under stream head.

Based upon the example in the `modlist` option, use the following command to insert a module with name `ipqos` under the `ip` module and above the firewall module:

```
example% ifconfig eri0 modinsert ipqos@2
```

A subsequent listing of all the modules in the stream of the device follows:

```
example% ifconfig eri0 modlist
0 arp
1 ip
2 ipqos
3 firewall
4 eri
```

`modlist`
List all the modules in the stream of the device.

The following example lists all the modules in the stream of the device:

```
example% ifconfig eri0 modlist
0 arp
1 ip
2 firewall
4 eri
```

`modremove` *mod_name@pos*
Remove a module with name *mod_name* from the stream of the device at position *pos*. The position is relative to the stream head.

Based upon the example in the `modinsert` option, use the following command to remove the firewall module from the stream after inserting the `ipqos` module:

```
example% ifconfig eri0 modremove firewall@3
```

A subsequent listing of all the modules in the stream of the device follows:

```
example% ifconfig eri0 modlist
0 arp
1 ip
2 ipqos
3 eri
```

Note that the core IP stack modules, for example, `ip` and `tun` modules, cannot be removed.

`mtu` *n*
Set the maximum transmission unit of the interface to *n*. For many types of networks, the `mtu` has an upper limit, for example, `1500` for Ethernet. This option sets the `FIXEDMTU` flag on the affected interface.

`netmask` *mask*
For IPv4 only. Specify how much of the address to reserve for subdividing networks into subnetworks. The mask includes the network part of the local address and the subnet part, which is taken from the host field of the address. The mask contains 1's for the bit positions

in the 32-bit address which are to be used for the network and subnet parts, and 0's for the host part. The mask should contain at least the standard network portion, and the subnet field should be contiguous with the network portion. The mask can be specified in one of four ways:

1. with a single hexadecimal number with a leading 0x,
2. with a dot-notation address,
3. with a "+" (plus sign) address, or
4. with a pseudo host name/pseudo network name found in the network database `networks(4)`.

If a "+" (plus sign) is given for the netmask value, the mask is looked up in the `netmasks(4)` database. This lookup finds the longest matching netmask in the database by starting with the interface's IPv4 address as the key and iteratively masking off more and more low order bits of the address. This iterative lookup ensures that the `netmasks(4)` database can be used to specify the netmasks when variable length subnetmasks are used within a network number.

If a pseudo host name/pseudo network name is supplied as the netmask value, netmask data may be located in the `hosts` or `networks` database. Names are looked up by first using `gethostbyname(3NSL)`. If not found there, the names are looked up in `getnetbyname(3SOCKET)`. These interfaces may in turn use `nsswitch.conf(4)` to determine what data store(s) to use to fetch the actual value.

For both `inet` and `inet6`, the same information conveyed by *mask* can be specified as a *prefix_length* attached to the *address* parameter.

`nud`
Enables the neighbor unreachability detection mechanism on a point-to-point physical interface.

`-nud`
Disables the neighbor unreachability detection mechanism on a point-to-point physical interface.

`plumb`
For a physical IP interface, open the datalink associated with the physical interface name and set up the plumbing needed for IP to use the datalink. When used with a logical interface name, this command is used to create a specific named logical interface on an existing physical IP interface.

An interface must be separately plumbed for IPv4 and IPv6 according to the *address_family* parameter (IPv4 if unspecified). Before an interface has been plumbed, it will not be shown by `ifconfig -a`.

Note that IPMP IP interfaces are not tied to a specific datalink and are instead created with the `ipmp` subcommand.

`private`
Tells the `in.routed` routing daemon that a specified logical interface should not be advertised.

`-private`
Specify unadvertised interfaces.

`removeif` *address*
Remove the logical interface on the physical interface specified that matches the *address* specified.

`router`
Enable IP forwarding on the interface. When enabled, the interface is marked `ROUTER`, and IP packets can be forwarded to and from the interface. Enabling `ROUTER` on any IP interface in an IPMP group enables it on all IP interfaces in that IPMP group.

`-router`
Disable IP forwarding on the interface. IP packets are not forwarded to and from the interface. Disabling `ROUTER` on any IP interface in an IPMP group disables it on all IP interfaces in that IPMP group.

`set`
Set the *address*, *prefix_length* or both, for a logical interface.

`standby`
Mark the physical IP interface as a `STANDBY` interface. If an interface is marked `STANDBY` and is part of an IPMP group, the interface will not be used for data traffic unless another interface in the IPMP group becomes unusable. When a `STANDBY` interface is functional but not being used for data traffic, it will also be marked `INACTIVE`. This operation is not permitted on an IPMP IP interface.

`-standby`
Clear `STANDBY` on the interface. This is the default.

`subnet`
Set the subnet *address* for an interface.

`tdst` *tunnel_dest_address*
Set the destination address of a tunnel. The address should not be the same as the `dest_address` of the tunnel, because no packets leave the system over such a tunnel.

This option is obsolete, superseded by the `dladm(1M)` `create-iptun` and `modify-iptun` subcommands.

`thoplimit` *n*
Set the hop limit for a tunnel interface. The hop limit value is used as the `TTL` in the IPv4 header for the IPv6–in-IPv4 and IPv4–in-IPv4 tunnels. For IPv6–in-IPv6 and IPv4–in-IPv6 tunnels, the hop limit value is used as the hop limit in the IPv6 header. This option simply modifies the `hoplimit` link property of the underlying IP tunnel link (see `dladm(1M)`).

This option is obsolete, superseded by the dladm(1M) `hoplimit` link property.

`token` *address*/*prefix_length*
Set the IPv6 token of an interface to be used for address autoconfiguration.

```
example% ifconfig eri0 inet6 token ::1/64
```

`trailers`
This flag previously caused a nonstandard encapsulation of IPv4 packets on certain link levels. Drivers supplied with this release no longer use this flag. It is provided for compatibility, but is ignored.

`-trailers`
Disable the use of a "trailer" link level encapsulation.

`tsrc` *tunnel_src_address*
Set the source address of a tunnel. This is the source address on an outer encapsulating IP header. It must be an address of another interface already configured using `ifconfig`.

This option is obsolete, superseded by the dladm(1M) `create-iptun` and `modify-iptun` subcommands.

`unplumb`
For a physical or IPMP interface, remove all associated logical IP interfaces and tear down any plumbing needed for IP to use the interface. For an IPMP IP interface, this command will fail if the group is not empty. For a logical interface, the logical interface is removed.

An interface must be separately unplumbed for IPv4 and IPv6 according to the *address_family* parameter (IPv4 if unspecified). Upon success, the interface name will no longer appear in the output of `ifconfig -a`.

`up`
Mark a logical interface `UP`. As a result, the IP module will accept packets destined to the associated address (unless the address is zero), along with any associated multicast and broadcast IP addresses. Similarly, the IP module will allow packets to be sent with the associated address as a source address. At least one logical interface must be `UP` for the associated physical interface to send or receive packets

`usesrc` [*name* | `none`]
Specify a physical interface to be used for source address selection. If the keyword `none` is used, then any previous selection is cleared.

When an application does not choose a non-zero source address using `bind`(3SOCKET), the system will select an appropriate source address based on the outbound interface and the address selection rules (see `ipaddrsel`(1M)).

When `usesrc` is specified and the specified interface is selected in the forwarding table for output, the system looks first to the specified physical interface and its associated logical interfaces when selecting a source address. If no usable address is listed in the forwarding table, the ordinary selection rules apply. For example, if you enter:

```
# ifconfig eri0 usesrc vni0
```

...and `vni0` has address 10.0.0.1 assigned to it, the system will prefer 10.0.0.1 as the source address for any packets originated by local connections that are sent through `eri0`. Further examples are provided in the `EXAMPLES` section.

While you can specify any physical interface (or even loopback), be aware that you can also specify the virtual IP interface (see `vni(7d)`). The virtual IP interface is not associated with any physical hardware and is thus immune to hardware failures. You can specify any number of physical interfaces to use the source address hosted on a single virtual interface. This simplifies the configuration of routing-based multipathing. If one of the physical interfaces were to fail, communication would continue through one of the remaining, functioning physical interfaces. This scenario assumes that the reachability of the address hosted on the virtual interface is advertised in some manner, for example, through a routing protocol.

Because the `ifconfig preferred` option is applied to all interfaces, it is coarser-grained than the `usesrc` option. It will be overridden by `usesrc` and `setsrc` (route subcommand), in that order.

`xmit`
Enable a logical interface to transmit packets. This is the default behavior when the logical interface is up.

`-xmit`
Disable transmission of packets on an interface. The interface will continue to receive packets.

`zone` *zonename*
Place the logical interface in zone *zonename*. The named zone must be active in the kernel in the ready or running state. The interface is unplumbed when the zone is halted or rebooted. The zone must be configure to be an shared-IP zone. `zonecfg(1M)` is used to assign network interface names to exclusive-IP zones.

`-zone`
Place IP interface in the global zone. This is the default.

Operands The *interface* operand, as well as address parameters that affect it, are described below.

interface
A string of one of the following forms:

- *name physical-unit*, for example, `eri0` or `ce1`
- *name physical-unit*`:`*logical-unit*, for example, `eri0:1`
- `ip.tun`*N*, `ip6.tun`*N*, or `ip6to4.tun`*N* for implicit IP tunnel links

If the interface name starts with a dash (-), it is interpreted as a set of options which specify a set of interfaces. In such a case, `-a` must be part of the options and any of the additional options below can be added in any order. If one of these interface names is given, the commands following it are applied to all of the interfaces that match.

`-a`
Apply the command to all interfaces of the specified address family. If no address family is supplied, either on the command line or by means of `/etc/default/inet_type`, then all address families will be selected.

`-d`
Apply the commands to all "down" interfaces in the system.

`-D`
Apply the commands to all interfaces not under DHCP (Dynamic Host Configuration Protocol) control.

`-u`
Apply the commands to all "up" interfaces in the system.

`-Z`
Apply the commands to all interfaces in the user's zone.

`-4`
Apply the commands to all IPv4 interfaces.

`-6`
Apply the commands to all IPv6 interfaces.

address_family
The address family is specified by the *address_family* parameter. The `ifconfig` command currently supports the following families: `inet` and `inet6`. If no address family is specified, the default is `inet`.

`ifconfig` honors the `DEFAULT_IP` setting in the `/etc/default/inet_type` file when it displays interface information . If `DEFAULT_IP` is set to `IP_VERSION4`, then `ifconfig` will omit information that relates to IPv6 interfaces. However, when you explicitly specify an address family (`inet` or `inet6`) on the `ifconfig` command line, the command line overrides the `DEFAULT_IP` settings.

address
For the IPv4 family (`inet`), the *address* is either a host name present in the host name data base (see `hosts(4)`) or in the Network Information Service (NIS) map `hosts`, or an IPv4 address expressed in the Internet standard "dot notation".

For the IPv6 family (`inet6`), the *address* is either a host name present in the host name data base (see `hosts(4)`) or in the Network Information Service (NIS) map `ipnode`, or an IPv6 address expressed in the Internet standard colon-separated hexadecimal format represented as *x:x:x:x:x:x:x:x* where *x* is a hexadecimal number between `0` and `FFFF`.

prefix_length
For the IPv4 and IPv6 families (`inet` and `inet6`), the *prefix_length* is a number between 0 and the number of bits in the address. For `inet`, the number of bits in the address is 32; for `inet6`, the number of bits in the address is 128. The *prefix_length* denotes the number of leading set bits in the netmask.

dest_address
If the *dest_address* parameter is supplied in addition to the *address* parameter, it specifies the address of the correspondent on the other end of a point-to-point link.

tunnel_dest_address
An address that is or will be reachable through an interface other than the tunnel being configured. This tells the tunnel where to send the tunneled packets. This address must not be the same as the interface destination address being configured.

tunnel_src_address
An address that is attached to an already configured interface that has been configured "up" with `ifconfig`.

Interface Flags The `ifconfig` command supports the following interface flags. The term "address" in this context refers to a logical interface, for example, `eri0:0`, while "interface" refers to the physical interface, for example, `eri0`.

`ADDRCONF`
The address is from stateless `addrconf`. The stateless mechanism allows a host to generate its own address using a combination of information advertised by routers and locally available information. Routers advertise prefixes that identify the subnet associated with the link, while the host generates an "interface identifier" that uniquely identifies an interface in a subnet. In the absence of information from routers, a host can generate link-local addresses. This flag is specific to IPv6.

`ANYCAST`
Indicates an `anycast` address. An `anycast` address identifies the nearest member of a group of systems that provides a particular type of service. An `anycast` address is assigned to a group of systems. Packets are delivered to the nearest group member identified by the `anycast` address instead of being delivered to all members of the group.

`BROADCAST`
This `broadcast` address is valid. This flag and `POINTTOPOINT` are mutually exclusive

`CoS`
This interface supports some form of Class of Service (CoS) marking. An example is the 802.1D user priority marking supported on VLAN interfaces. For IPMP IP interfaces, this will only be set if all interfaces in the group have CoS set.

Note that this flag is only set on interfaces over VLAN links and over Ethernet links that have their `dladm(1M)` `tagmode` link property set to `normal`.

`DEPRECATED`
This address is deprecated. This address will not be used as a source address for outbound packets unless there are no other addresses on this interface or an application has explicitly bound to this address. An IPv6 deprecated address is part of the standard mechanism for renumbering in IPv6 and will eventually be deleted when not used. For both IPv4 and IPv6, `DEPRECATED` is also set on all `NOFAILOVER` addresses, though this may change in a future release.

DHCPRUNNING
The logical interface's address is managed by dhcpagent(1M). For IPv6, this will also be set on the zeroth logical interface if DHCPv6 has been started on the interface; see in.ndpd(1M).

DUPLICATE
The logical interface has been disabled because the IP address configured on the interface is a duplicate. Some other node on the network is using this address. If the address was configured by DHCP or is temporary, the system will choose another automatically, if possible. Otherwise, the system will attempt to recover this address periodically and the interface will recover when the conflict has been removed from the network. Changing the address or netmask, or setting the logical interface to up will restart duplicate detection. Setting the interface to down terminates recovery and removes the DUPLICATE flag.

FAILED
The in.mpathd daemon has determined that the interface has failed. FAILED interfaces will not be used to send or receive IP data traffic. If this is set on a physical IP interface in an IPMP group, IP data traffic will continue to flow over other usable IP interfaces in the IPMP group. If this is set on an IPMP IP interface, the entire group has failed and no data traffic can be sent or received over any interfaces in that group.

FIXEDMTU
The MTU has been set using the -mtu option. This flag is read-only. Interfaces that have this flag set have a fixed MTU value that is unaffected by dynamic MTU changes that can occur when drivers notify IP of link MTU changes.

INACTIVE
The physical interface is functioning but is not used to send or receive data traffic according to administrative policy. This flag is initially set by the standby subcommand and is subsequently controlled by in.mpathd. It also set when FAILBACK=no mode is enabled (see in.mpathd(1M)) to indicate that the IP interface has repaired but is not being used.

IPMP
Indicates that this is an IPMP IP interface.

LOOPBACK
Indicates that this is the loopback interface.

MULTI_BCAST
Indicates that the broadcast address is used for multicast on this interface.

MULTICAST
The interface supports multicast. IP assumes that any interface that supports hardware broadcast, or that is a point-to-point link, will support multicast.

NOARP
There is no address resolution protocol (ARP) for this interface that corresponds to all interfaces for a device without a broadcast address. This flag is specific to IPv4.

NOFAILOVER
The address associated with this logical interface is available to `in.mpathd` for probe-based failure detection of the associated physical IP interface.

NOLOCAL
The interface has no address , just an on-link subnet.

NONUD
`NUD` is disabled on this interface. `NUD` (neighbor unreachability detection) is used by a node to track the reachability state of its neighbors, to which the node actively sends packets, and to perform any recovery if a neighbor is detected to be unreachable. This flag is specific to IPv6.

NORTEXCH
The interface does not exchange routing information. For RIP-2, routing packets are not sent over this interface. Additionally, messages that appear to come over this interface receive no response. The subnet or address of this interface is not included in advertisements over other interfaces to other routers.

NOXMIT
Indicates that the address does not transmit packets. RIP-2 also does not advertise this address.

OFFLINE
The interface is offline and thus cannot send or receive IP data traffic. This is only set on IP interfaces in an IPMP group. See `if_mpadm(1M)` and `cfgadm(1M)`.

POINTOPOINT
Indicates that the address is a point-to-point link. This flag and `BROADCAST` are mutually exclusive

PREFERRED
This address is a preferred IPv6 source address. This address will be used as a source address for IPv6 communication with all IPv6 destinations, unless another address on the system is of more appropriate scope. The `DEPRECATED` flag takes precedence over the `PREFERRED` flag.

PRIVATE
Indicates that this address is not advertised. For RIP-2, this interface is used to send advertisements. However, neither the subnet nor this address are included in advertisements to other routers.

PROMISC
A read-only flag indicating that an interface is in promiscuous mode. All addresses associated with an interface in promiscuous mode will display (in response to `ifconfig -a`, for example) the `PROMISC` flag.

ROUTER
Indicates that IP packets can be forwarded to and from the interface.

RUNNING
: Indicates that the required resources for an interface are allocated. For some interfaces this also indicates that the link is up. For IPMP IP interfaces, RUNNING is set as long as one IP interface in the group is active.

STANDBY
: Indicates that this physical interface will not be used for data traffic unless another interface in the IPMP group becomes unusable. The INACTIVE and FAILED flags indicate whether it is actively being used.

TEMPORARY
: Indicates that this is a temporary IPv6 address as defined in RFC 3041.

UNNUMBERED
: This flag is set when the local IP address on the link matches the local address of some other link in the system

UP
: Indicates that the logical interface (and the associated physical interface) is up. The IP module will accept packets destined to UP addresses (unless the address is zero), along with any associated multicast and broadcast IP addresses. Similarly, the IP module will allow packets to be sent with an UP address as a source address.

VIRTUAL
: Indicates that the physical interface has no underlying hardware. It is not possible to transmit or receive packets through a virtual interface. These interfaces are useful for configuring local addresses that can be used on multiple interfaces. (See also the usesrc option.)

L3PROTECT
: Indicates that Layer-3 protection has been enforced on the physical interface using the allowed-ips link property in dladm(1M).

PROBER
: Indicates that the FAILED underlying interface in an IPMP group is probing to discover if it has been repaired. The PROBER flag and its semantics are internal to the Solaris IPMP implementation and subject to change.

Logical Interfaces

Solaris TCP/IP allows multiple logical interfaces to be associated with a physical network interface. This allows a single machine to be assigned multiple IP addresses, even though it may have only one network interface. Physical network interfaces have names of the form *driver-name physical-unit-number*, while logical interfaces have names of the form *driver-name physical-unit-number*:*logical-unit-number*. A physical interface is configured into the system using the plumb command. For example:

```
example% ifconfig eri0 plumb
```

Once a physical interface has been "plumbed", logical interfaces associated with the physical interface can be configured by separate -plumb or -addif options to the ifconfig command.

```
example% ifconfig eri0:1 plumb
```

allocates a specific logical interface associated with the physical interface `eri0`. The command

```
example% ifconfig eri0 addif 192.168.200.1/24 up
```

allocates the next available logical unit number on the `eri0` physical interface and assigns an *address* and *prefix_length*.

A logical interface can be configured with parameters (*address*,*prefix_length*, and so on) different from the physical interface with which it is associated. Logical interfaces that are associated with the same physical interface can be given different parameters as well. Each logical interface must be associated with an existing and "up" physical interface. So, for example, the logical interface `eri0:1` can only be configured after the physical interface `eri0` has been plumbed.

To delete a logical interface, use the `unplumb` or `removeif` options. For example,

```
example% ifconfig eri0:1 down unplumb
```

will delete the logical interface `eri0:1`.

Ip Multipathing Groups

Physical interfaces that share the same link–layer broadcast domain *must* be collected into a single IP Multipathing (IPMP) group using the `group` subcommand. Each IPMP group has an associated IPMP IP interface, which can either be explicitly created (the preferred method) by using the `ipmp` subcommand or implicitly created by `ifconfig` in response to placing an IP interface into a new IPMP group. Implicitly-created IPMP interfaces will be named `ipmp`*N* where *N* is the lowest integer that does not conflict with an existing IP interface name or IPMP group name.

Each IPMP IP interface is created with a matching IPMP group name, though it can be changed using the `group` subcommand. Each IPMP IP interface hosts a set of highly-available IP addresses. These addresses will remain reachable so long as at least one interface in the group is active, where "active" is defined as having at least one `UP` address and having `INACTIVE`, `FAILED`, and `OFFLINE` clear. IP addresses hosted on the IPMP IP interface may either be configured statically or configured through DHCP by means of the `dhcp` subcommand.

Interfaces assigned to the same IPMP group are treated as equivalent and monitored for failure by `in.mpathd`. Provided that active interfaces in the group remain, IP interface failures (and any subsequent repairs) are handled transparently to sockets-based applications. IPMP is also integrated with the Dynamic Reconfiguration framework (see `cfgadm(1M)`), which enables network adapters to be replaced in a way that is invisible to sockets-based applications.

The IP module automatically load-spreads all outbound traffic across all active interfaces in an IPMP group. Similarly, all `UP` addresses hosted on the IPMP IP interface will be distributed

across the active interfaces to promote inbound load-spreading. The ipmpstat(1M) utility allows many aspects of the IPMP subsystem to be observed, including the current binding of IP data addresses to IP interfaces.

When an interface is placed into an IPMP group, any UP logical interfaces are "migrated" to the IPMP IP interface for use by the group, unless:

- the logical interface is marked NOFAILOVER;
- the logical interface hosts an IPv6 link-local address;
- the logical interface hosts an IPv4 0.0.0.0 address.

Likewise, once an interface is in a group, if changes are made to a logical interface such that it is UP and not exempted by one of the conditions above, it will also migrate to the associated IPMP IP interface. Logical interfaces never migrate back, even if the physical interface that contributed the address is removed from the group.

Each interface placed into an IPMP group may be optionally configured with a "test" address that in.mpathd will use for probe-based failure detection; see in.mpathd(1M). These addresses must be marked NOFAILOVER (using the -failover subcommand) prior to being marked UP. Test addresses may also be acquired through DHCP by means of the dhcp subcommand.

For more background on IPMP, please see the IPMP–related chapters of the *Oracle Solaris Administration: Network Interfaces and Network Virtualization*.

Configuring Ipv6 Interfaces

When an IPv6 physical interface is plumbed and configured "up" with ifconfig, it is automatically assigned an IPv6 link-local address for which the last 64 bits are calculated from the MAC address of the interface.

```
example% ifconfig eri0 inet6 plumb up
```

The following example shows that the link-local address has a prefix of fe80::/10.

```
example% ifconfig eri0 inet6
ce0: flags=2000841<UP,RUNNING,MULTICAST,IPv6>
          mtu 1500 index 2
        inet6 fe80::a00:20ff:fe8e:f3ad/10
```

Link-local addresses are only used for communication on the local subnet and are not visible to other subnets.

If an advertising IPv6 router exists on the link advertising prefixes, then the newly plumbed IPv6 interface will autoconfigure logical interface(s) depending on the prefix advertisements. For example, for the prefix advertisement 2001:0db8:3c4d:0:55::/64, the autoconfigured interface will look like:

```
eri0:2: flags=2080841<UP,RUNNING,MULTICAST,ADDRCONF,IPv6>
          mtu 1500 index 2
        inet6 2001:0db8:3c4d:55:a00:20ff:fe8e:f3ad/64
```

Even if there are no prefix advertisements on the link, you can still assign global addresses manually, for example:

```
example% ifconfig eri0 inet6 addif \
2001:0db8:3c4d:55:a00:20ff:fe8e:f3ad/64 up
```

Configuring IP-over-IP Tunnel Interfaces

An IP tunnel is conceptually comprised of two parts: a virtual link between two or more IP nodes, and an IP interface above this link which allows the system to transmit and receive IP packets encapsulated by the underlying link.

The dladm(1M) command is used to configure tunnel links, and `ifconfig` is used to configure IP interfaces over those tunnel links. An IPv4-over-IPv4 tunnel is created by plumbing an IPv4 interface over an IPv4 tunnel link. An IPv6-over-IPv4 tunnel is created by plumbing an IPv6 interface over an IPv6 tunnel link, and so forth.

When IPv6 interfaces are plumbed over IP tunnel links, their IPv6 addresses are automatically set. For IPv4 and IPv6 tunnels, source and destination link-local addresses of the form `fe80::`*interface-id* are configured. For IPv4 tunnels, the *interface-id* is the IPv4 tunnel source or destination address. For IPv6 tunnels, the *interface-id* is the last 64 bits of the IPv6 tunnel source or destination address. For example, for an IPv4 tunnel between 10.1.2.3 and 10.4.5.6, the IPv6 link-local source and destination addresses of the IPv6 interface would be `fe80::a01:203` and `fe80::a04:506`. For an IPv6 tunnel between `2000::1234:abcd` and `3000::5678:abcd`, the IPv6 link-local source and destination addresses of the interface would be `fe80::1234:abcd` and `fe80::5678:abcd`. These default link-local addresses can be overridden by specifying the addresses explicitly, as with any other point-to-point interface.

For 6to4 tunnels, a 6to4 global address of the form `2002:`*tsrc*`::1/16` is configured. The *tsrc* portion is the tunnel source IPv4 address. The prefix length of the 6to4 interface is automatically set to 16, as all 6to4 packets (destinations in the `2002::/16` range) are forwarded to the 6to4 tunnel interface. For example, for a 6to4 link with a tunnel source of 75.1.2.3, the IPv6 interface would have an address of `2002:4b01:203::1/16`.

Additional IPv6 addresses can be added using the `addif` option or by plumbing additional logical interfaces.

For backward compatibility, the plumbing of tunnel IP interfaces with special names will implicitly result in the creation of tunnel links without invoking `dladm create-iptun`. These tunnel names are:

`ip.tun`*N*	An IPv4 tunnel
`ip6.tun`*N*	An IPv6 tunnel
`ip.6to4tun`*N*	A 6to4 tunnel

These tunnels are “implicit tunnels”, denoted with the `i` flag in `dladm show-iptun` output. The tunnel links over which these special IP interfaces are plumbed are automatically created, and they are automatically deleted when the last reference is released (that is, when the last IP interface is unplumbed).

The `tsrc`, `tdst`, `encaplim`, and `hoplimit` options to `ifconfig` are obsolete and maintained only for backward compatibility. They are equivalent to their `dladm(1M)` counterparts.

Display of Tunnel Security Settings

The `ifconfig` output for IP tunnel interfaces indicates whether IPsec policy is configured for the underlying IP tunnel link. For example, a line of the following form will be displayed if IPsec policy is present:

```
tunnel security settings  -->  use 'ipsecconf -ln -i ip.tun1'
```

If you do net set security policy, using either `ifconfig` or `ipsecconf(1M)`, there is no tunnel security setting displayed.

Examples

EXAMPLE 1 Using the `ifconfig` Command

If your workstation is not attached to an Ethernet, the network interface, for example, `eri0`, should be marked "down" as follows:

```
example% ifconfig eri0 down
```

EXAMPLE 2 Printing Addressing Information

To print out the addressing information for each interface, use the following command:

```
example% ifconfig -a
```

EXAMPLE 3 Resetting the Broadcast Address

To reset each interface's broadcast address after the netmasks have been correctly set, use the next command:

```
example% ifconfig -a broadcast +
```

EXAMPLE 4 Changing the Ethernet Address

To change the Ethernet address for interface `ce0`, use the following command:

```
example% ifconfig ce0 ether aa:1:2:3:4:5
```

EXAMPLE 5 Configuring an IP-in-IP Tunnel

To configure an IP-in-IP tunnel, first create an IP tunnel link (`tunsrc` and `tundst` are hostnames with corresponding IPv4 entries in `/etc/hosts`):

```
example% dladm create-iptun -T ipv4 -s tunsrc -d tundst tun0
```

Then plumb a point-to-point interface, supplying the source and destination addresses (`mysrc` and `thedst` are hostnames with corresponding IPv4 entries in `/etc/hosts`):

```
example% ifconfig tun0 plumb mysrc thedst up
```

Use `ipsecconf(1M)`, as described above, to configure tunnel security properties.

EXAMPLE 5 Configuring an IP-in-IP Tunnel *(Continued)*

Configuring IPv6 tunnels is done by using a tunnel type of `ipv6` with `create-iptun`. IPv6 interfaces can also be plumbed over either type of tunnel.

EXAMPLE 6 Configuring 6to4 Tunnels

To configure 6to4 tunnels, first create a 6to4 tunnel link (`myv4addr` is a hostname with a corresponding IPv4 entry in `/etc/hosts`):

```
example% dladm create-iptun -T 6to4 -s myv4addr my6to4tun0
```

Then an IPv6 interface is plumbed over this link:

```
example% ifconfig my6to4tun0 inet6 plumb up
```

The IPv6 address of the interface is automatically set as described above.

EXAMPLE 7 Configuring IP Forwarding on an Interface

To enable IP forwarding on a single interface, use the following command:

```
example% ifconfig eri0 router
```

To disable IP forwarding on a single interface, use the following command:

```
example% ifconfig eri0 -router
```

EXAMPLE 8 Configuring Source Address Selection Using a Virtual Interface

The following command configures source address selection such that every packet that is locally generated with no bound source address and going out on `qfe2` prefers a source address hosted on `vni0`.

```
example% ifconfig qfe2 usesrc vni0
```

The `ifconfig -a` output for the `qfe2` and `vni0` interfaces displays as follows:

```
qfe2: flags=1100843<UP,BROADCAST,RUNNING,MULTICAST,ROUTER,IPv4> mtu
  1500 index 4
  usesrc vni0
  inet 1.2.3.4 netmask ffffff00 broadcast 1.2.3.255
  ether 0:3:ba:17:4b:e1
vni0: flags=20011100c1<UP,RUNNING,NOARP,NOXMIT,ROUTER,IPv4,VIRTUAL>
  mtu 0 index 5
  srcof qfe2
  inet 3.4.5.6 netmask ffffffff
```

Observe, above, the `usesrc` and `srcof` keywords in the `ifconfig` output. These keywords also appear on the logical instances of the physical interface, even though this is a per-physical

EXAMPLE 8 Configuring Source Address Selection Using a Virtual Interface *(Continued)*

interface parameter. There is no `srcof` keyword in `ifconfig` for configuring interfaces. This information is determined automatically from the set of interfaces that have `usesrc` set on them.

The following command, using the `none` keyword, undoes the effect of the preceding `ifconfig usesrc` command.

```
example% ifconfig qfe2 usesrc none
```

Following this command, `ifconfig -a` output displays as follows:

```
qfe2: flags=1100843<UP,BROADCAST,RUNNING,MULTICAST,ROUTER,IPv4> mtu
  1500 index 4
  inet 1.2.3.4 netmask ffffff00 broadcast 1.2.3.255
  ether 0:3:ba:17:4b:e1
vni0: flags=20011100c1<UP,RUNNING,NOARP,NOXMIT,ROUTER,IPv4,VIRTUAL>
  mtu 0 index 5
  inet 3.4.5.6 netmask ffffffff
```

Note the absence of the `usesrc` and `srcof` keywords in the output above.

EXAMPLE 9 Configuring Source Address Selection for an IPv6 Address

The following command configures source address selection for an IPv6 address, selecting a source address hosted on `vni0`.

```
example% ifconfig qfe1 inet6 usesrc vni0
```

Following this command, `ifconfig -a` output displays as follows:

```
qfe1: flags=2000841<UP,RUNNING,MULTICAST,IPv6> mtu 1500 index 3
  usesrc vni0
  inet6 fe80::203:baff:fe17:4be0/10
  ether 0:3:ba:17:4b:e0
vni0: flags=2002210041<UP,RUNNING,NOXMIT,NONUD,IPv6,VIRTUAL> mtu 0
  index 5
  srcof qfe1
  inet6 fe80::203:baff:fe17:4444/128
vni0:1: flags=2002210040<RUNNING,NOXMIT,NONUD,IPv6,VIRTUAL> mtu 0
  index 5
  srcof qfe1
  inet6 fec0::203:baff:fe17:4444/128
vni0:2: flags=2002210040<RUNNING,NOXMIT,NONUD,IPv6,VIRTUAL> mtu 0
  index 5
  srcof qfe1
  inet6 2000::203:baff:fe17:4444/128
```

EXAMPLE 9 Configuring Source Address Selection for an IPv6 Address *(Continued)*

Depending on the scope of the destination of the packet going out on `qfe1`, the appropriately scoped source address is selected from `vni0` and its aliases.

EXAMPLE 10 Using Source Address Selection with Shared-IP Zones

The following is an example of how the `usesrc` feature can be used with the `zones(5)` facility in Solaris. The following commands are invoked in the global zone:

```
example% ifconfig hme0 usesrc vni0
example% ifconfig eri0 usesrc vni0
example% ifconfig qfe0 usesrc vni0
```

Following the preceding commands, the `ifconfig -a` output for the virtual interfaces would display as:

```
vni0: flags=20011100c1<UP,RUNNING,NOARP,NOXMIT,ROUTER,IPv4,VIRTUAL>
   mtu 0 index 23
   srcof hme0 eri0 qfe0
   inet 10.0.0.1 netmask ffffffff
vni0:1:
   flags=20011100c1<UP,RUNNING,NOARP,NOXMIT,ROUTER,IPv4,VIRTUAL> mtu 0
   index 23
   zone test1
   srcof hme0 eri0 qfe0
   inet 10.0.0.2 netmask ffffffff
vni0:2:
   flags=20011100c1<UP,RUNNING,NOARP,NOXMIT,ROUTER,IPv4,VIRTUAL> mtu 0
   index 23
   zone test2
   srcof hme0 eri0 qfe0
   inet 10.0.0.3 netmask ffffffff
vni0:3:
   flags=20011100c1<UP,RUNNING,NOARP,NOXMIT,ROUTER,IPv4,VIRTUAL> mtu 0
   index 23
   zone test3
   srcof hme0 eri0 qfe0
   inet 10.0.0.4 netmask ffffffff
```

There is one virtual interface alias per zone (`test1`, `test2`, and `test3`). A source address from the virtual interface alias in the same zone is selected. The virtual interface aliases were created using `zonecfg(1M)` as follows:

```
example% zonecfg -z test1
zonecfg:test1> add net
zonecfg:test1:net> set physical=vni0
zonecfg:test1:net> set address=10.0.0.2
```

EXAMPLE 10 Using Source Address Selection with Shared-IP Zones *(Continued)*

The `test2` and `test3` zone interfaces and addresses are created in the same way.

EXAMPLE 11 Turning Off DHCPv6

The following example shows how to disable automatic use of DHCPv6 on all interfaces, and immediately shut down DHCPv6 on the interface named `hme0`. See `in.ndpd`(1M) and `ndpd.conf`(4) for more information on the automatic DHCPv6 configuration mechanism.

```
example% echo ifdefault StatefulAddrConf false >> /etc/inet/ndpd.conf
example% pkill -HUP -x in.ndpd
example% ifconfig hme0 dhcp release
```

Files

`/etc/netmasks`
Netmask data.

`/etc/default/inet_type`
Default Internet protocol type.

Attributes

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os
Interface Stability for command-line options	Committed
Interface Stability for command output	Uncommitted

See Also

`dhcpinfo`(1), `cfgadm`(1M), `dhcpagent`(1M), `dladm`(1M), `if_mpadm`(1M), `in.mpathd`(1M), `in.ndpd`(1M), `in.routed`(1M), `ipadm`(1M), `ipmpstat`(1M), `ipsecconf`(1M), `ndd`(1M), `netstat`(1M), `zoneadm`(1M), `zonecfg`(1M), `ethers`(3SOCKET), `gethostbyname`(3NSL), `getnetbyname`(3SOCKET), `hosts`(4), `inet_type`(4), `ndpd.conf`(4), `netmasks`(4), `networks`(4), `nsswitch.conf`(4), `attributes`(5), `privileges`(5), `zones`(5), `arp`(7P), `ipsecah`(7P), `ipsecesp`(7P)

System Administration Guide: IP Services

Diagnostics

`ifconfig` sends messages that indicate if:

- the specified interface does not exist
- the requested address is unknown
- the user is not privileged and tried to alter an interface's configuration

Notes

Do not select the names `broadcast`, `down`, `private`, `trailers`, `up` or other possible option names when you choose host names. If you choose any one of these names as host names, it can cause unusual problems that are extremely difficult to diagnose.

Name if_mpadm – administer interfaces in an IP multipathing group

Synopsis

```
if_mpadm -d | -r ifname
```

Description The `if_mpadm` utility administers IP interfaces that are part of an IP Multipathing (IPMP) group. Currently, administration is limited to offlining IP interfaces and undoing previous offline operations.

When an IP interface is taken offline, all IP data traffic that was flowing over the IP interface is moved to another IP interface in the IPMP group. In addition, all `UP` IP addresses hosted on the IP interface are brought down, causing `in.mpathd(1M)` to stop probe-based failure detection on the IP interface. As a result, an offline IP interface will not be used for any inbound or outbound IP traffic. Only IP interfaces that are in an IPMP group may be brought offline. If the IP interface is the last functioning interface in the IPMP group, the offline operation will fail.

When an offline operation is undone, any IP addresses hosted on that IP interface are brought `UP` and will be considered by `in.mpathd` for probe-based failure detection. In addition, provided the IP interface is otherwise active (see `in.mpathd(1M)`), it will again be used to send and receive IP data traffic for the IPMP group. Note that not all offline operations can be undone. For instance, `in.mpathd` may have offlined an IP interface because its hardware address was not unique within its IPMP group. The `ipmpstat` utility can be used to determine why an IP interface is offline, identify which IP interfaces in a group are being used for inbound and outbound IP traffic, and more; see `ipmpstat(1M)`.

Options The `if_mpadm` utility supports the following options:

`-d` *ifname* Offline the IP interface specified by *ifname*. If *ifname* is not in an IPMP group, or the offline would cause the IPMP group to lose network connectivity, the operation will fail.

`-r` *ifname* Undo a previous offline of the IP interface specified by *ifname*. If *ifname* is not offline, the operation will fail.

Examples **EXAMPLE 1** Offlining an IP Interface

The following command offlines the IP interface `under0`, causing any IP packets that were being sent and received through it to be handled by another IP interface in its group.

```
example% if_mpadm -d under0
```

EXAMPLE 2 Undoing a Previous Offline Operation

Use the following command to undo the operation in the previous example:

```
example% if_mpdadm -r under0
```

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os
Interface Stability	Uncommitted

See Also ifconfig(1M), in.mpathd(1M), ipmpstat(1M), attributes(5)

Diagnostics

`cannot offline: no other functioning interfaces are in its IPMP group.`
Description: This message means that offlining the IP interface would leave the IPMP group without network connectivity.

`cannot offline: not a physical interface or not in an IPMP group`
Description: This means that the IP interface is not an underlying interface in an IPMP group, and therefore is not eligible to be offlined.

Name ifparse – parse `ifconfig` command line

Synopsis `/usr/sbin/ifparse [-fs]` *addr_family commands*

Description Use the `ifparse` command to parse the `ifconfig(1M)` command line options and output substrings, one per line, as appropriate. If no options are specified, `ifparse` returns the entire `ifconfig` command line as a series of substrings, one per line.

Options The `ifparse` command supports the following options:

`-f` Lists only substrings of the `ifconfig` command line that are relevant to IP network multipath failover

`-s` Lists only substrings of the `ifconfig` command line that are not relevant to IP network multipath failover

Operands The `ifparse` command *does not* support the *interface* operand of the `ifconfig` command.

Examples **EXAMPLE 1** Parsing Command Line Options Relevant to Failover

The following example shows the use of the `ifparse` command to parse the command line options relevant to IP network multipath failover:

```
example# ifparse -f inet 1.2.3.4 up group one addif 1.2.3.5 -failover up
set 1.2.3.4 up
```

EXAMPLE 2 Parsing Command Line Options That Are Not Relevant to Failover

The following example shows the use of the `ifparse` command to parse the command line options that are not relevant to IP network multipath failover:

```
example# ifparse -s inet 1.2.3.4 up group one addif 1.2.3.5 -failover up
group one
addif 1.2.3.5 -failover up
```

EXAMPLE 3 Parsing the Command Line For All Options

The following example shows the use of the `ifparse` command to parse the command line for all `ifconfig` options:

```
example# ifparse inet 1.2.3.4 up group one addif 1.2.3.5 -failover up
group one
set 1.2.3.4 up
addif 1.2.3.5 -failover up
```

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Obsolete

See Also ifconfig(1M), attributes(5)

Diagnostics

`usage: -fs <addr_family> <commands>`
Description: This message indicates an invalid command line.

`ifparse: Not enough space`
Description: This message indicates insufficient memory.

`ifparse: dhcp not supported for inet6`
Description: DHCP operations are not supported for the `inet6` address family.

`ifparse: Operation <operation> not supported for <addr_family>`
Description: Most operations cannot be used with all address families. For example, the `broadcast` operation is not supported on the `inet6` address family.

`ifparse: no argument for <operation>`
Description: Some operations, for example `broadcast`, require an argument.

Notes The `ifparse` command is classified as an obsolete interface. It will likely be removed in a future release. You should not develop applications that depend upon this interface.

Name iiadm – command–line interface to control Sun StorageTek Availability Suite Point-in-Time Copy operations

Synopsis

```
iiadm -e {ind | shd} master_vol shadow_vol bitmap_vol

iiadm -ne ind master_vol shadow_vol bitmap_vol

iiadm [-p] [-n] {-c | -u} {s | m} volume_set

iiadm [-adDilR] volume_set

iiadm [-p] [-n] -w volume_set

iiadm [-hilLv]

iiadm -P delay units volume_set

iiadm -P volume_set

iiadm -A overflow_vol volume_set

iiadm [-OQ] overflow_vol

iiadm -E volume_set

iiadm [-IJ] volume_set bitmap

iiadm -g group_name [-aAcdDeEilLmPRuw]

iiadm [-C] cluster_tag [options]
```

Description Point-in-Time Copy software is a point-in-time snapshot feature of the Solaris operating system.

A Point-in-Time Copy snapshot is an instantly-available, time-fixed, replicated view of a momentarily quiesced volume. Once a snapshot is taken, Point-in-Time Copy software allows immediate read/write access to both the master and shadow volume data.

Point-in-Time Copy software tracks the differences between the master and shadow volumes (caused by writes) from the moment that the snapshot was established. This capability allows applications accessing the master volume's data to move forward in time independently of applications accessing the shadow volume's data, and vice-versa.

The Point-in-Time Copy software's tracking of differences between the master and shadow volumes facilitates a fast resynchronization or a full copy at a later time. The volume resynchronization can occur from either shadow to master or master to shadow.

Instantly after the point-in-time is (re-)established (either when the CLI prompt returns or the next shell script command is read), the master volume can be remounted or the applications using them can be resumed. Also, the shadow volume can be mounted and immediately accessed.

The `iiadm` command line utility performs only one action per command invocation. Because of this, you cannot combine multiple options, except in combination with the following overall command modifiers:

- If no action item is entered, `iiadm` displays the list of Point-in-Time Copy sets (non-suspended) currently configured. If more than one action item, or an incorrectly specified action item is entered, `iiadm` displays the specific error message to stderr, followed by a brief usage summary.
- For the Point-in-Time Copy options ENABLE (`-e`), COPY (`-c`) and UPDATE (`-u`), there are two associated shadow volume selection qualifiers, `{ind|dep}`, that are used to specify the type of Point-in-Time Copy volume set to create.

An independent (`ind`) snapshot causes Point-in-Time Copy software to perform a full volume copy operation from the master to the shadow. When the copy completes, the shadow volume data is identical to the master volume data at the moment that it was established. Create an independent shadow if you require two physical copies of the data. An independent shadow volume must be the same size or greater than the size of the master volume. Sun recommends that the master and shadow volumes be the same size for environments where resynchronization from shadow to master is a consideration.

A dependent (`dep`) snapshot causes Point-in-Time Copy software not to perform a full volume copy. The resulting shadow volume relies on the master volume for all unmodified data blocks, which are not copied until requested. Create a dependent shadow when you do not require two physical copies of the data. A dependent shadow volume can be either the same size or smaller than the master volume. A smaller shadow volume is called a *Compact Dependent Shadow Volume*, and is typically used when the amount of change that occurs to a Point-in-Time Copy volume set is small compared to the entire size of the master volume.

The following syntax allows you to create an exportable independent shadow volume in a Sun Cluster environment:

```
# iiadm -ne ind master shadow bitmap
```

An issue arises when using a Compact Dependent Shadow Volume in that its size is established at the time that the Point-in-Time Copy volume set is enabled. If the amount of change to the entire volume set over the duration of its usage exceeds the space allocated for the shadow volume, the shadow volume is marked as out of space. It is possible to read from the shadow volume even after it is out of space, until a portion of the data for which there was no room is requested. Once that happens, the read fails and the shadow volume is marked offline.

To address this issue, Point-in-Time Copy supports the ability to associate an *overflow* volume to an existing Point-in-Time Copy dependent volume set. Thus, if the size of the Compact Dependent Shadow Volume is too small, or an unscheduled amount of change occurs to the volume set, changed data can be redirected to the associated overflow volume. To facilitate efficient usage of this overflow volume, it can be associated with multiple Point-in-Time Copy volume sets on an as-needed basis.

Considerations

Prior to invoking an Point-in-Time Copy `enable`, `copy` or `update` operation, Point-in-Time Copy assures that the shadow volume is not mounted, to prevent a file system panic from occurring. Also, it is suggested that you either unmount or suspend (quiesce) all applications using the master volume, for only the instant when the point-in-time snapshot is taken. This assures that an atomically consistent point-in-time snapshot is taken.

It is suggested that, if the master volume was suspended rather than unmounted, the new point-in-time shadow volume's integrity be validated using volume validation utilities, such as `fsck(1M)`. The reason is that Point-in-Time Copy has made a point-in-time copy of a `mounted` master volume to an `unmounted` shadow volume. During the mounting of the shadow volume, the file system detects that it is in the `mounted` state. Typically this state occurs only when a system crashes, so the file system attempts to validate the integrity of the volume assuming a system failure occurred, not an Point-in-Time Copy.

ENVIRONMENT OPTIONS

The `ii_bitmap` variable in the `/usr/kernel/drv/ii.conf` configuration file determines the bitmap volume operational semantics as follows:

`0` Indicates that the bitmap is maintained in memory only or resume operation.

`1` Indicates that the bitmap is maintained in memory and on disk. This is the default value.

If a system failure occurrs while using `ii_bitmap=0`, the shadow volume might be inconsistent and fast resynchronization would not be possible.

If Point-in-Time Copy is used in conjunction with the Network Storage component Remote Mirror or in a Sun Cluster, set `ii_bitmap=1`.

The `ii_debug` variable in the `/usr/kernel/drv/ii.conf` configuration file determines the amount of information logging that is output to the system console `/dev/console` during Point-in-Time Copy processing.

`0` Indicates that no logging is sent to the system console.

`1` Indicates that informational logging is sent to the system console.

`2` Indicates that developmental logging is sent to the system console.

Options

The `iiadm` utility supports the following options.

`-e{ind|dep}` *master_vol shadow_vol bitmap_vol*
Enable Point-in-Time Copy for the specified master, shadow, and bitmap volumes.

The enable shadow set processing assures that the specified volumes are accessible, that the *shadow_vol* is not mounted, and that the *bitmap_vol* is correctly sized for the type of shadow set being created. Additionally, it assures that the volumes are under control of the SV driver (if they are not, it puts them there), initializes the bitmap volume, and, if the volume set is an independent shadow set, a full copy operation is initiated.

On a successful enable, Point-in-Time Copy stores the specified *master_vol*, *shadow_vol* and *bitmap_vol* names, plus the enabling type (`ind` or `dep`), into the Point-in-Time Copy configuration store. The configuration store contains all currently configured Point-in-Time Copy Volume Sets and their associated configuration attributes. (See discussion above on independent and dependent shadow volume semantics.)

master_vol is the volume from which a point-in-time snapshot is made.

shadow_vol is the volume that contains the point-in-time snapshot.

bitmap_vol is used for tracking differences between the shadow and master volumes. When Point-in-Time Copy shadow operations are suspended or resumed, the bitmap volume (maintained in kernel memory) can be stored in or retrieved from permanent storage. The storage associated with the bitmap volume should be as redundant as that of the shadow volume storage.

The *shadow_vol* name is the name that the Point-in-Time Copy Shadow Set is known by for all `iiadm` options requiring specification of a *volume_set* name.

`-d` *volume_set*
Disable the Point-in-Time Copy volume set associated with the specified *volume_set*.

If Point-in-Time Copy was running in `independent` mode as specified in the `-e ind` options, above, the shadow volume data contains the same data as it did before it was disabled (assuming no writes have occurred). Users can access the master and shadow volumes, as they are now standalone point-in-time copies.

During the time that the full copy is active, an `independent` volume operates as though it is a `dependent` volume. To assure that the volume is no longer in full copy mode, issue the following command to wait for the full copy to complete:

```
# iiadm -w volume_set
```

`[-p] -u s` *volume_set*
Update the shadow volume from the master.

Updates a point-in-time copy of the master volume to the shadow volume. *volume_set* is the Point-in-Time Copy shadow set containing the master and shadow volumes. This option provides a fast resynchronization of the shadow volume, creating an incremental copy of the master. This update copies all 32KB segments flagged as different between the master and shadow volumes. It does not copy all master volume data, only changed data. While the data is being copied, the shadow is dependent upon the master volume.

Before using this option, momentarily quiesce the workload to the volumes; stop the host application from writing to the volumes. This ensures that the point-in-time data is consistent. You can visually check the status of this copy or update operation with `iiadm -i` *volume_set*, or interactively (by means of a shell or script) with `iiadm -w` *volume_set*, before using the target volume for any other operations.

This command supports PID (Process IDentifier) locking, by using the option `-p`, `iiadm -p -u s`. Enabling this option prevents other processes from taking a new point-in-time snapshot, thus invalidating prior point-in-time data.

[`-p`] [`-n`] `-u m` *volume_set*

Updates a point-in-time copy of the master volume from the shadow. *volume_set* is the Point-in-Time Copy volume set containing the master and shadow. This option provides a fast resynchronization of the master volume, creating an incremental copy of the shadow. This update copies all 32KB segments flagged as different between the master and shadow volumes. It does not copy all shadow volume data, only changed data. While the data is being copied, the master is dependent upon the shadow volume.

Before using this option, momentarily quiesce the workload to the volumes; stop the host application from writing to the volumes. This ensures that the point-in-time data is consistent. You can visually check the status of this copy or update operation with `iiadm -i` *volume_set*, or interactively (by means of a shell or script) with `iiadm -w` *volume_set*, before using the target volume for any other operations.

This command is query enabled to prevent accidentally overwriting the data on a master volume. When this command option is used in scripts, add the `-n` option to prevent the query from occurring.

This command supports PID (Process IDentifier) locking, by using the option `-p`, `iiadm -p -u m`. Enabling this option prevents other processes from taking a new point-in-time snapshot, thus invalidating prior point-in-time data.

[`-p`] `-c s` *volume_set*

Copy the master volume to the shadow.

Creates a point-in-time copy of the master volume to the shadow volume. *volume_set* is the Point-in-Time Copy volume set containing the master and shadow. This option writes all data in the point-in-time copy of the master volume to the shadow volume. While the data is being copied from master to shadow, the shadow is dependent on the master volume.

This option performs a full volume copy. Use `iiadm -u s` unless the integrity of the data on the independent shadow volume is in doubt. Otherwise, use this option to synchronize the master and shadow volumes; that is, make the data on each volume match.

Before using this option, momentarily quiesce the workload to the volumes; stop the host application from writing to the volumes. This ensures that the point-in-time data is consistent. You can visually check the status of this copy or update operation with `iiadm -i` *volume_set*, or interactively (by means of a shell or script) with `iiadm -w` *volume_set*, before using the target volume for any other operations.

This command supports PID (Process IDentifier) locking, by using the `-p` option, `iiadm -p -c s`. Enabling this option prevents other processes from taking a new point-in-time snapshot, thus invalidating prior point-in-time data.

`-c m` *volume_set*
Copy the shadow volume to the master.

Creates a point-in-time copy of the shadow volume to the master volume. *volume_set* is the Point-in-Time Copy volume set containing the master and shadow volumes. This option writes all data in the point-in-time copy of the shadow volume to the master volume. While the data is being copied from the shadow to the master, the master is dependent upon the shadow volume.

This option performs a full volume copy. Use `iiadm -u m` unless the integrity of the data on the independent master is in doubt. Otherwise, use this option to synchronize the master and shadow volumes; that is, make the data on each volume match.

Before using this option, momentarily quiesce the workload to the volumes; stop the host application from writing to the volumes. This ensures that the point-in-time data is consistent. You can visually check the status of this copy or update operation with `iiadm -i` *volume_set*, or interactively (by means of a shell or script) with `iiadm -w` *volume_set*, before using the target volume for any other operations.

This command is query-enabled to prevent accidentally overwriting the data on a master volume. When this command option is used in scripts, add the `-n` option to prevent the query from occurring.

This command supports PID (Process IDentifier) locking, by using the `-p` option, `iiadm -p -c m`. Enabling this option prevents other processes from taking a new point-in-time snapshot, thus invalidating prior point-in-time data.

`-a` *volume_set*
Abort any current copy operation that might be active between the master and shadow volumes. *volume_set* is the Point-in-Time Copy volume set containing the master and shadow volumes. After executing `iiadm -a`, the update or copy to the target (master or shadow) volume is incomplete. The target volume is now a dependent copy of the source volume. Reissue the update or copy command option to resynchronize the volumes.

`[-p] [-n] -w` *volume_set*
Wait until any in-progress copy or update operation completes or is aborted. *volume_set* is the Point-in-Time Copy volume set containing the master and shadow volumes.

This option waits until the current Point-in-Time Copy operation is complete, thus preventing a subsequent `iiadm` command (from a shell or script) from executing. Use this command option when you need to be sure the copy or update operation has completed.

This command supports PID (Process IDentifier) unlocking. If a prior copy or update, using a command `iiadm -p {-c|-u} {m|s}`, was invoked with the `-p` option, upon completion of the wait processing, if the current PID was the PID that locked the point-in-time data, this option unlocks the data.

`-i` *volume_set*
Display status for the Point-in-Time Copy currently-enabled or -suspended volume set. *volume_set* is the Point-in-Time Copy volume set containing the master and shadow volumes. If no *volume_set* is specified, status is displayed for all Point-in-Time Copy volume sets that are configured.

`-l`
List all currently configured Point-in-Time Copy volumes.

`-O` *overflow_vol*
This option causes Point-in-Time Copy to initialize the specified *overflow_vol* for subsequent use as an overflow volume in conjunction with Compact Dependent Shadow Volumes. To facilitate efficient, shared usage of this overflow volume, it can be associated with multiple Point-in-Time Copy volume sets on an as-needed basis.

During initialization of the *overflow_vol*, the initiator of this option, must answer the following question: "Initialize this overflow volume? yes/no" A response of either "yes/no" is required before proceeding.

This option supports the `-n` option, so that the requested action is performed without prompting. This option is useful for inclusion in a script. The `-n` option must be specified first. For example, "`iiadm -nO vol`" is valid; "`iiadm -On vol`" is not.

Make sure you want to initialize the data on the specified *overflow_vol*, especially when using the `-n` option.

`-A` *overflow_vol volume_set*
This option enables the specified *overflow_vol*, for subsequent use as an overflow volume in a situation where the size of the Compact Dependent Shadow Volume is too small, or an unscheduled amount of change occurs to the volume set. Overflow changed data would be redirected to the associated overflow volume. *volume_set* is the Point-in-Time Copy volume set containing the master and shadow volumes.

If the *overflow_vol* has not been initialized, this option initializes the *overflow_vol* (see `-O` option), then attaches the *overflow_vol* to the *volume_set*.

If *overflow_vol* was previously initialized, this option attaches the *overflow_vol* to the *volume_set*.

This option supports the `-n` option, so that the requested action is performed without prompting. This option is useful for inclusion in a script. The `-n` option must be specified first. For example, "`iiadm -nA vol`" is valid; "`iiadm -An vol`" is not.

Make sure you want to initialize the data on the specified *overflow_vol*, especially when using the `-n` option.

`-D` *volume_set*
This option removes the overflow volume currently associated with the specified *volume_set*. If the overflow volume is currently in use by the *volume_set*, this operation fails with an "Overflow volume still in use" error message. To resolve this situation, perform one of the operations described below on the *volume_set*. These operations momentarily clear out all overflow writes that are associated with this volume set.

`abort(-a)`
Abort copy operation.

`disable(-d)`
Dissolve the volume set.

`update(-u)`
Update the volume set.

`-L`
This option lists all overflow volumes which are associated with one or more volume sets.

`-Q` *overflow_vol*
This option displays the current status of the *overflow_vol*.

`-E` *volume_set*
Export the independent shadow volume of the Point-in-Time Copy volume set specified by *volume_set*. The shadow volume is to be made available to another host for read/write access, by means of an enabling technology, such as multi-ported devices. This other host is responsible for maintaining a bitmap of differences that is used to merge with locally recorded differences to the master when the shadow volume is rejoined to its master volume. While a shadow volume is exported it must not be subject to an update or copy operation. Perform an `iiadm -w` *volume_set* command prior to invoking an export command.

`-I` *volume_set bitmap_vol*
Import the independent shadow volume of the Point-in-Time Copy volume set specified by *volume_set*. The shadow volume must have been previously exported from a host by means of an enabling technology, such as multi-ported devices. The import operation causes this host to start maintaining a bitmap of differences as the volume is modified. The *bitmap_vol* should not be the same as that used when the shadow volume was originally formed into a shadow group.

After the exported/imported independent shadow volume is no longer needed by the other node, you must enter a disable command so that the *bitmap_vol* and its associated *shadow_vol* are consistent, prior to performing a join operation. For example,

```
# iiadm -d volume_set
```

`-J` *volume_set bitmap_vol*
Join the *volume_set*, using the *bitmap_vol*, with the master volume set of the Point-in-Time Copy volume set. The bitmap volume supplied is read and merged with the original

volume to reconstruct the original volume set consisting of the master, shadow, and bitmap volumes. The *bitmap_vol* to be merged is the one obtained on the node that had imported the independent shadow volume. There must be no write activity to the shadow volume on the importing machine from the time the bitmap is copied over until the shadow is once again imported.

`-g` *group_name* `-m` *volume_set [volume_set2 ...]*
Add one or more existing Point-in-Time Copy *volume_set(s)* into a user specified *group_name*. This association of one or more Point-in-Time Copy volume sets in a group allows the list of `iiadm` options shown below to be performed on all volume sets within the *group_name* as a whole.

Only the commands `COPY` (`-c`) and `UPDATE` (`-u`) are performed atomically across all Point-in-Time Copy sets within the group. All other grouped, `iiadm` commands are performed sequentially on each member of the group.

The syntax of an `iiadm` group command is as follows:

`iiadm -g` *group_name* `[`*options*`]`

The *options* are as follows:

`-a`
Abort copy operation on all sets within *group_name*.

`-A`
Attach *overflow_vol* to all sets within *group_name*.

`-c {s | m}`
Copy shadow/master for all sets within *group_name*.

`-D`
Detach *overflow_vol* from all sets within *group_name*.

`-d`
Disable all sets within *group_name*.

`-E`
Export all volume sets within *group_name*.

`-i`
Status of all volume sets within *group_name*.

`-l`
List all volume sets within *group_name*.

`-L`
List all groups.

`-n`
Do not ask if an update of the master volume is what the user really intended.

`-P`
Set parameters on all volume sets within *group_name*.

`-R`
Reset all volume sets within *group_name*.

`-u {s | m}`
Update shadow/master for all sets within *group_name*.

`-w`
Wait for all volume sets within *group_name*.

`-g -" " -m` *volume_set* [*volume_set2 ...*]
Remove one or more existing Point-in-Time Copy *volume_set(s)* from their currently associated *group_name*. By default, or until moved into a user specified *group_name*, all Point-in-Time Copy *volume_set(s)* are in the blank (" ") group. This association allows all the previously documented `iiadm` group commands to be performed against the blank (" ") `iiadm` *group_name*.

`-C` *cluster_tag*
This Point-in-Time Copy option is a modifier that limits configuration operations to only those volumes belonging to a Sun Cluster Resource Group, or Disk Group.

In a Sun Cluster where the volume manager is Sun Cluster-aware, `iiadm` automatically obtains the correct Disk Group information, therefore this option is typically not required unless the volumes are part of an encompassing Resource Group.

In a Sun Cluster where the volumes are accessible on the local node only, the special *cluster_tag* of `local` is used to indicate volumes that are not part of a Sun Cluster Resource Group or Disk Group.

If "`-L`" is given as a the *cluster_tag* argument, then `iiadm` lists all cluster tags associated with Point-in-Time Copy.

This option is invalid when used on a Solaris system on which the Sun Cluster package has not been installed or configured.

`-h`
Prints the `iiadm` usage summary.

`-v`
Display the current version of the Point-in-Time Copy software components.

Contact Sun Enterprise Services for assistance in using the remaining commands in this section.

`-P` *delay unit volume_set*
Alter the Point-in-Time Copy volume set tuning parameters for the specified *volume_set* to *delay* ticks, every *unit* I/O's. Delay ranges from 2 to 10000 inclusive; unit ranges from 100 to 60000 inclusive.

`-R` *volume*
After a volume has failed, Point-in-Time Copy places it offline. After replacing the volume, place it back online using this option. Associated dependent volumes in the Point-in-Time Copy volume set are also placed online. After the volume is placed online, this command also starts any necessary point-in-time volume updates.

Exit Status

`0` Command completed successfully.

`>0` An error occurred.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	storage/avs/avs-point-in-time-copy
Interface Stability	Committed

See Also `dscfg(1M)`, `svadm(1M)`, `ds.log(4)`, `rdc.cf(4)`, `attributes(5)`, `ii(7D)`, `sv(7D)`

Name iicpbmp – copy Availability Suite Point-In-Time bitmap volumes

Synopsis

```
iicpbmp [-c] old_bitmap new_bitmap...
```

Description The `iicpbmp` command copies an Availability Suite Point-in-Time bitmap volume, rewriting the bitmap header so that it is consistent with the new bitmap volume name. The configuration entry for the shadow set is rewritten to reflect the location of the new bitmap.

No checks on the current use of either the old or new bitmap volumes are made. The `iicpbmp` command should only be run when the Point–In–Time Copy shadow set using the old bitmap is suspended.

Options

`-c` Do not attempt to update the Availability Suite configuration for the Point-in-Time shadow set that uses the bitmap. This option produces a duplicate of the bitmap but does not affect the shadow set using the old bitmap volume.

Operands

old_bitmap new_bitmap
The old and new Point-In-Time bitmap volumes.

Warnings The `iicpbmp` should be run only when a system is in single-user mode. `iicpbmp` makes no attempt to check if an Point–In–Time Copy set is in use at the time the copy is made. Running `iicpbmp` without the `-c` flag while Point–In–Time Copy is using the shadow set results in inconsistencies in the shadow set the next time Point–In–Time Copy is started.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	storage/avs/avs-point-in-time-copy
Interface Stability	Committed

See Also `iiadm(1M)`, `iicpshd(1M)`, `attributes(5)`

Name iicpshd – copy Availability Suite Point-in-Time shadow volume

Synopsis

```
iicpshd [-s] old_shadow new_shadow...
```

Description The `iicpshd` command copies an Availability Suite Instant Image shadow volume, updating the bit map header and Availability Suite configuration to reflect the new shadow volume.

No checks on the current use of either the old or new shadow volumes are made. The `iicpshd` command should only be run when the Instant Image shadow set using the old shadow volume is suspended.

Options The `iicpshd` command supports the following option:

`-s` Update the StorageTek configuration information for the Point-in-Time shadow set, but do *not* copy data from the old shadow volume to the new shadow volume.

Operands A `iicpshd` command line has the following operands:

old_shadow new_shadow
`iicpshd` copies the data of the old Availability Suite Instant Image shadow volume to the new shadow volume and updates the bit map header and configuration data.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	storage/avs/avs-point-in-time-copy
Interface Stability	Committed

See Also `iiadm(1M)`, `iicpbmp(1M)`, `attributes(5)`

Warnings `iicpshd` should be run only when the system is in single-user mode. When you run `iicpshd`, the command makes no attempt to check if a Point-in-Time set is in use. Running `iicpshd` with the `-s` flag while Point-in-Time is using the old shadow volume can result in shadow volume data loss. If you use the `-s` option, you must manually copy the data on the old shadow volume to the new shadow volume.

Name ikeadm – manipulate Internet Key Exchange (IKE) parameters and state

Synopsis

```
ikeadm [-np]

ikeadm [-np] get [debug | priv | stats | defaults]

ikeadm [-np] set [debug | priv] [level] [file]

ikeadm [-np] [get | del] [p1 | rule | preshared] [id]

ikeadm [-np] add [rule | preshared] { description }

ikeadm [-np] token [login | logout] PKCS#11_Token_Object

ikeadm [-np] [read | write] [rule | preshared | certcache] file

ikeadm [-np] dump [p1 | rule | preshared | certcache | groups
     | encralgs | authalgs]

ikeadm [-np] flush [p1 | certcache]

ikeadm help
     [get | set | add | del | read | write | dump | flush | token]
```

Description The `ikeadm` utility retrieves information from and manipulates the configuration of the Internet Key Exchange (IKE) protocol daemon, `in.iked(1M)`.

`ikeadm` supports a set of operations, which may be performed on one or more of the supported object types. When invoked without arguments, `ikeadm` enters interactive mode which prints a prompt to the standard output and accepts commands from the standard input until the end-of-file is reached.

Because `ikeadm` manipulates sensitive keying information, you must be superuser to use this command. Additionally, some of the commands available require that the daemon be running in a privileged mode, which is established when the daemon is started.

For details on how to use this command securely see Security.

Options The following options are supported:

`-n`
: Prevent attempts to print host and network names symbolically when reporting actions. This is useful, for example, when all name servers are down or are otherwise unreachable.

`-p`
: Paranoid. Do not print any keying material, even if saving Security Associations. Instead of an actual hexadecimal digit, print an `X` when this flag is turned on.

Usage

Commands The following commands are supported:

`add`
Add the specified object. This option can be used to add a new policy rule or a new preshared key to the current (running) in.iked configuration. When adding a new preshared key, the command cannot be invoked from the command line, as it will contain keying material. The rule or key being added is specified using appropriate id-value pairs as described in the `ID FORMATS` section.

`del`
Delete a specific object or objects from `in.iked`'s current configuration. This operation is available for IKE (Phase 1) SAs, policy rules, and preshared keys. The object to be deleted is specified as described in the `Id Formats`.

`dump`
Display all objects of the specified type known to `in.iked`. This option can be used to display all Phase 1 SAs, policy rules, preshared keys, implemented Diffie-Helman groups, encryption and authentication algorithms available for Phase 1, or the certificate cache. A large amount of output might be generated by this command.

`flush`
Remove all IKE (Phase 1) SAs or cached certificates from `in.iked`.

Note that flushing the `certcache` will also (as a side-effect) update IKE with any new certificates added or removed.

`get`
Lookup and display the specified object. May be used to view the current debug or privilege level, global statistics and default values for the daemon, or a specific IKE (Phase 1) SA, policy rule, or preshared key. The latter three object types require that identifying information be passed in; the appropriate specification for each object type is described below.

`help`
Print a brief summary of commands, or, when followed by a command, prints information about that command.

`read`
Update the current `in.iked` configuration by reading the policy rules or preshared keys from either the default location or from the file specified.

`set`
Adjust the current debug or privilege level. If the debug level is being modified, an output file may optionally be specified; the output file *must* be specified if the daemon is running in the background and is not currently printing to a file. When changing the privilege level, adjustments may only be made to lower the access level; it cannot be increased using ikeadm.

`write`
Write the current `in.iked` policy rule set or preshared key set to the specified file. A destination file must be specified. This command should not be used to overwrite the existing configuration files.

`token`
Log into a PKCS#11 token object and grant access to keying material or log out and invalidate access to keying material.

`token` can be run as a normal user with the following authorizations:

- `token` login: `solaris.network.ipsec.ike.token.login`
- `token` logout: `solaris.network.ipsec.ike.token.logout`

Object Types debug
Specifies the daemon's debug level. This determines the amount and type of output provided by the daemon about its operations. The debug level is actually a bitmask, with individual bits enabling different types of information.

Description	Flag	Nickname
Certificate management	0x0001	cert
Key management	0x0002	key
Operational	0x0004	op
Phase 1 SA creation	0x0008	phase1
Phase 2 SA creation	0x0010	phase2
PF_KEY interface	0x0020	pfkey
Policy management	0x0040	policy
Proposal construction	0x0080	prop
Door interface	0x0100	door
Config file processing	0x0200	config
Label processing	0x0400	label
All debug flags	0x07ff	all

When specifying the debug level, either a number (decimal or hexadecimal) or a string of nicknames may be given. For example, `88`, `0x58`, and `phase1+phase2+policy` are all equivalent, and will turn on debug for `phase 1 sa` creation, `phase 2 sa` creation, and policy management. A string of nicknames may also be used to remove certain types of information; `all-op` has the effect of turning on all debug *except* for operational messages; it is equivalent to the numbers `1019` or `0x3fb`.

priv
Specifies the daemon's access privilege level. The possible values are:

```
Description                 Level   Nickname
Base level                  0       base
Access to preshared key info 1      modkeys
Access to keying material   2       keymat
```

By default, `in.iked` is started at the base level. A command-line option can be used to start the daemon at a higher level. `ikeadm` can be used to lower the level, but it cannot be used to raise the level.

Either the numerical level or the nickname may be used to specify the target privilege level.

In order to get, add, delete, dump, read, or write preshared keys, the privilege level must at least give access to preshared key information. However, when viewing preshared keys (either using the get or dump command), the key itself will only be available if the privilege level gives access to keying material. This is also the case when viewing Phase 1 SAs.

stats
Global statistics from the daemon, covering both successful and failed Phase 1 SA creation.

Reported statistics include:

- Count of current P1 SAs which the local entity initiated
- Count of current P1 SAs where the local entity was the responder
- Count of all P1 SAs which the local entity initiated since boot
- Count of all P1 SAs where the local entity was the responder since boot
- Count of all attempted P1 SAs since boot, where the local entity was the initiator; includes failed attempts
- Count of all attempted P1 SAs since boot, where the local entity was the responder; includes failed attempts
- Count of all failed attempts to initiate a P1 SA, where the failure occurred because the peer did not respond
- Count of all failed attempts to initiate a P1 SA, where the peer responded
- Count of all failed P1 SAs where the peer was the initiator
- Whether a PKCS#11 library is in use, and if applicable, the PKCS#11 library that is loaded. See Example 11.

defaults
Display default values used by the `in.iked` daemon. Some values can be overriden in the daemon configuration file (see `ike.config(4)`); for these values, the token name is displayed in the `get defaults` output. The output will reflect where a configuration token has changed the default.

Default values might be ignored in the event a peer system makes a valid alternative proposal or they can be overriden by per-rule values established in `ike.config`. In such instances, a `get defaults` command continues to display the default values, not the values used to override the defaults.

p1
: An IKE Phase 1 SA. A `p1` object is identified by an IP address pair or a cookie pair; identification formats are described below.

rule
: An IKE policy rule, defining the acceptable security characteristics for Phase 1 SAs between specified local and remote identities. A rule is identified by its label; identification formats are described below.

preshared
: A preshared key, including the local and remote identification and applicable IKE mode. A preshared key is identified by an IP address pair or an identity pair; identification formats are described below.

Id Formats

Commands like `add`, `del`, and `get` require that additional information be specified on the command line. In the case of the delete and get commands, all that is required is to minimally identify a given object; for the add command, the full object must be specified.

Minimal identification is accomplished in most cases by a pair of values. For IP addresses, the local addr and then the remote addr are specified, either in dot-notation for IPv4 addresses, colon-separated hexadecimal format for IPv6 addresses, or a host name present in the host name database. If a host name is given that expands to more than one address, the requested operation will be performed multiple times, once for each possible combination of addresses.

Identity pairs are made up of a local type-value pair, followed by the remote type-value pair. Valid types are:

prefix
: An address prefix.

fqdn
: A fully-qualified domain name.

domain
: Domain name, synonym for fqdn.

user_fqdn
: User identity of the form *user*@fqdn.

mailbox
: Synonym for `user_fqdn`.

A cookie pair is made up of the two cookies assigned to a Phase 1 Security Association (SA) when it is created; first is the initiator's, followed by the responder's. A cookie is a 64-bit number.

Finally, a label (which is used to identify a policy rule) is a character string assigned to the rule when it is created.

Formatting a rule or preshared key for the add command follows the format rules for the in.iked configuration files. Both are made up of a series of id-value pairs, contained in curly braces ({ and }). See `ike.config(4)` and `ike.preshared(4)` for details on the formatting of rules and preshared keys.

Security

The `ikeadm` command allows a privileged user to enter cryptographic keying information. If an adversary gains access to such information, the security of IPsec traffic is compromised. The following issues should be taken into account when using the `ikeadm` command.

- Is the TTY going over a network (interactive mode)?

  If it is, then the security of the keying material is the security of the network path for this TTY's traffic. Using `ikeadm` over a clear-text telnet or rlogin session is risky. Even local windows may be vulnerable to attacks where a concealed program that reads window events is present.

- Is the file accessed over the network or readable to the world (read/write commands)?

  A network-mounted file can be sniffed by an adversary as it is being read. A world-readable file with keying material in it is also risky.

If your source address is a host that can be looked up over the network, and your naming system itself is compromised, then any names used will no longer be trustworthy.

Security weaknesses often lie in misapplication of tools, not the tools themselves. It is recommended that administrators are cautious when using the `ikeadm` command. The safest mode of operation is probably on a console, or other hard-connected TTY.

For additional information regarding this subject, see the afterward by Matt Blaze in Bruce Schneier's *Applied Cryptography: Protocols, Algorithms, and Source Code in C.*

Examples

EXAMPLE 1 Emptying out all Phase 1 Security Associations

The following command empties out all Phase 1 Security Associations:

```
example# ikeadm flush p1
```

EXAMPLE 2 Displaying all Phase 1 Security Associations

The following command displays all Phase 1 Security Associations:

```
example# ikeadm dump p1
```

EXAMPLE 3 Deleting a Specific Phase 1 Security Association

The following command deletes the specified Phase 1 Security Associations:

```
example# ikeadm del p1 local_ip remote_ip
```

EXAMPLE 4 Adding a Rule From a File

The following command adds a rule from a file:

```
example# ikeadm add rule rule_file
```

EXAMPLE 5 Adding a Preshared Key

The following command adds a preshared key:

```
example# ikeadm
     ikeadm> add preshared { localidtype ip localid local_ip
             remoteidtype ip remoteid remote_ip ike_mode main
             key 1234567890abcdef1234567890abcdef }
```

EXAMPLE 6 Saving All Preshared Keys to a File

The following command saves all preshared keys to a file:

```
example# ikeadm write preshared target_file
```

EXAMPLE 7 Viewing a Particular Rule

The following command views a particular rule:

```
example# ikeadm get rule rule_label
```

EXAMPLE 8 Reading in New Rules from `ike.config`

The following command reads in new rules from the ike.config file:

```
example# ikeadm read rules
```

EXAMPLE 9 Lowering the Privilege Level

The following command lowers the privilege level:

```
example# ikeadm set priv base
```

EXAMPLE 10 Viewing the Debug Level

The following command shows the current debug level

```
example# ikeadm get debug
```

EXAMPLE 11 Using stats to Verify Hardware Accelerator

The following example shows how stats may include an optional line at the end to indicate if IKE is using a PKCS#11 library to accelerate public-key operations, if applicable.

```
example# ikeadm get stats
Phase 1 SA counts:
Current:  initiator:     0    responder:      0
Total:    initiator:    21   responder:      27
Attempted:initiator:    21   responder:      27
```

EXAMPLE 11 Using stats to Verify Hardware Accelerator *(Continued)*

```
Failed:   initiator:     0   responder:       0
            initiator fails include 0 time-out(s)
PKCS#11 library linked in from /opt/system/core-osonn/lib/libpkcs11.so
example#
```

EXAMPLE 12 Displaying the Certificate Cache

The following command shows the certificate cache and the status of associated private keys, if applicable:

```
example# ikeadm dump certcache
```

EXAMPLE 13 Logging into a PKCS#11 Token

The following command shows logging into a PKCS#11 token object and unlocking private keys:

```
example# ikeadm token login "Sun Metaslot"
Enter PIN for PKCS#11 token:
ikeadm: PKCS#11 operation successful
```

Exit Status

The following exit values are returned:

`0` Successful completion.

`non-zero` An error occurred. Writes an appropriate error message to standard error.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os
Interface Stability	Not an Interface

See Also

`in.iked(1M)`, `ike.config(4)`, `ike.preshared(4)`, `attributes(5)`, `ipsec(7P)`

Schneier, Bruce, *Applied Cryptography: Protocols, Algorithms, and Source Code in C*, Second Edition, John Wiley & Sons, New York, NY, 1996.

Notes

As `in.iked` can run only in the global zone and exclusive-IP zones, this command is not useful in shared-IP zones.

Name ikecert – manipulates the machine's on-filesystem public-key certificate databases

Synopsis

```
ikecert certlocal
     [-a | -e | -h | -k | -l | -r | -U | -C | -L]
     [[-p] -T PKCS#11 token identifier]
     [option_specific_arguments]...

ikecert certdb [-a | -e | -h | -l | -r | -U | -C | -L]
     [[-p] -T PKCS#11 token identifier]
     [option_specific_arguments]...

ikecert certrldb [-a | -e | -h | -l | -r]
     [option_specific_arguments]...

ikecert tokens
```

Description The `ikecert` command manipulates the machine's on-filesystem public-key certificate databases. See the "Files" section, below.

`ikecert` has three subcommands, one for each of the three major repositories, plus one for listing available hardware tokens:

- `certlocal` deals with the private-key repository,
- `certdb` deals with the public-key repository, and:
- `certrldb` deals with the certificate revocation list (CRL) repository.
- `tokens` shows the available PKCS#11 tokens for a given PKCS#11 library.

The only supported PKCS#11 library and hardware is the Sun Cryptographic Accelerator 4000.

Options Except for `tokens`, each subcommand requires one option, possibly followed by one or more option-specific arguments.

The `tokens` subcommand lists all available tokens in the PKCS#11 library specified in `/etc/inet/ike/config`.

The following options are supported:

`-a`

certlocal
When specified with the `certlocal` subcommand, this option installs (adds) a private key into the Internet Key Exchange (IKE) local ID database. The key data is read from standard input, and is in either Solaris-only format or unencrypted PKCS#8 DER format. Key format is automatically

detected. PKCS#8 key files in PEM format and files in password protected, encrypted format are not recognized, but can be converted appropriately using tools available in OpenSSL.

This option cannot be used with PKCS#11 hardware objects when the corresponding public certificate is not already present in the IKE database. When importing both a public certificate and a private key, the public portion must be imported first using the `certdb` subcommand.

certdb

When specified with the `certdb` subcommand, this option reads a certificate from standard input and adds it to the IKE certificate database. The certificate must be a `X.509` certificate in `PEM Base64` or `ASN.1 BER` encoding. The certificate adopts the name of its identity.

This option can import a certificate into a PKCS#11 hardware key store one of two ways: Either a matching public key object *and* an existing private key object were created using the `certlocal -kc` option, or if a PKCS#11 token is explicitly specified using the `-T` option.

certrldb

When specified with the `certrldb` subcommand, this option installs (adds) a CRL into the IKE database. The CRL reads from standard input.

`-e` [`-f` pkcs8] *slot*

certlocal
When specified with the `certlocal` subcommand, this option extracts a private key from the IKE local ID database. The key data are written to standard output. The slot specifies which private key to extract. Private keys are only extracted in binary/ber format.

Use this option with extreme caution. See the "Security" section, below.

This option will not work with PKCS#11 hardware objects.

When used in conjunction with "`-f pkcs8`", the private key is extracted in unencrypted PKCS#8 format.

`-e` [`-f` *output-format*] `certspec`

certdb
When specified with the `certdb` subcommand, this option extracts a certificate from the IKE certificate database which matches the certspec and writes it to standard output. The *output-format* option specifies the encoding format. Valid options are `PEM` and `BER`. This extracts the first matching identity. The default output format is `PEM`.

certrldb
When specified with the `certrldb` subcommand, this option extracts a CRL from the IKE database. The key data are written to standard output. The `certspec` specifies which CRL that is extracted. The first one that matches in the database is extracted. See `NOTES`, below, for details on `certspec`

patterns.

`-kc -m` *keysize* `-t` *keytype* `-D` *dname* `-A` *altname*[...]
[`-S` *validity start_time*][`-F` *validity end_time*]
[`-T` *PKCS#11 token identifier*]

certlocal
When specified with the `certlocal` subcommand, this option generates a IKE public/private key pair and adds it into the local ID database. It also generates a certificate request and sends that to standard output. For details on the above options see Notes for details on the *dname* argument and see ALTERNATIVE NAMES for details on the *altname* argument(s) to this command.

If `-T` is specified, the hardware token will generate the pair of keys.

If `-p` is specified with `-T`, the PKCS#11 token pin is stored in the clear on-disk, with root-protected file permissions. If not specified, one must unlock the token with `ikeadm`(1M) once `in.iked`(1M) is running.

`-ks -m` *keysize* `-t` *keytype* `-D` *dname* `-A` *altname*[...]
[`-S` *validity start_time*][`-F` *validity end_time*]
[`-f` *output-format*][[`-p`] `-T` *PKCS#11 token identifier*]

certlocal
When specified with the `certlocal` subcommand, generates a public/private key pair and adds it into the local ID database. This option also generates a self-signed certificate and installs it into the certificate database. See `NOTES`, below, for details on the *dname* and *altname* arguments to this command.

If `-T` is specified, the hardware token will generate the pair of keys, and the self-signed certificate will also be stored in the hardware.

`-l` [`-v`] [*slot*]

certlocal
When specified with the `certlocal` subcommand, this option lists private keys in the local ID database. The `-v` option switches output to a verbose mode where the entire certificate is printed.

Use the `-v`*option with extreme caution.* See the "Security" section, below. The `-v` option will not work with PKCS#11 hardware objects.

`-l` [`-v`] [certspec]

certdb
When specified with the `certdb` subcommand, this option lists certificates in the IKE certificate database matching the certspec, if any pattern is given. The list displays the identity string of the certificates, as well as, the private key if in the key database. The `-v` switches the output to a verbose mode where the entire certificate is printed.

If the matching ceritifcate is on a hardware token, the token ID is also listed.

certrldb
When specified with the `certrldb` subcommand, this option lists the CRLs in the IKE database along with any certificates that reside in the database and match the Issuer Name. `certspec` can be used to

specify to list a specific CRL. The `-v` option switches the output to a verbose mode where the entire certificate is printed. See `NOTES`, below, for details on`certspec` patterns.

`-r` *slot*

certlocal
When specified with the `certlocal` subcommand, deletes the local ID in the specified slot. If there is a corresponding public key, it is not be deleted. If this slot is deemed as "corrupted" or otherwise unrecognizable, it is deleted as well.

If this is invoked on a PKCS#11 hardware object, it will also delete the PKCS#11 public key and private key objects. If the public key object was already deleted by `certdb -r`, that is not a problem.

`-r` certspec

certdb
Removes certificates from the IKE certificate database. Certificates matching the specified certificate pattern are deleted. Any private keys in the `certlocal` database corresponding to these certificates are not deleted. This removes the first matching identity.

If the pattern specifies a slot and the slot is deemed as "corrupted" or otherwise unrecognizable, it is deleted as well.

If this is invoked on a PKCS#11 hardware object, it will also delete the certificate and the PKCS#11 public key object. If the public key

object was already deleted by `certlocal -r`, that is not a problem.

certrldb
When specified with the `certrldb` subcommand, this option deletes the CRL with the given `certspec`.

`-U` slot

`certlocal`
When specified with the `certlocal` subcommand and the `-T` flag, this option unlinks a PKCS#11 private key object from the IKE database. There will be no attempt to access the hardware keystore or to validate or remove the on-token private key object. The object is simply disassociated from the IKE database.

`certdb`
When specified with the `certdb` subcommand and the `-T` flag, this option unlinks a PKCS#11 certificate object from the IKE database. There will be no attempt to access the hardware keystore or to validate or remove the on-token certificate or public key objects. The objects are simply disassociated from the IKE database.

`-C` certspec

certlocal
When specified with the `certlocal` subcommand, this option copies both the private key and its corresponding certificate and the public key from the on-disk keystore to the hardware keystore specified by its PKCS#11 token. This subcommand attempts

to create each of these components, even if one part fails. In all cases, the original on-disk private key and public certificate are still retained and must be deleted separately. Some hardware keystores, such as FIPS-140 compliant devices, may not support migration of private key objects in this manner.

certdb
When specified with the `certdb` subcommand, this option copies the certificate matching the given `certspec` and corresponding public key from the on-disk keystore to the hardware keystore specified by its PKCS#11 token. The original public certificate is still retained and must be deleted separately, if desired.

If `-p` is specified, the PKCS#11 token pin is stored in the clear on-disk, with root-protected file permissions. If not specified, one must unlock the token with `ikeadm(1M)` once `in.iked(1M)` is running.

`-L` pattern

certlocal
When specified with the `certlocal` subcommand, this option links an existing on-token private key object to the IKE database. The object itself remains on the token. This option simply lets the IKE infrastructure know that the object exists, as if it had been originally created on-token with the Solaris IKE utilities.

certdb
When specified with the `certdb` subcommand, this option links an

existing on-token certificate object to the IKE database. The object itself remains on the token. This option simply lets the IKE infrastructure know that the object exists, as if it had been originally created on-token with the Solaris IKE utilities.

If `-p` is specified, the PKCS#11 token pin is stored in the clear on-disk, with root-protected file permissions. If not specified, one must unlock the token with `ikeadm`(1M) once `in.iked`(1M) is running.

Parameters The following parameters are supported:

certspec

Specifies the pattern matching of certificate specifications. Valid `certspecs` are the Subject Name, Issuer Name, and Subject Alternative Names.

These can be specified as certificates that match the given `certspec` values and that do not match other `certspec` values. To signify a `certspec` value that is not supposed to be present in a certificate, place an `!` in front of the tag.

Valid `certspecs` are:

```
<Subject Names>
SUBJECT=<Subject Names>
ISSUER=<Issuer Names>
SLOT=<Slot Number in the certificate database>

Example:"ISSUER=C=US, O=SUN" IP=1.2.3.4 !DNS=example.com
Example:"C=US,   O=CALIFORNIA" IP=5.4.2.1 DNS=example.com
```

Valid arguments to the alternative names are as follows:

```
IP=<IPv4 address>
DNS=<Domain Name Server address>
EMAIL=<email (RFC 822) address>
URI=<Uniform Resource Indicator value>
DN=<LDAP Directory Name value>
RID=<Registered Identifier value>
```

Valid Slot numbers can be specified without the keyword tag. Alternative name can also be issued with keyword tags.

`-A`
Subject Alternative Names the certificate. The argument that follows the `-A` option should be in the form of *tag=value*. Valid tags are `IP`, `DNS`, `EMAIL`, `URI`, `DN`, and `RID` (See example below).

`-D`
`X.509` distinguished name for the certificate subject. It typically has the form of: `C`=country, `O`=organization, `OU`=organizational unit, `CN`=common name. Valid tags are: `C`, `O`, `OU`, and `CN`.

`-f`
Encoding output format. `pem` for `PEM` `Base64` or `ber` for `ASN.1` `BER`. If `-f` is not specified, `pem` is assumed.

`-F` *validity end_time*
Finish certificate validity time. If the `-F` flag is not specified, the validity end time is calculated at four years from the validity start time. See `NOTES` for an explanation for the validity date and time syntax.

`-m`
Key size. It can be `512`, `1024`, `2048`, `3072`, or `4096`. Use the following command to determine the key sizes supported by the Solaris Cryptographic Framework:

```
% cryptoadm list -vm
```

The mechanisms displayed by the preceding command are described in `pkcs11_softtoken(5)`. If your system has hardware acceleration, the mechanisms supported by the hardware will be listed in a separate section for each provider. Mechanisms can be any of:

```
CKM_RSA_PKCS_KEY_PAIR_GEN
CKM_DSA_KEY_PAIR_GEN
CKM_DH_PKCS_KEY_PAIR_GEN
```

Note – Some hardware does not support all key sizes. For example, the Sun Cryptographic Accelerator 4000's keystore (when using the `-T` option, below), supports only up to 2048-bit keys for RSA and 1024-bit keys for DSA.

`-S` *validity start_time*
Start certificate validity time. If the `-S` flag is not specified, the current date and time is used for the validity start time. See `NOTES`, below, for an explanation for the validity date and time syntax.

`-t`
Key type. It can be `rsa-sha1`, `rsa-md5`, or `dsa-sha1`.

`-T`
PKCS#11 token identifier for hardware key storage. This specifies a hardware device instance in conformance to the PKCS#11 standard. A PKCS#11 library must be specified in `/etc/inet/ike/config`. (See `ike.config(4)`.)

A token identifier is a 32-character space-filled string. If the token given is less than 32 characters long, it will be automatically padded with spaces.

If there is more than one PKCS#11 library on a system, keep in mind that only one can be specified at a time in `/etc/inet/ike/config`. There can be multiple tokens (each with individual key storage) for a single PKCS#11 library instance.

Security This command can save private keys of a public-private key pair into a file. Any exposure of a private key may lead to compromise if the key is somehow obtained by an adversary.

The PKCS#11 hardware object functionality can address some of the shortcomings of on-disk private keys. Because IKE is a system service, user intervention at boot is not desireable. The token's PIN, however, is still needed. The PINfor the PKCS#11 token, therefore, is stored where normally the on-disk cryptographic keys would reside. This design decision is deemed acceptable because, with a hardware key store, *possession* of the key is still unavailable, only *use* of the key is an issue if the host is compromised. Beyond the PIN, the security of `ikecert` then reduces to the security of the PKCS#11 implementation. The PKCS#11 implementation should be scrutinized also.

Refer to the afterword by Matt Blaze in Bruce Schneier's *Applied Cryptography: Protocols, Algorithms, and Source Code in C* for additional information.

Examples **EXAMPLE 1** Generating a Self-Signed Certificate

The following is an example of a self-signed certificate:

```
example# ikecert certlocal -ks -m 512 -t rsa-md5 -D "C=US, O=SUN" -A
IP=1.2.3.4
Generating, please wait...
Certificate generated.
Certificate added to database.
-----BEGIN X509 CERTIFICATE-----
MIIBRDCB76ADAgECAgEBMA0GCSqGSIb3DQEBBAUAMBsxCzAJBgNVBAYTAlVTMQww
CgYDVQQKEwNTVU4wHhcNMDEwMzE0MDEzMDM1WhcNMDUwMzE0MDEzMDM1WjAbMQsw
CQYDVQQGEwJVUzEMMAoGA1UEChMDU1VOMFowDQYJKoZIhvcNAQEBBQADSQAwRgJB
APDhqpKgjgRoRUr6twTMTtSuNsReEnFoReVer!ztpXpQK6ybYlRH18JIqU/uCV/r
26R/cVXTy5qc5NbMwA40KzcCASOjIDAeMAsGA1UdDwQEAwIFoDAPBgNVHREECDAG
hwQBAgMEMA0GCSqGSIb3DQEBBAUAA0EApTRD23KzN95GMvPD71hwwCluk slKLVg8
f1xm9ZsHLPJLRxHFwsqqjAad4j4wwwriiUmGAHLTGB0lJMl8xsgxag==
-----END X509 CERTIFICATE-----
```

EXAMPLE 2 Generating a CA Request

Generating a `CA` request appears the same as the self-signed certificate. The only differences between the two is the option `-c` instead of `-s`, and the certificate data is a `CA` request.

```
example# ikecert certlocal -kc -m 512 -t rsa-md5 \
   -D "C=US, O=SUN" -A IP=1.2.3.4
```

EXAMPLE 3 A CA Request Using a Hardware Key Store

The following example illustrates the specification of a token using the `-T` option.

```
example# # ikecert certlocal -kc -m 1024 -t rsa-md5 -T vca0-keystore \
  -D "C=US, O=SUN" -A IP=1.2.3.4
```

Exit Status

The following exit values are returned:

`0`
Successful completion.

`non-zero`
An error occurred. Writes an appropriate error message to standard error.

Files

`/etc/inet/secret/ike.privatekeys/*`
Private keys. A private key *must* have a matching public-key certificate with the same filename in `/etc/inet/ike/publickeys/`.

`/etc/inet/ike/publickeys/*`
Public-key certificates. The names are only important with regard to matching private key names.

`/etc/inet/ike/crls/*`
Public key certificate revocation lists.

`/etc/inet/ike/config`
Consulted for the pathname of a PKCS#11 library.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os
Interface Stability	Committed

See Also

`ikeadm(1M)`, `in.iked(1M)`, `getdate(3C)`, `ike.config(4)`, `attributes(5)`, `pkcs11_softtoken(5)`

Schneier, Bruce. *Applied Cryptography: Protocols, Algorithms, and Source Code in C*. Second Edition. John Wiley & Sons. New York, NY. 1996.

RSA Labs, PKCS#11 v2.11: *Cryptographic Token Interface Standards*, November 2001.

Notes

The following is the validity date and time syntax when the `-F` or `-S` flags are used:

For relative dates, the syntax is as follows:

```
{+,-}[Ns][Nm][Nh][Nd][Nw][NM][Ny]
```

where:

N
    represents an integer

s
    represents seconds

m
    represents minutes

h
    represents hours

d
    represents days

w
    represents weeks

M
    represents months

y
    represents years

These parameters can be given in any order. For example, "+3d12h" is three and a half days from now, and "-3y2M" is three years and 2 months ago.

All parameters with fixed values can be added up in absolute seconds. Months and years, which have variable numbers of seconds, are calculated using calendar time. Months and years, which are not of fixed length, are defined such that adding a year or month means the same day next year or month. For instance, if it is Jan 26, 2005 and the certificate should expire 3 years and 1 month from today, the expiration (end validity time) date will be Feb 26, 2008. Overflows are dealt with accordingly. For example, one month from Jan 31, 2005 is March 3, 2005, since February has only 28 days.

For absolute dates, the syntax of the date formats included in the file `/etc/datemsk` are accepted (See `getdate(3C)` for details). Any date string prepended with a "+" or "-" is treated as a time relative to the current time, while others are treated as absolute dates. Sanity checking is also done to ensure that the end validity date is greater than the start validity date. For example, the following command would create a certificate with start date 1 day and 2 hours ago and an end date of Jan 22nd, 2007 at 12:00:00 local time.

```
# ikecert certlocal -ks -t rsa-sha1 -m 1024 \
    -D "CN=mycert, O=Sun, C=US" \
    -S -1d2h -F "01/22/2007 12:00:00"
```

As `in.iked(1M)` can run only in the global zone and exclusive-IP zones, this command is not useful in shared-IP zones.

Name ilbadm – establish and manipulate load balancing rules

Synopsis

```
ilbadm create-rule [-e] [-p] -i vip=value,port=value[,protocol=value]
     -m lbalg=value,type=value[,proxy-src=ip-range][,pmask=mask]
     [-h hc-name=value[,hc-port=value]]
     [-t [conn-drain=N][,nat-timeout=N],[persist-timeout=N]]
     -o servergroup=value name

ilbadm show-rule [-e|-d] [-f |[-p] -o key[,key ...]] [name ...]

ilbadm delete-rule -a | name ...

ilbadm enable-rule [name ...]

ilbadm disable-rule [name ...]

ilbadm show-statistics [-p] -o field[,field] [-thAdvi]
     [-r rulename] | [-s servername] [interval [count]]

ilbadm create-servergroup [-s server=hostspec[:portspec...]] groupname

ilbadm delete-servergroup groupname

ilbadm show-servergroup [-s|-f|[-p] -o field[,field]] [[-v] name]

ilbadm enable-server server ...

ilbadm disable-server server ...

ilbadm show-server [[-p] -o field[,field...]] [rulename...]

ilbadm add-server -s server=value[,value ... ] name

ilbadm remove-server -s server=value[,value ... ] name

ilbadm create-healthcheck [-n] -h hc-test=value
     [,hc-timeout=value][,hc-count=value][,hc-interval=value] hcname

ilbadm delete-healthcheck hcname

ilbadm show-healthcheck [hcname ...]

ilbadm show-hc-result [rule-name]

ilbadm show-nat [count]

ilbadm show-persist [count]

ilbadm export-config filename

ilbadm import-config [-p] filename
```

Description The `ilbadm` command manipulates or displays information about Integrated Load Balancer (ILB) rules using the subcommands described below.

Rule names are case insensitive, but case is preserved as it is entered. Rule names are limited in length to 19 characters. Server names cannot exceed 14 characters.

All parseable output (invoked with the `-p` option) requires that the fields to be printed or displayed be specified with the `-o` option. Fields will be displayed in the same order they are encountered on the command line. Multiple fields are separated by the colon (`:`) character. If a colon or backslash (`\`) occurs in the displayed string itself, it will be preceded by a backslash. No headers will be displayed for parseable output.

Server IDs are generated by the system when a server is added, using either the `create-servergroup` or the `add-server` subcommands.

Server IDs are guaranteed to be unique within the server group. A rule can be attached to only one server group, with the result that serverIDs are unique for rules as well. Note that since more than one rule can attach to the same server group, the server ID alone is not sufficient to indicate a rule.

To be able to distinguish server IDs from hostnames, server IDs are prefixed with a leading underscore (_).

As noted below, the server group and heathcheck entities must be defined before they can be used in the `create-rule` subcommand.

Sub-commands Following are the `ilbadm` subcommands, along with their related options and operands. Note that subcommands have a normal and a short form; for example, `create-rule` and `create-rl`, saving you from having to type a few additional characters.

`create-rule`|`create-rl` [`-e`] [`-p`] `-i` *incoming* `-m` *method_attributes* `-o` *outgoing_spec* [`-h` *healthcheck*] [`-t` *timers*] *name*

Creates a rule *name* with a set of specified characteristics. *incoming* and *method_attributes* are both specified as a set of *key=value* pairs. If *name* already exists, the command will fail. If a given tuple (virtual IP address, port(s), and protocol) matches another rule, the command will also fail. `create-rule` has the following options that control the overall effect of the command:

`-e` Enable the `create-rule` function. The default is that `create-rule` is disabled.

`-p` Create the rule as persistent (sticky). The default is that the rule exists only for the current session.

Keys and values are introduced by one-letter identifiers. These identifiers and their related keys and acceptable values are as follows.

`-i`

Introduces the matching criteria for incoming packets.

`vip`

(Virtual) destination IP address

`port[-port]`
Port number or name, for example, `telnet` or `dns`. A port can be specified by port number or symbolic name (as in `/etc/services`). Port number ranges are also supported.

`protocol`
`TCP` (the default) or `UDP` (see `/etc/services`).

`-m`
Specifies the keys describing how to handle a packet.

`lbalg`
The default is `roundrobin`, or its short form, `rr`. Other alternatives are: `hash-ip` (short form: `hip`), `hash-ip-port` (short form: `hipp`), `hash-ip-vip` (short form: `hipv`).

`type`
Refers to topology of network. Can be `DSR` (or `dsr` or `d`), `NAT` (or `n` or `nat`), `HALF-NAT` (or `h` or `half-nat`).

`proxy-src`
Required for full NAT only. Specifies the IP address range to use as the proxy source address range. The range is limited to ten IP addresses.

`pmask`
Optional. Has an alias of: `stickiness`. Specifies that this rule is to be persistent. The argument is a prefix length in CIDR notation; that is, 0–32 for IPv4 and 0–128 for IPv6. Use the `-p` option to specify this keyword.

`-o`
Specifies destination(s) for packets that match the criteria specified by the `-i` "clause". This identifier has one well-known argument:

servergroup Specify a single server group as target. The server group must already have been created.

`-h`
The health check option has two arguments:

`hc-name`
Specifies the name of a predefined health check method

`hc-port`
Specifies the port(s) for the HC test program to check. The value can be keywords `ALL` or `ANY`, or a specific port number within the port range of the server group.

`-t`
Specifies customized timers, in seconds. A value of `0` means to use the system default value. The following are valid modifiers for `-t`:

`conn-drain`
If a server's `type` is `NAT` or `HALF-TYPE`, `conn-drain` is the timeout after which the server's connection state is deleted following the server's removal from a rule. This deletion occurs even if the server is not idle.

The default for TCP is that the connection state remains stable until the connection is gracefully shutdown. The default for UDP is that the connection state remains stable until the connection has been idle for the period `nat-timeout`.

`nat-timeout`
Applies only to NAT and half-NAT type connections. If such a connection is idle for the `nat-timeout` period, the connection state will be removed. The default is `120` for TCP and `60` UDP.

`persist-timeout`
When persistent mapping is enabled, if a numeric-only mapping has not been used for `persist-timeout` seconds, the mapping will be removed. The default is 60.

Note that server group and health check must be defined before they can be used in `create-rule`.

`delete-rule`|`delete-rl` `-a` *name*[...]
Remove all information pertaining to rule *name*. If *name* does not exist, command will fail. `delete-rule` has one option:

`-a`
Delete all rules. (*name* is ignored.)

`enable-rule`|`enable-rl` *name*[...]
Enables a named rule, or all rules, if no name is specified). Enabling rules that are already enabled has no effect.

`disable-rule`|`disable-rl` *name*[...]
Disables a named rule, or all rules, if no name is specified. Disabling rules that are already disabled has no effect.

`show-statistics`|`show-stats` [[`-p`] `-o` *field*[,...]] [`-tv`] [`-A` | `-d`] [[`-i`] `-r` *rulename* | `-s` *servername*] [*interval* [*count*]]
Displays statistics, the output of which is subject to the use of the options described below. The syntax and semantics of this subcommand are modeled on `vmstat(1M)`.

`-t`
Prepend a timestamp with every sample.

`-d`
Display the delta over entire interval. The default is changes per second. Cannot be used with the `-a` option.

`-A`
Display absolute numbers. That is, numbers since module initialization, rule creation, and server addition. Cannot be used with the `-d` option.

`-r` *rulename*
Display statistics only for the specified *rulename*. In combination with the `-i` option, display a line for each server.

`-s` *servername*
Display statistics only for *server*. In combination with the `-i` option, display a line for each rule.

`-i`
Itemize the information displayed by the `-r` and `-s` options. These are the only options with which `-i` is valid. Does not work with the `-v` option.

`-v`
Display additional details for droppages. Note that, when the rule name is specified, drops are counted per rule and not per server. Does not work with the `-i` option.

`-p`
Display parseable format. Requires use of `-o` option.

`-o` *field*
Can be one or more from the list below. *field* can be uppercase or lowercase.

`PKT_P`	Packets processed.
`BYTES_P`	Bytes processed.
`PKT_U`	Unprocessed packets.
`BYTES_U`	Unprocessed bytes.
`PKT_D`	Packets dropped.
`BYTES_D`	Bytes dropped.
`ICMP_P`	ICMP echo requests processed.
`ICMP_D`	ICMP echo requests dropped.
`ICMP2BIG_P`	ICMP fragmentation needed; message processed.
`ICMP2BIG_D`	Fragmentation needed; message dropped.
`NOMEMP_D`	Packets dropped because of out-of-memory condition.
`NOPORTP_D`	Packets dropped in NAT mode because no source port was available.

Note that when a question mark (?) is displayed as a column entry, it indicates that the proper value cannot be determined, most often because a rule or server was added or deleted.

Note that headers are displayed once for each ten samples. The timestamp format follows the date(1) format for the C locale. Neither the addition nor removal of a rule is detected.

`show-rule`|`show-rl` [`-d`|`-e`] [`-f`| [`-p`] `-o` *field*[,...]] [*name*...]
Displays characteristics of the specified rules, or all, if no rule is specified. The subcommand has the following options:

`-d`
Display only disabled rules.

`-e`
Display only enabled rules.

`-f`
Display a full list.

`-o` *field*[,...]
Display output for *field*(s). Cannot be used with `-f` option.

`-p`
Display parsable output in the format described in "Description". Requires the `-o` option.

Note that the `-o` (with or without `-p`) and `-f` options are mutually exclusive.

`show-nat` *count*
Displays NAT table information. If *count* is specified, displays *count* entries from the NAT table. If no count is specified, displays the entire NAT table.

count

No assumptions should be made about the relative positions of elements in consecutive runs of this command. For example, executing `show-nat 10` twice is not guaranteed to display the same ten items twice, especially on a busy system.

Display format:

```
T: IP1 > IP2 >>> IP3 > IP4
```

These items are described as follows:

`T` The transport protocol used in this entry.

`IP1` The client's IP address and port.

`IP2` The VIP and port.

`IP3` If half NAT mode, the client's IP address and port. If full NAT mode, the NAT'ed client's IP address and port.

`IP4` The backend server's IP address and port.

`show-persist`|`show-pt` *count*
Displays persistence table information. If *count* is specified, displays *count* entries from the table. If no count is specified, displays the entire persistence table.

No assumptions should be made about the relative positions of elements in consecutive runs of this command. For example, executing `show-persist 10` twice is not guaranteed to display the same ten items twice, especially on a busy system.

Display format:

```
R: IP1 --> IP2
```

These items are described as follows:

`R`
The rule this persistence entry is tied to.

`IP1`
The client's IP address and port.

`IP2`
The backend server's IP address.

`export-config`|`export-cf` [*filename*]
Exports the current configuration in a format suitable for re-import using `ilbadm import`. If no filename is specified, the subcommand writes to stdout.

`import-config`|`import-cf` [`-p`] [*filename*]
Reads configuration contents of a file. By default, this overrides any existing configuration. If no filename is specified, the subcommand reads from stdin. This subcommand has the following option:

`-p`
Preserve existing configuration and do incremental import.

`create-servergroup`|`create-sg` [`-s` *server=hostspec*[:*portspec*...]] *groupname*
Creates a server group. Additional servers can be added later using the `add-server` subcommand. Server groups are the only entity that can be used during rule creation to indicate back-end servers. If the specified server group is associated with one or more rules, the server is enabled when it is added. This subcommand has the following option and operands:

`-s` *server=hostspec*[:*portspec*...]
Specifies a list of servers to be added to the server group.

hostspec is a hostname or IP address. IPv6 addresses must be enclosed in brackets (`[]`) to distinguish them from "*:portspec*"

portspec is a service name or port number. If the port number is not specified, a number in the range 1–65535 is used.

`disable-server`|`disable-srv` *server*
Disable one or more server(s). That is, tell the kernel not to forward traffic to this server. `disable-server` applies to all rules that are attached to the server group this server is part of.

server is a server ID.

`enable-server`|`enable-srv` *server*...
Reenables disabled servers.

`show-server`|`show-srv` [[`-p`] `-o` *field*[,*field*...]] [*rulename*...]
Displays servers associated with named rules, or all servers if no rulename is specified. The subcommand has the following options.

`-o` *field*[,*field*...]
Display only the specified fields.

`-p`
Display fields in parsable format. Requires the `-o` option.

`delete-servergroup`|`delete-sg` *groupname*
Deletes a server group.

`show-servergroup`|`show-sg` [[`-p`] `-o` *field*[,...]] [*name*]
Lists a server group, or all server groups, if no *name* is specified. The subcommand has the following options:

`-o` *field*[,...]
Display output for *field*(s).

`-p`
Display parsable output in the format described in "Description". Requires the `-o` option.

`add-server`|`add-srv` `-s` *server=value*[, *value*...] *servergroup*
Add specified server(s) to *servergroup*. See description of `create-servergroup` for definition of *value*.

`-s`
See `create-servergroup`.

Performing an `add-server` to a server group immediately after performing a `remove-server` on that server group might fail because of incomplete connection draining. Refer to the description of the `remove-server` subcommand for instructions on how to avoid this failure.

`remove-server`|`remove-srv` `-s` *server=value*[, *value*...] *servergroup*
Remove specified server(s) from *servergroup*.

`-s`
One or more of a server ID.

If a server is being used by a NAT/half-NAT rule, it is recommended that the server be disabled (using `disable-server`) before removal. By disabling a server, the server enters the connection-draining state. After all of the connections are drained, the server can then be removed by `remove-server`. If the `conn-drain` timeout value is set, the connection-draining state will be finished upon conclusion of the timeout period. Note that the default `conn-drain` timeout is 0, meaning it will keep waiting until a connection is gracefully shut down.

`create-healthcheck`|`create-hc` [`-n`] `-h` hc-test=*value*,hc-timeout=*value*, hc-count=*num_value*,hc-interval=*value hcname*

Sets up a health check object for rules to use. All servers associated with a rule are checked using the same test. A health check event of a server consists of one to `hc-count` number of `hc-test` executions. If an `hc-test`'s result shows a server to be unresponsive, further `hc-test` checks are made, up to `hc-count` invocations, before a server is considered to be down.

`-h`

The `hc-test` is performed `hc-count` times until it succeeds or `hc-timeout` has expired. For a given rule, all servers are checked using the same test. The tests are as follows:

`hc-test`

`PING`, `TCP`, external method (script or binary). An external method should be specified with a full path name.

`hc-timeout`

Threshold at which a test is considered failed following interim failures of `hc-test`. If you kill an `hc-test` test, the result is considered a failure. The default value is five seconds.

`hc-count`

Maximum number of attempts to run `hc-test` before marking a server as down. The default value is three iterations.

`hc-interval`

Interval between invocations of `hc-test`. This value must be greater than `hc-timeout` times `hc-count`. The default value is 30 seconds.

The following arguments are passed to external methods:

`$1`

VIP (literal IPv4 or IPv6 address).

`$2`

Server IP (literal IPv4 or IPv6 address).

`$3`

Protocol (`UDP`, `TCP` as a string).

`$4`

The load balance mode, `DSR`, `NAT`, `HALF_NAT`.

`$5`
Numeric port.

`$6`
Maximum time (in seconds) the method should wait before returning failure. If the method runs for longer, it can be killed, and the test considered failed.

External methods should return `0` (or the round-trip time to the back end server, in microseconds) for success and `-1` if the server is considered down.

Before higher layer health check(s), TCP, UDP, and external tests start, a default `ping` test is performed first. The higher layer test will not be performed if `ping` fails. You can turn off the default `ping` check for these high layer health checks by through use of `-n`.

`-n`
Disable default `ping` test for high layer health check tests.

`delete-healthcheck`|`delete-hc` *hcname*...
Delete the named health check object(s) (*hcname*). If the given health check object is associated with enabled rule(s), deletion of the object will fail.

`show-healthcheck`|`show-hc` [*hcname*...]
List the health check information for the specified health check (*hcname*). If no health check is specified, list information for all existing health checks.

`show-hc-result`|`show-hc-res` [*rule-name*]
List the health check result for the servers that are associated with *rule-name*. If *rule-name* is not given, the health check results for all servers are displayed.

Examples

EXAMPLE 1 Configuring NAT Mode

The following commands create a rule with health check and timers set (port range shifting and session persistence).

```
# ilbadm create-healthcheck -h hc-test=tcp,hc-timeout=2,hc-count=3, \
     hc-interval=10 hc1
# ilbadm create-servergroup -s \
     server=60.0.0.10:6000-6009,60.0.0.11:7000-7009 sg1
# ilbadm create-rule -e -i vip=81.0.0.10,port=5000-5009,protocol=tcp \
-m lbalg=rr,type=NAT,proxy-src=60.0.0.101-60.0.0.104, \
     pmask=24 \
-h hc-name=hc1 \
-t conn-drain=180,nat-timeout=180,persist-timeout=180 \
-o servergroup=sg1 rule1
```

The following command creates a rule with the default timer values and without health check.

```
# ilbadm create-servergroup -s server=60.0.0.10 sg1
 # lbadm create-rule -e -i vip=81.0.0.10,port=5000 \
     -m lbalg=rr,type=NAT,proxy-src=60.0.0.105 \
```

EXAMPLE 1 Configuring NAT Mode *(Continued)*

```
     -o servergroup=sg1 rule1
# ilbadm add-server -e -s server=60.0.0.11sg1
# ilbadm enable-rule rule1
```

EXAMPLE 2 Configuring Half-NAT Mode

The following command configures half-NAT mode and exemplifies port range collapsing.

```
# ilbadm create-servergroup sg1
# ilbadm create-rule -e -i vip=81.0.0.10,port=5000-5009 \
     -m lbalg=rr,type=h -o servergroup=sg1 rule1
# ilbadm add-server -s server=60.0.0.10:6000,60.0.0.11:7000 sg1
```

EXAMPLE 3 Configuring DSR Mode and Preparing Two Sets of Rules

The following command establishes two sets of rules to enable load balancing between HTTP and FTP traffic. Note both types of traffic traverse interface `60.0.0.10`.

```
# ilbadm create-servergroup -s servers=60.0.0.9,60.0.0.10 websg
# ilbadm create-servergroup -s servers=60.0.0.10,60.0.0.11 ftpgroup

# ilbadm create-rule -e -i vip=81.0.0.10,port=80 \
     -m lbalg=hash-ip-port,type=DSR \
     -o servergroup=websg webrule
# ilbadm create-rule -e -i vip=81.0.0.10,port=ftp \
     -m lbalg=hash-ip-port,type=DSR,pmask=24 \
     -o servergroup=ftpgroup ftprule
# ilbadm create-rule -e -p -i vip=81.0.0.10,port=ftp-data \
     -m lbalg=hash-ip-port,type=DSR,pmask=24 \
     -o servergroup=ftpgroup ftpdatarule
```

EXAMPLE 4 Deleting Rule, Server Group, and Health Check

The following commands delete the rule, server group, and health check established in the first example.

```
# ilbadm ilbadm delete-rule -a
# ilbadm delete-servergroup sg1
# ilbadm delete-healthcheck hc1
```

EXAMPLE 5 Display a List of Rules

The following command displays a list of rules.

```
# ilbadm show-rule
RULENAME            STATUS LBALG      TYPE    PROTOCOL VIP  PORT
r2                  E      hash-ip    NAT     TCP 45.0.0.10 81
r1                  E      hash-ip    NAT     TCP 45.0.0.10 80
```

EXAMPLE 5 Display a List of Rules *(Continued)*

```
# ilbadm show-rule -f
       RULENAME: rule1
         STATUS: E
           PORT: 80
       PROTOCOL: TCP
          LBALG: roundrobin
           TYPE: HALF-NAT
      PROXY-SRC: --
        PERSIST: --
        HC-NAME: hc1
        HC-PORT: ANY
     CONN-DRAIN: 0
    NAT-TIMEOUT: 120
PERSIST-TIMEOUT: 60
    SERVERGROUP: sg1
            VIP: 80.0.0.2
        SERVERS: _sg1.0,_sg1.1
```

EXAMPLE 6 Exporting and Importing Rules

The following commands show how to export rules to and import rules from stdout, and to/from a file.

```
# ilbadm export-config

create-servergroup ftpgroup
add-server -s server=10.1.1.3:21 ftpgroup
add-server -s server=10.1.1.4:21 ftpgroup
create-servergroup webgroup_v6
add-server -s server=[2000::ff]:80 webgroup_v6
create-rule -e protocol=tcp,VIP=1.2.3.4,port=ftp \
        -m lbalg=roundrobin,type=DSR \
        -o servergroup=ftpgroup rule4
create-rule protocol=tcp,VIP=2003::1,port=ftp \
        -m lbalg=roundrobin,type=DSR \
        -o servergroup=ftpgroup6 rule3
create-rule -e protocol=tcp,VIP=2002::1,port=http \
        -m lbalg=roundrobin,type=DSR \
        -o serverrgroup=webgrp_v6 RULE-all
```

The following command exports rules to a file.

```
# ilbadm export-config /tmp/ilbrules
```

Following this command, `/tmp/ilbrules` contains the output displayed in the previous command.

The following command imports rules from a file.

EXAMPLE 6 Exporting and Importing Rules *(Continued)*

```
# ilbadm import-config /tmp/ilbrules
```

This command replaces whatever rules were in place with the contents of `/tmp/ilbrules`.

The following command imports rules from `stdin`.

```
# cat /tmp/ilbrules | ilbadm import-config
```

The effect of this command is identical to the effect of the preceding command.

EXAMPLE 7 Creating a Single Health Check

The following command creates a single health check.

```
# ilbadm create-healthcheck -h hc-timeout=3,hc-count=2,hc-interval=8,\
     hc-test=tcp hc1
```

EXAMPLE 8 Listing All Healthchecks

The following command lists all extant health checks.

```
# ilbadm show-healthcheck
HCNAME        TIMEOUT COUNT   INTERVAL DEF_PING TEST
hc1           2       1       10       Y        tcp
hc2           2       1       10       N        /usr/local/bin/probe
```

EXAMPLE 9 Deleting a Single Health Check

The following command deletes a single health check.

```
# ilbadm delete-healthcheck hc1
```

EXAMPLE 10 Displaying Statistics

The following command displays statistics at an interval of one seconds, for three iterations.

```
# ilbadm show-stats -A 1 3
 PKT_P   BYTES_P   PKT_U   BYTES_U   PKT_D   BYTES_D
0       0         0       0         4       196
0       0         0       0         4       196
0       0         0       0         4       196
```

The following is the command you would use to display statistics in verbose mode at intervals of one second. Output is too wide to fit within the page boundary.

```
# ilbadm show-stats -v 1
```

The following command displays statistics for rule `r1` at an interval of one second for three iterations.

EXAMPLE 10 Displaying Statistics *(Continued)*

```
# ilbadm show-stats -A -r r1 1 3
PKT_P   BYTES_P   PKT_U   BYTES_U   PKT_D   BYTES_D
0       0         0       0         4       196
0       0         0       0         4       196
0       0         0       0         4       196
```

The following command displays statistics for rule `r1` for each of its servers, for an interval of one second and a count of 3.

```
# ilbadm show-stats -A -r r1 -i 1 3
SERVERNAME          PKT_P   BYTES_P
_sg1.0              0       0
_sg1.1              0       0
_sg1.2              0       0
_sg1.0              0       0
_sg1.1              0       0
_sg1.2              0       0
_sg1.0              0       0
_sg1.1              0       0
_sg1.2              0       0
```

The following command displays itemized statistics, with timestamps, for server `_sg1.0`, at an interval of one second and a count of 3.

```
# ilbadm show-stats -A -s _sg1.0 -it 1 3
RULENAME            PKT_P   BYTES_P   TIME
r1                  0       0         2009-07-20:16.10.20
r1                  0       0         2009-07-20:16.10.21
r1                  0       0         2009-07-20:16.10.22
```

The following command displays statistics with specific option fields, at an interval of one second and a count of 3.

```
# ilbadm show-stats -o BYTES_D,TIME 1 3
BYTES_D   TIME
196       2009-07-20:16.14.25
0         2009-07-20:16.14.26
0         2009-07-20:16.14.27
```

EXAMPLE 11 Displaying Health Check Results

The following command displays the results of a health check.

```
# ilbadm show-hc-result rule1
RULENAME   HCNAME     SERVERID   STATUS   FAIL LAST     NEXT     RTT
rule1      hc1        _sg1.0     dead     6    04:45:17 04:45:30 698
rule1      hc1        _sg1.1     alive    0    04:45:11 04:45:25 260
rule1      hc1        _sg1.2     unreach  6    04:45:17 04:45:30 0
```

EXAMPLE 12 Displaying the NAT Table

The following command displays the NAT table.

```
# ilbadm show-nat 5
UDP: 124.106.235.150.53688>85.0.0.1.1024>>>82.0.0.39.4127>82.0.0.56.1024
UDP: 71.159.95.31.61528> 85.0.0.1.1024>>> 82.0.0.39.4146> 82.0.0.55.1024
UDP: 9.213.106.54.19787> 85.0.0.1.1024>>> 82.0.0.40.4114> 82.0.0.55.1024
UDP: 118.148.25.17.26676> 85.0.0.1.1024>>>82.0.0.40.4112> 82.0.0.56.1024
UDP: 69.219.132.153.56132>85.0.0.1.1024>>>82.0.0.39.4134> 82.0.0.55.1024
```

In actual `ilbadm` output, spaces are interspersed for greater readability.

EXAMPLE 13 Displaying the Persistence Table

The following command displays the persistence table.

```
# ilbadm show-persist 5
rule2: 124.106.235.150 --> 82.0.0.56
rule3: 71.159.95.31 --> 82.0.0.55
rule3: 9.213.106.54 --> 82.0.0.55
rule1: 118.148.25.17 --> 82.0.0.56
rule2: 69.219.132.153 --> 82.0.0.55
```

EXAMPLE 14 Displaying Server Groups

The following command displays basic information about server groups.

```
# ilbadm show-servergroup
sg1: id:sg1.2 35.0.0.4:80
sg1: id:sg1.1 35.0.0.3:80
sg1: id:sg1.0 35.0.0.2:80
sg2: id:sg2.3 35.0.0.5:81
sg2: id:sg2.2 35.0.0.4:81
sg2: id:sg2.1 35.0.0.3:81
sg2: id:sg2.0 35.0.0.2:81
```

The following command displays all available information about server groups.

```
# ilbadm show-servergroup -o all
sgname         serverID           minport maxport IP_address
sg1            _sg1.0             --      --      1.1.1.1
sg2            _sg2.1             --      --      1.1.1.6
sg3            _sg3.0             9001  9001      1.1.1.1
sg3            _sg3.1             9001  9001      1.1.1.2
sg3            _sg3.2             9001  9001      1.1.1.3
sg3            _sg3.3             9001  9001      1.1.1.4
sg3            _sg3.4             9001  9001      1.1.1.5
sg3            _sg3.5             9001  9001      1.1.1.6
sg3            _sg3.6             9001  9001      1.1.1.11
sg3            _sg3.7             9001  9001      1.1.1.12
sg3            _sg3.8             9001  9001      1.1.1.13
```

EXAMPLE 14 Displaying Server Groups *(Continued)*

```
sg3             _sg3.9              9001  9001      1.1.1.14
sg3             _sg3.10             9001  9001      1.1.1.15
sg3             _sg3.11             9001  9001      1.1.1.16
sg4             _sg4.0              9001  9006      1.1.1.1
sg4             _sg4.1              9001  9006      1.1.1.6
```

EXAMPLE 15 List Servers Associated with a Rule

The following command lists the servers that are associated with a rule.

```
# ilbadm show-server r1
SERVERID            ADDRESS         PORT RULENAME     STATUS SERVERGROUP
_sg1.0              35.0.0.10       80   rule1        E      sg1
_sg1.1              35.0.0.11       80   rule1        E      sg1
_sg1.2              35.0.0.12       80   rule1        D      sg1
```

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	service/network/load-balancer/ilb
Interface Stability	Committed

See Also

`ilbd(1M)`, `vmstat(1M)`, `attributes(5)`

Name ilbd – ILB daemon

Synopsis

```
/usr/lib/inet/ilbd
```

Description The Integrated Load Balancer (ILB) daemon, `ilbd`, provides user-level services for the ILB. It is a system daemon started by the Service Management Facility (see `smf(5)`). Its fault management resource identifier (FMRI) is:

```
svc:/network/loadbalancer/ilb:default
```

Note that `ilbd` is a Consolidation Private interface. See `attributes(5)`.

`ilbd` provides no administrative interface. All ILB administration should be done through `ilbadm(1M)` or the programming library `libilb(3LIB)`.

Options The `ilbd` daemon has no options.

Examples **EXAMPLE 1** Enabling the ILB Service

The following command enables the ILB service:

```
# svcadm enable svc:/network/loadbalancer/ilb:default
```

EXAMPLE 2 Disabling the ILB Service

The following command disables the ILB service:

```
# svcadm disable svc:/network/loadbalancer/ilb:default
```

Errors The `ilbd` daemon uses `syslog(3C)` to report status and error messages. All of the messages are logged with the `LOG_DAEMON` facility. Error messages are logged with the `LOG_ERR` and `LOG_NOTICE` priorities, and informational messages are logged with the `LOG_INFO` priority. The default entries in the `/etc/syslog.conf` file log all of the `ilbd` daemon error messages to the `/var/adm/messages` log.

Files

`/usr/lib/inet/ilbd`
: ILB daemon binary

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	service/network/load-balancer/ilb
Interface Stability	Project Private

See Also `svcs(1)`, `ilbadm(1M)`, `svcadm(1M)`, `syslog(3C)`, `libilb(3LIB)`, `syslog.conf(4)`, `attributes(5)`, `smf(5)`

Notes The `ilbd` service is managed by the service management facility, `smf(5)`, under the fault management resource identifier (FMRI):

```
svc:/network/loadbalancer/ilb:default
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. The service's status can be queried using the `svcs(1)` command. To administer the service, the authorization `solaris.smf.manage.ilb` must be granted. Alternatively, the rights profile “ILB Management” can be granted.

The ILB service must be enabled for the `ilbadm(1M)` command and any other `libilb(3LIB)` client applications to function properly.

Name ilomconfig – ILOM LAN configuration utility

Synopsis

```
ilomconfig list subcommand [options]
ilomconfig modify subcommand [options]
ilomconfig enable subcommand [options]
ilomconfig disable subcommand [options]
```

Description `ilomconfig` is an Oracle Integrated Lights Out Management (ILOM) configuration utility that allows an administrator to configure the LAN interface between the Service Processor (SP) and host. The administrator can view the LAN configuration settings by means of the `list` command and configure the interface using the `enable`, `disable`, and `modify` commands.

Sub-commands The following subcommands are available under the `list` command:

`system-summary`
List product summary information and ILOM version.

`interconnect`
List Host-to-ILOM interconnect settings.

The following subcommands are available under the `modify` command:

`interconnect`
Modifies Host-to-ILOM interconnect settings.

The following subcommand is available under the `enable` command:

`interconnect`
Enables the Host-to-ILOM interconnect.

The following subcommand is available under the `disable` command:

`interconnect`
Disables the Host-to-ILOM interconnect.

Options The following options are available for the `enable` and `modify` commands:

`-ipaddress=`*ipaddress*
Set the IP address for the SP side of the LAN interconnect.

`-hostipaddress=`*ipaddress*
Set the IP address for the host side of the LAN interconnect.

`-netmask=`*netmask*
Set the network mask for the IP addresses used for the LAN.

The following are general options:

`-?`, `-h`, `--help`
Display a brief usage message and then exit.

`-q`, `--quiet`
Suppress informational message output and return only error codes.

`-V`, `--version`
Display version information for `ilomconfig` and then exit.

Return Values `ilomconfig` can return any of the following values.

```
0 - Command successful
1 - Invalid option
3 - Invalid command
4 - The version of ILOM does not support this functionality
6 - Internal error
7 - Insufficient memory to execute command
9 - Insufficient privilege to execute command
10 - ILOM is not suipported on this system
50 - Cannot connect to ILOM
51 - An option must be specified
53 - Invalid IP Address
54 - An ILOM error occurred
55 - Cannot disable interconnect as it is already disabled
```

Files `/var/log/ilomconfig/ilomconfig.log`
`ilomconfig`-specific log file.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/management/ilomconfig
Interface Stability	Committed

See Also `attributes(5)`

Name imqadmin – launch the Message Queue administration console

Synopsis

```
/usr/bin/imqadmin [-javahome path]

/usr/bin/imqadmin -h

/usr/bin/imqadmin -v
```

Description `imqadmin` launches the graphical user interface application that performs most Message Queue administration tasks. These tasks include managing broker instances (including physical destinations) and administered objects.

Options The following options are supported:

`-h` Display usage help. The application is not launched.

`-javahome` *path* Specify a path to an alternate Java 2 compatible runtime.

`-v` Display version information.

Environment Variables The following environment variables affect the execution of this command:

`IMQ_JAVAHOME` Specify the Java 2 compatible runtime. When this environment variable is not set it defaults to `/usr/j2se`.

Exit Status The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

Files

`$HOME/.imq/admin/brokerlist.properties`
Contains user settings, a list of broker instances being managed.

`$HOME/.imq/admin/objectstorelist.properties`
Contains user settings, a list of object stores being managed.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWiq

See Also `imqbrokerd(1M)`, `imqcmd(1M)`, `imqdbmgr(1M)`, `imqkeytool(1M)`, `imqobjmgr(1M)`, `imqusermgr(1M)`, `attributes(5)`

Sun Java System Message Queue Administrator's Guide

Name imqbrokerd – start a Message Queue broker instance

Synopsis

```
/usr/bin/imqbrokerd [option...]

/usr/bin/imqbrokerd -h
```

Description `imqbrokerd` starts an instance of the Message Queue broker. The Message Queue broker is the main component of a Message Queue message server. The broker performs reliable delivery of messages to and from Java Message Service (JMS) clients.

`imqbrokerd` uses command line options to specify broker configuration properties.

Options The following options are supported:

`-backup` *fileName*
: Back up a Master Broker's configuration change record to *fileName*. This option only applies to broker clusters.

`-cluster` *brokerList*
: Specify the list of broker instances which are connected in a cluster. This list is merged with the list in the imq.cluster.brokerlist property. This option only applies to broker clusters.

  brokerList is a comma-separated list of broker instances, each specified by *hostName*:*port* (the host on which the broker instance is running and the port number it is using) If you don't specify a value for *hostName*, `localhost` is used. If you don't specify a value for *port*, the value of `7676` is used. For example: `host1:8899,host2,:7878`.

`-dbpassword` *password*
: Specify the password for a plugged-in JDBC-compliant database used as a Message Queue data store.

`-dbuser` *userName*
: Specify the user name for a plugged-in JDBC-compliant data store.

`-D`*property*`-=`*value*
: Set the specified broker configuration property to the *value*. The system does not validate either the configuration property or *value*. Therefore, spelling, formatting, and case is important. Message Queue can not set incorrect values passed using the `-D` option.

`-force`
: Perform action without user confirmation. This option only applies when you use the `-remove` *instance* option, which normally requires confirmation.

`-h`	Display usage help. Execute nothing else on the command line.
`-javahome` *path*	Specify the path to an alternate Java 2-compatible Java Development Kit (JDK) or Java Runtime Environment (JRE) The default is to use the runtime bundled with the operating system.
`-ldappassword` *password*	Specify the password for accessing an LDAP user repository when using an LDAP server (as opposed to a built-in flat-file repository) to authenticate users of a Message Queue message server.
`-license` [*name*]	Specify the license to load, if different from the default for your Message Queue product edition. If you don't specify a license name, this lists all licenses installed on the system. Depending on the installed Message Queue edition, the values for *name* are `pe` (Platform Edition-basic features), `try` (Platform Edition-90-day trial enterprise features), and `unl` (Enterprise Edition).
`-loglevel` *level*	Specify the logging level. Valid values for *level* are `NONE`, `ERROR`, `WARNING`, or `INFO`. The default value is `INFO`.
`-metrics` *int*	Report metrics at a specific interval. Specify *int* as the number of seconds.
`-name` *brokerName*	Specify the instance name of this broker and use the corresponding instance configuration file. If you do not specify a broker name, the name of the file is set to `imqbroker`. If you run more than one instance of a broker on the same host, each must have a unique name.
`-passfile` *filename*	Specify the name of the file from which to read the passwords for the SSL keystore, LDAP user repository, or JDBC-compliant database.
`-password` *keypassword*	Specify the password for the SSL certificate keystore.
`-port` *number*	Specify the broker's Port Mapper port number. By default, this is set to `7676`. To run two instances of a broker on the same server, each broker's Port

Mapper must have a different port number. JMS clients connect to the broker instance using this port number.

`-remove` *instance*

Remove the broker instance. Delete the instance configuration file, log files, data store, and other files and directories associated with the broker instance. This option requires user confirmation unless you also specify the `-force` option.

`-reset` store|messages|durables|props

Reset the data store (or a subset of the store) or resets the configuration properties of the broker instance when the broker instance is started. The action depends on the argument provided.

store — Clear all persistent data in the data store, including messages, durable subscriptions, and transaction information store.

messages — Clear all persistent messages durable.

durables — Clear all durable subscriptions.

props — Clear all configuration information in the `config.props` instance configuration file. All properties assume default values.

`-restore` *filename*

Replace the Master Broker's configuration change record with the specified backup file. This file must have been previously created using the `-backup` option. This option only applies to broker clusters.

`-shared`

Specify that the jms connection service be implemented using the shared threadpool model, in which threads are shared among connections to increase the number of connections supported by a broker instance.

`-silent`

Turn off logging to the console.

`-tty`

Display all messages be to the console. `WARNING` and `ERROR` level messages are displayed on the console by default.

`-upgrade-store-nobackup`	Specify that an earlier, incompatible version Message Queue data store is automatically removed when migrating to Message Queue 3.5 format. If you do not use this option, you must manually delete the earlier data store. This applies to both built-in (flat-file) persistence and plugged-in (JDBC-compliant) persistence. Migration of the earlier data store to a Message Queue 3.5 data store takes place the first time you start a Message Queue 3.5 broker instance on an earlier version data store.
`-version`	Display the version number of the installed product.
`-vmargs` *are* [[*arg*]…]	Specify arguments to pass to the Java VM. Separate arguments with spaces. If you want to pass more than one argument or if an argument contains a space, use enclosing quotation marks. For example: `imqbrokerd -tty -vmargs " -Xmx128m -Xincgc"`

Environment Variables

The following environment variables affect the execution of this command:

`IMQ_JAVAHOME` Specify the Java 2 compatible runtime. When this environment variable is not set it defaults to `/usr/j2se`.

Exit Status

The following exit values are returned:

`0` Successful completion.

>`0` An error occurred.

Files

`/etc/init.d/imq`
Shell script for starting `imqbrokerd`. This file looks at the `/etc/imq/imqbrokerd.conf` file.

`/etc/imq/imqbrokerd.conf`
Configuration file which controls the behavior of the broker startup script.

`/etc/imq/passwd`
Flat file user repository for authenticating users.

`/etc/imq/accesscontrol.properties`
Controls client access to broker functionality.

`/etc/imq/passfile.sample`
Sample passfile used by the `-passfile` option.

`/var/imq/instances/brokerName/props/config.properties`
Broker instance configuration file.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWiq

See Also `imqadmin(1M)`, `imqcmd(1M)`, `imqdbmgr(1M)`, `imqkeytool(1M)`, `imqobjmgr(1M)`, `imqusermgr(1M)`, `attributes(5)`

Sun Java System Message Queue Administrator's Guide

Name imqcmd – manage Message Queue brokers

Synopsis

```
/usr/bin/imqcmd subcommand argument [option...]

/usr/bin/imqcmd [-h | -H]

/usr/bin/imqcmd -v
```

Description `imqcmd` manages the Message Queue broker, including resources such as connection services, physical destinations, durable subscriptions, and transactions. The utility provides a number of subcommands for managing these resources.

`imqcmd` supports many subcommands. Basic connection and authentication is required for the execution of every `imqcmd` subcommand. Use the `-secure` option to specify secure connections. Subcommands and their corresponding arguments and options follow the `imqcmd` command on the command line. See `USAGE` and `OPTIONS`.

Options The following options are supported:

`-b` *hostName*`:`*port*
: Specify the name of the host on which the broker instance is running and the port number it is using.

  The default value is `localhost:7676`. If you do not specify the `-b` option, `imqcmd` uses the default.

  To specify `port` only, use: `-b :7878`. This is equivalent to `-b localhost:7878`

  To specify `name` only, use: `-b somehost`. This is equivalent to `-b somehost:7676`.

`-c` *clientID*
: Specify the ID of the durable subscriber to a topic.

`-d` *topicName*
: Specify the name of the topic.

  Use this option with the `list dur` and `destroy dur` subcommands.

`-f`
: Perform action without user confirmation.

  Use this option with any subcommand.

`-h`
: Display usage help. Execute nothing else on the command line.

`-H`
: Display usage help, attribute list, and examples. Execute nothing else on the command line.

`-int` *interval*
: Specify the interval, in seconds, at which the `metrics bkr`, `metrics dst`, and `metrics svc` subcommands display metrics output.

  Use this option with the `metrics` subcommand.

`-javahome`
: Specify an alternate Java 2 compatible runtime to use.

`-m` *metricType*
: Specify the type of metric information to diplay.

  Use this option with the `metrics bkr`, `metrics dst`, and `metrics svc` subcommands. The value of *metricType* depends on whether the metrics are generated for a destination, a service, or a broker.

  Use one of the following values to specify *metricType*:

  `ttl` Total of messages in and out of the broker (default)

  `rts` Provides the same information as `ttl`, but specifies the number of messages per second

  `cxn` Connections, virtual memory heap, threads

  The following command displays connection, VM heap, and threads metric information for the default broker instance (`localhost:7676`) every five seconds:

  ```
  imqcmd metrics bkr -m cxn -int 5
  ```

`-msp` *numSamples*
: Specify the number of samples the `metrics bkr`, `metrics dst`, and `metrics svc` subcommands display in the metrics output.

`-n` *argumentName*
: Specify the name of the subcommand argument. Depending on the subcommand, this might be the name of a service, a physical destination, a durable subscription, or a transaction ID.

`-o` *attribute=value*
: Specify the value of an attribute. Depending on the subcommand argument, this might be the attribute of a broker, service, or destination.

`-p` *password*
: Specify the administrator password.

  This option is deprecated. Use the `-passfile` option instead.

`-passfile`
: Specify the administrator password.

`-pst` *pauseType*
: Specify whether producers, consumers, or both are paused when pausing a destination.

  Use this option with the `pause dst` subcommand. Use one of the following values:

  CONSUMERS Pause delivery of messages to consumers.

  PRODUCERS Pause delivery of messages from producers.

  ALL Pause delivery of messages to consumers and from producers.

If the `-pst` option is not specified, pauses both consumers and producers (the equivalent of `-pst ALL`).

`-rtm` *timeout*
: Specify the timeout period in seconds of an `imqcmd` subcommand. The default value is `10`.

`-rtr` *numRetries*
: Specify the number of retries attempted after an `imqcmd` subcommand times out.

  The default value is `5`.

`-s`
: Silent mode. No output is displayed.

  Use this option with any subcommand.

`-secure`
: Specify a secure administration connection to the broker instance. You must first configure the broker to enable a secure connection service.

  Use this option whenever you want a secure communication with the broker.

`-svn` *serviceName*
: Specify the service for which the connections are listed.

  Use this option with the `list cxn` subcommand.

`-t` *destinationType*
: Specify the type of a destination: `t` (topic) or `q` (queue).

`-tmp`
: Include temporary destinations when listing destinations using the list `dst` subcommand.

`-u` *name*
: Specify the administrator user name.

  If you omit this value, you are prompted for it.

`-v`
: Display version information. Execute nothing else on the command line.

Usage

Subcommands and Options

The following subcommands and associated arguments and options are supported:

compact dst [`-t` *type* `-n` *destName*]
: Compact the flat-file data store for the destination of the specified type and name. If no type and name are specified, all destinations are compacted. Destinations must be paused before they can be compacted.

commit txn `-n` *transaction_id*
: Commit the specified transaction

create dst `-t` *destinationType* `-n` *destName* [`-o` *attribute=value*] [`-o` *attribute=value1*]...
Create a destination of the specified type, with the specified name, and the specified attributes. Destination names must contain only alphanumeric characters (no spaces) and can begin with an alphabetic character or the underscore character (_).

destroy dst `-t` *destinationType* `-n` *destName*
Destroy the destination of the specified type and name.

destroy dur `-n` *subscrName* `-c` *client_id*
Destroy the specified durable subscription for the specified Client Identifier.

list cxn [`-svn` *serviceName*] [`-b` *hostName:port*]
List all connections of the specified service name on the default broker or on a broker at the specified host and port. If the service name is not specified, all connections are listed.

list dst [`-tmp`]
List all destinations, with option of listing temporary destinations as well .

list dur `-d` *destination*
List all durable subscriptions for the specified destination.

list svc
List all connection services on the broker instance.

list txn
List all transactions, being tracked by the broker.

metrics bkr [`-m` *metricType*] [`-int` *interval*] [`-msp` *numSamples*]
Display broker metrics for the broker instance.

Use the `-m` option to specify the type of metric to display. Use one of the following values to specify *metricType*:

`ttl` Specifies the total of messages in and out of the broker (default) .

`rts` Provides the same information as `ttl`, but specifies the number of messages per second.

`cxn` Connections, virtual memory heap, threads.

Use the `-int` option to specify the interval (in seconds) at which to display the metrics. The default is 5 seconds.

Use the `-msp` option to specify the number of samples displayed in the output. A value of -1 means an unlimited number. The default value is -1.

metrics dst `-t` *type* `-n` *destName* [`-m` *metricType*] [`-int` *interval*] [`-msp` *numSamples*]
Displays metrics information for the destination of the specified type and name.

Use the `-m` option to specify the type of metrics to display. Use one of the following values to specify *metricType*:

`ttl` Specifies the number of messages flowing in and out of the broker and residing in memory.

`rts` Provides the same information as `ttl`, but specifies the number of messages per second.

`con` Displays consumer related metrics.

`dsk` Displays disk usage metrics.

Use the `-int` option to specify the interval (in seconds) at which to display the metrics. The default is 5 seconds.

Use the `-msp` option to specify the number of samples displayed in the output. A value of -1 means an unlimited number. The default value is 5.

metrics svc `-n` *serviceName* [`-m` *metricType*] [`-int` *interval*] [`-msp` *numSamples*]
List metrics for the specified service on the broker instance. Use the `-m` option to specify the type of metric to display. Use one of the following values to specify *metricType*:

`ttl` Total of messages in and out of the broker (default)

`rts` Provides the same information as `ttl`, but specifies the number of messages per second

`cxn` Connections, virtual memory heap, threads

Use the `-int` option to specify the interval (in seconds) at which to display the metrics. The default is 5 seconds.

Use the `-msp` option to specify the number of samples displayed in the output. A value of -1 means an unlimited number. The default value is -1.

pause bkr
Pause the broker instance.

pause dst [`-t` *type* `-n` *destName*] [`-pst` *pauseType*]
Pause the delivery of messages to consumers (`-pst CONSUMERS`), or from producers (`-pst PRODUCERS`), or both (`-pst ALL`), for the destination of the specified type and name. If no destination type or name are specified, all destinations are paused.

pause svc `-n` *serviceName*
Pause the specified service running on the broker instance. You cannot pause the administative service.

purge dst `-t` *destinationType* `-n` *destName*
Purge messages at the destination with the specified type and name.

purge dur `-n` *subscrName* `-c` *client_id*
Purge all messages for the specified client identifier.

query bkr
List the current settings of properties of the broker instance. Show the list of running brokers (in a multi-broker cluster) that are connected to the specified broker.

query dst `-t` *destinationType* `-n` *destName*
List information about the destination of the specified type and name.

query svc `-n` *serviceName*
Display information about the specified service running on the broker instance.

query txn `-n` *transaction_id*
List information about the specified transaction.

reload cls
Forces all the brokers in a cluster to reload the `imq.cluster.brokerlist` property and update cluster information. This subcommand only applies to broker clusters.

restart bkr
Shut down and restart the broker instance. This command restarts the broker using the options specified when the broker was first started. If you want different options to be in effect, you must shut down the broker and then start it again, specifying the options you want.

resume bkr
Resume the broker instance.

resume dst [`-t` *type*] [`-n -destName`]
Resumes the delivery of messages for the paused destination of the specified type and name. If no destination type and name are specified, all destinations are resumed.

resume svc `-n` *serviceName*
Resume the specified service running on the broker instance.

rollback txn `-n` *transaction_id*
Roll back the specified transaction.

shutdown bkr
Shut down the broker instance

update bkr `-o` *attribute=value* [`-o` *attribute=value*]...
Change the specified attributes for the broker instance.

update dst `-t` *destinationType* `-n` *destName* `-o` *attribute=value* [`-o` *attribute=value1*]...
Update the value of the specified attributes at the specified destination..

update svc `-n` *serviceName* `-o` *attribute=value* [`-o` *attribute=value1*]...
Update the specified attribute of the specified service running on the broker instance.

Attribute Value Pairs

You can specify attributes with the create and update subcommands. Applicable attributes depend on the subcommand arguments.

The following attributes are supported:

Queue (dst):

maxTotalMsgBytes	Value: Integer (maximum total size of messages, in bytes) Default: `0` (unlimited)
maxBytesPerMsg	Value: Integer (maximum size of a single message, in bytes) Default: `0` (unlimited)
maxNumMsgs	Value: Integer (maximum total number of messages) Default: `0` (unlimited)
consumerFlowLimit	Value: Integer Initial number of queued messages sent to active consumers before load-balancing starts A value of `-1` means an unlimited number. Default: `1000`
isLocalOnly	Value: Boolean (destination limited to delivering messages to local consumers only) Default: `false`
limitBehavior	Value: Specify how broker responds when memory-limit is reached. Use one of the following values: FLOW_CONTROL Slows down producers REMOVE_OLDEST Purges oldest messages REJECT_NEWEST Rejects the newest messages Default: REJECT_NEWEST
localDeliveryPreferred	Value: Boolean Specify messages be delivered to remote consumers only if there are no consumers on the local broker. Requires that the destination not be restricted to local-only delivery (`isLocalOnly = false`) Default: `false`
maxNumActiveConsumers	Value: Integer (maximum number of active consumers in load-balanced delivery) A value of `-1` means an unlimited number. Default: `1`
maxNumBackupConsumers	Value: Integer (maximum number of backup consumers in load-balanced delivery) A value of `-1` means an unlimited number. Default: `0`

maxNumProducers
: Value: (maximum total number of producers) A value of `-1` means an unlimited number.

  Default: `-1`

useDMQ
: Specify whether a destination's dead messages are discarded or put on the dead message queue.

  Default: `true`

Topic (dst):

consumerFlowLimit
: Value: Integer Maximum number of messages delivered to a consumer in a single batch. A value of `-1` means an unlimited number.

  Default: `1000`

isLocalOnly
: Value: Boolean (destination limited to delivering messages to local consumers only)

  Default: `false`

limitBehavior
: Value: Specify how broker responds when memory-limit is reached. Use one of the following values:

FLOW_CONTROL	Slows down producers
REMOVE_OLDEST	Purges the oldest messages
REJECT_NEWEST	Rejects the newest messages

  Default: REJECT_NEWEST

maxBytesPerMsg
: Value: Integer (maximum size of a single message, in bytes)

  Default: `0` (unlimited)

maxNumMsgs
: Value: Integer (maximum total number of messages) A value of `-1` means an unlimited number.

  Default: `-1`

maxNumProducers
: Value: (maximum total number of producers)

  Default: `0` (unlimited)

maxTotalMsgBytes
: Value: Integer (maximum total size of messages, in bytes) A value of `-1` means an unlimited number.

  Default: `-1`

useDMQ	Specify whether a destination's dead messages are discarded or put on the dead message queue. Default: `true`
Broker (bkr):	
imq.autocreate.destination.useDMQ	Value: Boolean. Set the `useDMQ` attribute to `true` to enable all autocreated physical destinations on a broker to use the dead message queue. Set the `useDMQ` attribute to `false` to disable all autocreated physical destinations on a broker from using the dead message queue. Default: `true`
imq.autocreate.queue	Value: Boolean Default: `true`
imq.autocreate.queue.maxNumActiveConsumers	Value: Integer (maximum number of consumers that can be active in load-balanced delivery from an autocreated queue destination) A value of `-1` means an unlimited number. Default: `1`
imq.autocreate.queue.maxNumBackupConsumers	Value: Integer (maximum number of backup consumers that can take the place of active consumers) A value of `-1` means an unlimited number. Default: `0`
imq.autocreate.topic	Value: Boolean Default: `true`
imq.cluster.url	Value: String (location of cluster configuration file) Default: none
imq.log.file.rolloverbytes	Value: Integer (maximum size of a log file, in bytes)

	Default: `0` (no rollover based on size)
imq.log.file.rolloversecs	Value: Integer (maximum age of a log file, in seconds)
	Default: `0` (no rollover based on age)
imq.log.level	Value: String (`NONE`, `ERROR`, `WARNING`, `INFO`)
	Default: `INFO`
imq.message.max_size	Value: Integer (maximum size of a single message, in bytes)
	Default: `70m`
imq.portmapper.port	Value: Integer
	Default: `7676`
imq.system.max_count	Value: Integer (maximum total number of messages)
	Default: `0` (no limit)
imq.system.max_size	Value: Integer (maximum total size of messages, in bytes)
	Default: `0` (no limit)

Service (svc):

maxThreads	Value: Integer (maximum threads assigned)
	Default: Depends on service
minThreads	Value: Integer (minimum threads assigned)
	Default: Depends on service
port	Value: Integer
	Default: `0` (dynamically allocated)

Examples

EXAMPLE 1 Shutting Down a Broker

The following command shuts down a broker for hostname `myserver` on port `7676`:

```
mqcmd shutdown bkr -b myserver:7676
```

EXAMPLE 2 Restarting a Broker

The following command restarts a broker for hostname `myserver`:

```
imqcmd restart bkr -b myserver
```

EXAMPLE 3 Pausing a Service

The following command pauses a broker for hostname `localhost` on port `7676`, with a *serviceName* of `jms`:

```
imqcmd pause svc -n jms -b :7676
```

EXAMPLE 4 Resuming a Service

The following command resumes a service for hostname `localhost` on port `7676`, with a *serviceName* of `jms`:

```
imqcmd resume svc -n jms -b myserver:7676
```

EXAMPLE 5 Creating a Queue Destination

The following command creates a queue destination for hostname `myserver` on port `7676`, with a *destName* of `myFQ`, a *queueDeliveryPolicy* of `Failover`, and a *maxBytesPerMsg* of 10000:

```
imqcmd create dst -n myFQ -t q -o "queueDeliveryPolicy=f" \
     -o "maxBytesPerMsg=10000" -b myserver:7676
```

EXAMPLE 6 Purging a Queue Destination

The following command purges a queue destination for hostname `myserver` on port `7676`, with a *destName* of `myFQ`:

```
imqcmd purge dst -n myFQ -t q -b myserver:7676
```

EXAMPLE 7 Listing Destinations on a Broker

The following command lists destinations for hostname `myserver` on port `7676`:

```
imqcmd list dst -b myserver:7676
```

EXAMPLE 8 Updating a Portmapper Port

The following command updates a portmapper port on hostname `myserver` from port `7676` to `7878`:

```
imqcmd update bkr -o "imq.portmapper.port=7878"
```

EXAMPLE 9 Updating the Maximum Number of Messages in the Queue

The following command updates the maximum number of messages in the queue to `2000` for `myserver` on port `8080` with a *destName* of `TestQueue`:

```
imqcmd update dst -b myserver:8080 -n TestQueue -t q -o "maxNumMsgs=2000"
```

EXAMPLE 10 Updating the Maximum Threads

The following command updates the maximum threads `jms` connection service to `200` for hostname `localhost` on port `7676`:

```
imqcmd update svc -n jms -o "minThreads=200"
```

EXAMPLE 11 Listing Durable Subscriptions

The following command lists durable subscriptions for a topic with hostname `localhost` on port `7676` with a *destName* of `myTopic`:

```
imqcmd list dur -d myTopic
```

EXAMPLE 12 Destroying Durable Subscriptions

The following command destroys subscriptions for hostname `localhost` on port `7676` with a *dursubName* of `myDurSub` and a *client_ID* of `111.222.333.444`:

```
imqcmd destroy dur -n myDurSub -c "111.222.333.444"
```

EXAMPLE 13 Listing All Transactions

The following command lists all transactions on a broker with hostname `localhost` on port `7676`:

```
imqcmd list txn
```

EXAMPLE 14 Displaying Information About a Transaction

The following command displays information about a transaction with hostname `localhost` on port `7676`, and a *transactionID* of `1234567890`

```
imqcmd query txn -n 1234567890
```

EXAMPLE 15 Committing a Transaction

The following command commits a transaction with hostname `localhost` on port `7676`, and a *transactionID* of `1234567890`:

```
imqcmd commit txn -n 1234567890
```

Environment Variables

The following environment variables affect the execution of this command:

`IMQ_JAVAHOME` Specify the Java 2 compatible runtime. When this environment variable is not set it defaults to `/usr/j2se`.

Exit Status

The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWiq

See Also imqadmin(1M), imqbrokerd(1M), imqdbmgr(1M), imqkeytool(1M), imqobjmgr(1M), imqusermgr(1M), attributes(5)

Sun Java System Message Queue Administrator's Guide

Name imqdbmgr – manage a plugged-in JDBC-compliant Message Queue data store

Synopsis

```
/usr/bin/imqdbmgr subcommand argument [ [option...]]

/usr/bin/imqdbmgr -h | -help

/usr/bin/imqdbmgr -v | -version
```

Description The `imqdbmgr` utility creates and manages a Java DataBase Connectivity (JDBC) compliant database used for Message Queue persistent storage.

The database can be either embedded or external. To use a JDBC-compliant database (and the `imdbmgr` utility), you need to first set a number of JDBC-related properties in the broker instance configuration file. See the *Sun Java System Message Queue Administrator's Guide* for additional information.

`imqdbmgr` supports four management subcommands. These *subcommand*s, and their corresponding *argument*s and *option*s follow the `imqdbmgr` command on the command line. See `USAGE` and `OPTIONS`.

The following subcommands are supported:

create	Create a Message Queue database schema.
delete	Delete Message Queue database tables in the current data store.
recreate	Delete Message Queue database tables and recreate Message Queue database schema in the curent data store.
reset	Reset the database table lock to allow other processes to access database tables.

The `imqdbmgr` subcommands support the following arguments:

all	Indicates the subcommand applies to the data store, as well as the database tables.
lck	Indicates the subcommand applies to the database table lock.
oldtbl	Indicates the subcommand applies to an older version of the database tables.
tbl	Indicates the subcommand applies to the database tables only.

Options The following options are supported:

`-b` *brokerName*	Specify the broker instance name and corresponding instance configuration properties. If this option is not specified, the default broker instance is assumed. Use this option with the `create`, `delete`, `recreate`, or `reset` subcommands.
`-D` *property=value*	Set system property *property* to *value*.

Use this option with the `create`, `delete`, `recreate`, or `reset` subcommands.

`-h` | `-help` Display usage help. Execute nothing else on the command line.

`-p` *password* Specify the database password.

Use this option with the `create`, `delete`, `recreate`, or `reset` subcommands.

`-u` *userName* Specify the database user name.

Use this option with the `create`, `delete`, `recreate`, or `reset` subcommands.

`-v` | `-version` Display version information. Execute nothing else on the command line.

Usage The following subcommands and associated arguments are supported:

create all — Create a new embedded data store and Message Queue database schema for a specified or default broker instance.

create tbl [`-u` *userName*] [`-p` *password*] — Create Message Queue database schema in an existing data store for a specified or default broker instance.

delete tbl [`-u` *userName*] [`-p` *password*] — Delete Message Queue database tables in the current data store for a specified or default broker instance.

delete oldtbl [`-u` *userName*] [`-p` *password*] — Delete the earlier version of Message Queue database tables. Used after the data store has been automatically migrated to the current version of Message Queue.

recreate tbl [`-u` *userName*] [`-p` *password*] — Delete Message Queue database tables and recreate Message Queue database schema in the current data store for a specified or default broker instance.

reset lck — Reset the database table lock to allow other processes to access database tables.

Environment Variables The following environment variables affect the execution of this command:

`IMQ_JAVAHOME` Specify the Java 2 compatible runtime. When this environment variable is not set it defaults to `/usr/j2se`.

Exit Status The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

Files

`/var/imq/instances/`*brokerName*`/dbstore`	Recommended directory in which to create an embedded database.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWiq

See Also `imqadmin(1M)`, `imqbrokerd(1M)`, `imqcmd(1M)`, `imqusermgr(1M)`, `imqkeytool(1M)`, `imqobjmgr(1M)`, `attributes(5)`

Sun Java System Message Queue Administrator's Guide

Name imqkeytool – generate a self-signed certificate for secure communication

Synopsis

```
/usr/bin/imqkeytool [-broker] [-servlet keystore_location]

/usr/bin/imqkeytool -h
```

Description The `imqkeytool` utility generates a self-signed certificate for secure communication. The certificate can be used by a broker instance to establish a secure connection with a client, or by a Message Queue-supplied HTTPS servlet to establish a secure connection with a broker instance. An HTTPS servlet is an SSL-enabled varient of the HyperText Transfer Protocol that establishes a secure connection with a broker instance.

Without an option, `imqkeytool` generates a self-signed certificate for a broker instance.

`imqkeytool` uses command line options to specify whether the certificate is used by a broker instance or by a servlet.

Options The following options are supported:

`-broker` — Generate a self-signed certificate for the broker and places it in the Message Queue keystore. All broker instances running on a system must use the same certificate.

`-h` — Display usage help. Do not execute anything else on the command line.

`-servlet` *keystore_location* — Generate a self-signed certificate for an HTTPS servlet and places it in *keystore_location*.

keystore_location refers to the location of the keystore. You should move this keystore to a location where it is accessible and readable by the Message Queue HTTPS servlet to establish a secure connection with a broker.

Environment Variables The following environment variables affect the execution of this command:

`IMQ_JAVAHOME` — Specify the Java 2 compatible runtime. When this environment variable is not set it defaults to `/usr/j2se`.

Exit Status The following exit values are returned:

`0` — Successful completion.

`>0` — An error occurred.

Files

`/etc/imq/keystore` — Contains Message Queue keystore in which `imqkeytool` stores a self-signed certificate for brokers.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWiq

See Also imqadmin(1M), imqbrokerd(1M), imqcmd(1M), imqdbmgr(1M), imqobjmgr(1M), imqusermgr(1M), attributes(5)

Sun Java System Message Queue Administrator's Guide

Name imqobjmgr – manage Message Queue administered objects

Synopsis

```
/usr/bin/imqobjmgr subcommand [[option]...]

/usr/bin/imqobjmgr -i fileName

/usr/bin/imqobjmgr -h | [-H] | -help | -Help

/usr/bin/imqobjmgr -v
```

Description `imqobjmgr` manages Message Queue administered objects in an object store accessible using JNDI. Administered objects allow JMS clients to be provider-independent by insulating them from provider-specific naming and configuration formats.

`imqobjmgr` supports five management subcommands. These *subcommand*s, and their corresponding *option*s follow the `imqobjmgr` command on the command line. See `USAGE` and `OPTIONS`.

The following subcommands are supported:

add	Add a new administered object
delete	Delete an administered object
list	Display a list of administered objects
query	Display information about administered objects
update	Update administered objects

You can use the `-i` option to specify the name of an input file that uses java property file syntax to represent all or part of any `imqobjmgr` subcommand clause. The `-f`, `-s`, and `-pre` options can be used with any `imqobjmgr` subcommand.

Options The following options are supported:

`-f`	Perform action without user confirmation.
`-h` | `-help`	Display usage help. Execute nothing else on the command line.
`-H` | `-Help`	Display usage help, attribute list, and examples. Execute nothing else on the command line.
`-i` *fileName*	Specify the name of an input file containing all or part of the subcommand clause, specifying object type, lookup name, object attributes, object store attributes, or other options. Use this option for repetitive information, such as object store attributes.
`-j` *attribute=value*	Specify attributes necessary to identify and access a JNDI object store.
`-javahome`	Specify an alternate Java 2 compatible runtime to use. `imqobjmgr` uses the runtime bundled with the operating system by default.

`-l` *lookupName*	Specify the JNDI lookup name of an administered object. This name must be unique in the object store's context.
`-o` *attribute=value*	Specify the attributes of an administered object.
`-pre`	Run command in preview mode. Preview mode indicates what will be done without performing the command.
`-r` *read-only_state*	Specify if an administered object is a read-only object. A value of true indicates the administered object is a read-only object. JMS clients cannot modify the attributes of read-only administered objects. The read-only state is set to false by default.
`-s`	Silent mode. No output is displayed.

`-t` *type*	Specify the type of an administered object:
	`q = queue`
	`t = topic`
	`cf = ConnectionFactory`
	`qf = queueConnectionFactory`
	`tf = topicConnectionFactory`
	`xcf = XA ConnectionFactory (distributed transactions)`
	`xqf = XA queueConnectionFactory (distributed transactions)`
	`xtf = XA topicConnectionFactory (distributed transactions)`
	`e = SOAP endpoint (used to support SOAP messaging)`

`-v`	Display version information. Execute nothing else on the command line.

Usage

This section provides information on subcommands, options, and attribute value pairs.

Subcommands and Options

The following subcommands and corresponding options are supported:

add `-t` *type* `-l` *lookupName* [`-o` *attribute=value*]... `- j` *attribute=value...*
: Add a new administered object of the specified type, lookup name , and object attributes to an object store.

delete `-t` *type* `-l` *lookupName* `-j` *attribute=value...*
: Delete an administered object, of the specified type and lookup name from an object store.

list [`-t` *type*] `-j` *attribute=value...*
: Display a list of administered objects of a specified type, or all administered objects, in an object store.

query `-l` *lookupName* `-j` *attribute=value...*
: Display information about an administered object of a specified lookup name in an object store.

update `-l` *lookupName* [`-o` *attribute=value*]... `-j` *attribute=value*...
Update the specified attribute values of an administered object of the specified lookup name in an object store.

Attribute Value Pairs

The following attribute value pairs are supported for the specified administered object types:

Type = ConnectionFactories: ConnectionFactory, TopicConnectionFactory, QueueConnectionFactory, XAConnectionFactory, XATopicConnectionFactory, and XAQueueConnectionFactory

imqAckOnAcknowledge
Value: String (`true`, `false`, not specified)

Default: not specified

imqAckOnProduce
Value: String (`true`, `false`, not specified)

Default: not specified

imqAckTimeout
Value: String (time in milliseconds)

Default: `0` (no timeout)

imqAddressList
Value: String

Default: not specified

imqAddressListBehavior
Value: String

Default: Priority

imqAddressListIterations
Value: Integer

Default: 1

imqBrokerHostName
Used if imqConnectionType is `TCP` or `TLS`. This attribute type is only supported in Message Queue 3.0.

Value: String

Default:`localhost`

imqBrokerHostPort
Used if imqConnectionType is `TCP` or `TLS`. This attribute type is only supported in Message Queue 3.0.

Value: Integer

Default: `7676`

imqBrokerServicePort
Used if imqConnectionType is `TCP` or `TLS`. This attribute type is only supported in Message Queue 3.0.

Value: Integer

Default: `0`

imqConfiguredClientID
Value: String (ID number)

Default: no ID specified

imqConnectionFlowCount
Value: Integer

Default: 100

imqConnectionFlowLimit
Value: Integer

Default: 1000

imqConnectionFlowLimitEnabled
Value: Boolean

Default: `false`

imqConnectionType
This attribute type is only supported in Message Queue 3.0.

Value: String (`TCP`, `TLS`, `HTTP`).

Default: `TCP`

imqConnectionURL
Used if imqConnectionType is `HTTP`. This attribute type is only supported in Message Queue 3.0.

Value: String

Default: `http://`*localhost*`/imq/tunnel`

imqConsumerFlowLimit
Value: Integer

Default: 1000

imqConsumerFlowThreshold
Value: Integer

Default: 50

imqDefaultPassword
Value: String

Default: `guest`

imqDefaultUsername
Value: String

Default: `guest`

imqDisableSetClientID
Value: Boolean

Default: `false`

imqJMSDeliveryMode
Value: Integer (`1`=non-persistent, `2`=persistent)

Default: `2`

imqJMSExpiration
Value: Long (time in milliseconds)

Default: `0` (does not expire)

imqJMSPriority
Value: Integer (`0` to `9`)

Default: `4`

imqLoadMaxToServerSession
Value: Boolean

Default: `true`

imqOverrideJMSDeliveryMode
Value: Boolean

Default: `false`

imqOverrideJMSExpiration
Value: Boolean

Default: `false`

imqOverrideJMSHeadersToTemporaryDestinations
Value: Boolean

Default: `false`

imqOverrideJMSPriority
Value: Boolean

Default: `false`

imqQueueBrowserMaxMessagesPerRetrieve
Value: Integer

Default: `1000`

imqBrowserRetrieveTimeout
Value: Long (time in milliseconds)

Default: `60,000`

imqReconnectAttempts
Value: Integer

Default: `0`

imqReconnectEnabled
Value: Boolean

Default: `false`

imqReconnectInterval
Value: Long (time in milliseconds)

Default: `3000`

imqSetJMSXAppID
Value: Boolean

Default: `false`

imqSetJMSXConsumerTXID
Value: Boolean

Default: `false`

imqSetJMSXProducerTXID
Value: Boolean

Default: `false`

imqSetJMSXRcvTimestamp
Value: Boolean

Default: `false`

imqSetJMSXUserID
Value: Boolean

Default: `false`

imqSSLIsHostTrusted
Used if imqConnectionType is `TLS`. This attribute type is only supported in Message Queue 3.0.

Value: Boolean

Default: `true`

Type = Destinations: Topic and Queue

imqDestinationDescription — Value: String

Default: no description

imqDestinationName — Value: String

Default: `Untitled_Destination_Object`

Type = Endpoint (SOAP Endpoint)

imqEndpointDescription — Value: String

Default: `A description for the endpoint object`

imqEndpointName — Value: String

Default: `Untitled_Endpoint_Object`

imqSOAPEndpointList — Value: String (one or more space-separated URLs)

Default: no url

Examples

EXAMPLE 1 Adding a Topic Administered Object to an Object Store

Where JNDI lookup name=`myTopic` and `imqDestinationName=MyTestTopic`, the following command adds to an LDAP server object store:

```
imqobjmgr add -t t -l "cn=myTopic"\
-o "imqDestinationName=MyTestTopic"\
-j "java.naming.factory.initial=com.sun.jndi.ldap.LdapCtxFactory"\
-j "java.naming.provider.url=ldap://mydomain.com:389/o=imq"
```

Where JNDI lookup name=`myTopic` and `imqDestinationName=MyTestTopic`, the following command adds to a file system object store:

```
imqobjmgr add -t -l "cn=myTopic"\
-o "imqDestinationName=MyTestTopic"\
-j \
  "java.naming.factory.initial=com.sun.jndi.fscontext.RefFSContextFactory"\
-j "java.naming.provider.url=file:/home/foo/imq_admin_objects"
```

EXAMPLE 1 Adding a Topic Administered Object to an Object Store *(Continued)*

Where JNDI lookup name=`myTopic` and `imqDestinationName=MyTestTopic`, the following command adds to a file system object store, using an input file:

```
imqobjmgr -i inputfile
```

The associated input file consists of the following:

```
cmdtype=add
obj.type=t
obj.lookupName=cn=myTopic
obj.attrs.imqDestinationName=MyTestTopic
objstore.attrs.java.naming.factory.initial=com.sun.jndi.fscontext.\
  RefFSContextFactory
objstore.attrs.java.naming.provider.url=file:/home/foo/imq_admin_objects
```

EXAMPLE 2 Adding a QueueConnectionFactory Administered Object to an Object Store

Where JNDI lookup name=`myQCF`, `read-only state=true`, `imqAddressList=mq://foohost:777/jms`, the following command adds to an LDAP server object store:

```
imqobjmgr add -t qf -l "cn=myQCF" -r true\

-o "imqAddressList=mq://foohost:777/jms"\
-j "java.naming.factory.initial=com.sun.jndi.ldap.LdapCtxFactory"\
-j "java.naming.provider.url=ldap://mydomain.com:389/o=imq"\
```

Where JNDI lookup name=`myQCF`, `read-only state=true`, `imqAddressList=mq://foohost:777/jms`, the following command adds to an LDAP server object store using an input file:

```
imqobjmgr -i inputfile
```

The associated input file consists of the following:

```
cmdtype=add
obj.type=qf
obj.lookupName=cn=myQCF
obj.readOnly=true
obj.attrs.imqAddressList=mq://foohost:777/jms
objstore.attrs.java.naming.factory.initial=com.sun.jndi.\
  ldap.LdapCtxFactory
objstore.attrs.java.naming.provider.url=ldap://mydomain.com:389/o=imq
```

Where JNDI lookup name=`myQCF`, read-only state=`true`, `imqAddressList=mq://foohost:777/jms`, the following command adds to an LDAP server object store, using both an input file and command options:

EXAMPLE 2 Adding a QueueConnectionFactory Administered Object to an Object Store *(Continued)*

```
imqobjmgr add -t qf -l "cn=myQCF"\
-o "imqAddressList=mq://foohost:777/jms"\
-i inputfile
```

The associated input file consists of the following:

```
objstore.attrs.java.naming.factory.initial=com.sun.jndi.ldap.LdapCtxFactory
objstore.attrs.java.naming.provider.url=ldap://mydomain.com:389/o=imq
```

EXAMPLE 3 Deleting a Topic Administered Object from an Object Store

Where JNDI lookup name=`myTopic` and no confirmation is requested, the following command deletes from an LDAP server object store:

```
imqobjmgr delete -f -l "cn=myTopic"\
-j "java.naming.factory.initial=com.sun.jndi.ldap.LdapCtxFactory"\
-j "java.naming.provider.url=ldap://mydomain.com:389/o=imq"
```

EXAMPLE 4 Querying Information About a Topic Administered Object

Where JNDI lookup name=`myTopic`, the following command queries from an LDAP server object store using simple authentication scheme:

```
imqobjmgr query -l "cn=myTopic"\
-j "java.naming.factory.initial=com.sun.jndi.ldap.LdapCtxFactory"\
-j "java.naming.provider.url=ldap://mydomain.com:389/o=imq"\
-j "java.naming.security.authentication=simple"\
-j "java.naming.security.principal=uid=foo,ou=imqobjmgr,o=imq"\
-j "java.naming.security.credentials=foo"
```

Exit Status

The following exit values are returned:

`0` Successful completion.

>`0` An error occurred.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWiq

See Also

`imqadmin(1M)`, `imqcmd(1M)`, `imqbrokerd(1M)`, `imqkeytool(1M)`, `imqusermgr(1M)`, `attributes(5)`

Sun Java System Message Queue Administrator's Guide

Name imqusermgr – command utility for managing a Message Queue user repository

Synopsis

```
/usr/bin/imqusermgr subcommand [[option]...]

/usr/bin/imqusermgr -h

/usr/bin/imqusermgr -v
```

Description The `imqusermgr` utility manages a file-based user repository to authenticate and authorize users of a Message Queue message server.

`imqusermgr` provides subcommands for adding, deleting, updating, and listing user entries in the repository.

`imqusermgr` supports four management subcommands. These *subcommand*s, and their corresponding *option*s follow the `imqusermgr` command on the command line. See `USAGE` and `OPTIONS`.

The following subcommands are supported:

add	Add a new user and associated password to the repository.
delete	Delete a user from the repository.
list	Display information users in the repository.
update	Update the password or state of a user in the repository.

Options The following options are supported:

`-a` *active_state*	Specify if user's state is active or inactive. An inactive user cannot create connections to the Message Queue message server. Valid values for *active_state* are `true` or `false`. Specify `true` for active or `false` for inactive. the default is `true`. Use this option with the `update` subcommmand.
`-f`	Perform action without user confirmation. Use this option with the `delete` and `update` subcommmands.
`-g` *group*	Specify the group of the user. Valid values for group are `admin`, `user`, and `anonymous`. Use this option with the `add` subcommmand.
`-h`	Display usage help. Exceute nothing else on the command line.
`-i` *brokerName*	Specify the broker instance user repository to which the command applied. If you do not specify *brokerName*, the default *brokerName* is assumed.

Use this option with the `add`, `delete`, `list`, and `update` subcommmands.

`-p` *password* Specify user password.

Use this option with the `add` and `update` subcommmands.

`-s` Silent mode. Display no output

Use this option with the `add`, `delete`, and `update` subcommands.

`-u` *userName* Specify user name.

userName cannot contain the following characters: asterisk (`*`), colon (`:`), NEWLINE, or RETURN.

Use this option with the `add`, `delete`, `update` and `list` subcommands.

`-v` Display version information. Exceute nothing else on the command line.

Usage The following subcommands and corresponding options are supported:

add `-u` *userName* `-p` *password* [`-g` *group*] [`-s`] [`-i` *brokerName*]
Add a new user and associated password to the repository, and optionally specify the user's group.

delete `-u` *userName* [`-s`] [`-f`] [`-i` *brokerName*]
Delete a user from the repository.

list [`-u` *user_name*] [`-i` *brokerName*]
Display information about the specified user in the repository. If no user is specified, all users are displayed.

update `-u` *userName* `-p` *password* [`-a` *state*] [`-s`] [`-f`] [`-i` *brokerName*]
update `-u` *userName* `-a` *state* [`-p` *password*] [`-s`] [`-f`] [`-i` *brokerName*]
Update the password or state (or both) of a user.

Environment Variables The following environment variables affect the execution of this command:

`IMQ_JAVAHOME` Specify the Java 2 compatible runtime. When this environment variable is not set, it defaults to `/usr/j2se`.

Exit Status The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

Files `/etc/imq/passwd` Flat-file user repository.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWiq

See Also imqadmin(1M), imqbrokerd(1M), imqcmd(1M), imqdbmgr(1M), imqkeytool(1M), imqobjmgr(1M), attributes(5)

Sun Java System Message Queue Administrator's Guide

Name in.chargend – UDP or TCP character generator service daemon

Synopsis

```
in.chargend

FMRI svc:/internet/chargen:default
```

Description FMRI stands for Fault Management Resource Identifier. It is used to identify resources managed by the Fault Manager. See `fmd(1M)` and `smf(5)`.

The `in.chargend` service provides the server-side of the character-generator protocol. This protocol is used for debugging and bandwidth measurement and is available on both TCP and UDP transports, through port 19.

The in.chargend service is an `inetd(1M)` `smf(5)` delegated service. The `in.chargend` detects which transport is requested by examining the socket it is passed by the `inetd` daemon.

TCP-based service — Once a connection is established, the `in.chargend` generates a stream of data. Any data received is discarded. The server generates data until the client program terminates the connection. Note that the data flow is limited by TCP flow control mechanisms.

UDP-based service — The `in.chargend` listens for UDP datagrams. When a datagram is received, the server generates a UDP datagram in response containing a random number of ASCII characters (ranging from 0 to 512 characters). Any received data is ignored.

The `in.chargend` data consists of a pattern of 72 character lines containing the printable, 7–bit ASCII characters. Each line is terminated with a carriage return and a line feed character.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	service/network/legacy-network-services
Interface Stability	Committed

See Also `inetd(1M)`, `attributes(5)`, `smf(5)`

RFC 864

Name in.comsat, comsat – biff server

Synopsis

```
/usr/sbin/in.comsat
```

Description `comsat` is the server process which listens for reports of incoming mail and notifies users who have requested to be told when mail arrives. It is invoked as needed by `inetd(1M)`, and times out if inactive for a few minutes.

`comsat` listens on a datagram port associated with the `biff` service specification (see `services(4)`) for one line messages of the form

user@*mailbox*-*offset*

If the *user* specified is logged in to the system and the associated terminal has the owner execute bit turned on (by a `biff y`), the *offset* is used as a seek offset into the appropriate mailbox file, and the first 7 lines or 560 characters of the message are printed on the user's terminal. Lines which appear to be part of the message header other than the `From`, `To`, `Date`, or `Subject` lines are not printed when displaying the message.

Files `/var/adm/utmpx` user access and administration information

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	service/network/network-servers

See Also `svcs(1)`, `inetadm(1M)`, `inetd(1M)`, `svcadm(1M)`,`services(4)`, `attributes(5)`, `smf(5)`

Notes The message header filtering is prone to error.

The `in.comsat` service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/network/comsat:default
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. Responsibility for initiating and restarting this service is delegated to `inetd(1M)`. Use `inetadm(1M)` to make configuration changes and to view configuration information for this service. The service's status can be queried using the `svcs(1)` command.

Name in.daytimed – UDP or TCP daytime protocol service daemon

Synopsis

```
in.daytimed

FMRI svc:/internet/daytime:default
```

Description FMRI stands for Fault Management Resource Identifier. It is used to identify resources managed by the Fault Manager. See `fmd(1M)` and `smf(5)`.

The `in.daytimed` service provides the server-side of the daytime protocol. This protocol is used for debugging and bandwidth measurement and is available on both TCP and UDP transports, through port 13.

The `in.daytimed` service is an `inetd(1M)` `smf(5)` delegated service. The `in.daytimed` detects which transport is requested by examining the socket it is passed by the `inetd` daemon.

TCP-based service — Once a connection is established, the `in.daytimed` generates the current date and time in `ctime(3C)` format as 7-bit ASCII and sends it through the connection. The server then closes the connection. Any data received from the client side of the connection is discarded.

UDP-based service — The `in.daytimed` listens for UDP datagrams. When a datagram is received, the server generates the current date and time in `ctime(3C)` format as 7-bit ASCII and inserts it in a UDP datagram sent in response to the client's request. Any received data is ignored.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	service/network/legacy-network-services
Interface Stability	Committed

See Also `inetd(1M)`, `attributes(5)`, `smf(5)`

RFC 867

Name in.dhcpd – Dynamic Host Configuration Protocol server

Synopsis

```
/usr/lib/inet/in.dhcpd  [-denv] [-h relay_hops] [-i interface, ...]
 [-l syslog_local_facility] [-b automatic | manual]
 [-o DHCP_offer_time] [-t dhcptab_rescan_interval]

/usr/lib/inet/in.dhcpd  [-dv] [-h relay_hops] [-i interface,]...
 [-l syslog_local_facility] -r IP_address | hostname, ...
```

Description `in.dhcpd` is a daemon that responds to Dynamic Host Configuration Protocol (DHCP) requests and optionally to BOOTP protocol requests. The daemon forks a copy of itself that runs as a background process. It must be run as root. The daemon has two run modes, DHCP server (with optional BOOTP compatibility mode) and BOOTP relay agent mode.

The first line in the `SYNOPSIS` section illustrates the options available in the DHCP/BOOTP server mode. The second line in the SYNOPSIS section illustrates the options available when the daemon is run in BOOTP relay agent mode.

The DHCP and BOOTP protocols are used to provide configuration parameters to Internet hosts. Client machines are allocated their IP addresses as well as other host configuration parameters through this mechanism.

The DHCP/BOOTP daemon manages two types of DHCP data tables: the `dhcptab` configuration table and the DHCP network tables.

See `dhcptab(4)` regarding the dhcptab configuration table and `dhcp_network(4)` regarding the DHCP network tables.

The `dhcptab` contains macro definitions defined using a `termcap`-like syntax which permits network administrators to define groups of DHCP configuration parameters to be returned to clients. However, a DHCP/BOOTP server always returns hostname, network broadcast address, network subnet mask, and IP maximum transfer unit (MTU) if requested by a client attached to the same network as the server machine. If those options have not been explicitly configured in the `dhcptab`, `in.dhcpd` returns reasonable default values.

The `dhcptab` is read at startup, upon receipt of a `SIGHUP` signal, or periodically as specified by the `-t` option. A `SIGHUP` (sent using the command `svcadm refresh network/dhcp-server`) causes the DHCP/BOOTP daemon to reread the `dhcptab` within an interval from `0-60` seconds (depending on where the DHCP daemon is in its polling cycle). For busy servers, users should run `svcadm restart network/dhcp-server` to force the `dhcptab` to be reread.

The DHCP network tables contain mappings of client identifiers to IP addresses. These tables are named after the network they support and the datastore used to maintain them.

The DHCP network tables are consulted during runtime. A client request received from a network for which no DHCP network table exists is ignored.

This daemon is obsolete and is subject to removal in a future release of Oracle Solaris. Scripts, programs, or procedures that use this command will likely need modification when upgrading to future Solaris software releases.The command line options provided with the `in.dhcpd`

daemon are used only for the current session, and include only some of the server options you can set. The dhcpsvc.conf(4) contains all the server default settings, and can be modified by using the dhcpmgr utility. See dhcpsvc.conf(4) for more details.

Options The following options are supported:

`-b automatic | manual` This option enables BOOTP compatibility mode, allowing the DHCP server to respond to BOOTP clients. The option argument specifies whether the DHCP server should automatically allocate permanent lease IP addresses to requesting BOOTP clients if the clients are not registered in the DHCP network tables (`automatic`) or respond only to BOOTP clients who have been manually registered in the DHCP network tables (`manual`). This option only affects DHCP server mode.

`-d` Debugging mode. The daemon remains as a foreground process, and displays verbose messages as it processes DHCP and/or BOOTP datagrams. Messages are displayed on the current TTY. This option can be used in both DHCP/BOOTP server mode and BOOTP relay agent mode.

`-h` *relay_hops* Specifies the maximum number of relay agent hops that can occur before the daemon drops the DHCP/BOOTP datagram. The default number of relay agent hops is 4. This option affects both DHCP/BOOTP server mode and BOOTP relay agent mode.

`-i` *interface, . . .* Selects the network interfaces that the daemon should monitor for DHCP/BOOTP datagrams. The daemon ignores DHCP/BOOTP datagrams on network interfaces not specified in this list. This option is only useful on machines that have multiple network interfaces. If this option is not specified, then the daemon listens for DHCP/BOOTP datagrams on all network interfaces. The option argument consists of a comma-separated list of interface names. It affects both DHCP/BOOTP server and BOOTP relay agent run modes.

`-l` *syslog_local_facility* The presence of this option turns on transaction logging for the DHCP server or BOOTP relay agent. The value specifies the `syslog` local facility (an integer from `0` to `7` inclusive) the DHCP daemon should use for tagging the transactions. Using a facility separate from the `LOG_DAEMON` facility allows the network administrator to capture these transactions separately from other DHCP daemon events for such

purposes as generating transaction reports. See syslog(3C), for details about local facilities. Transactions are logged using a record with 9 space-separated fields as follows:

1. Protocol:

```
Relay mode:     "BOOTP"
Server mode:    "BOOTP" or "DHCP" based upon client
                     type.
```

2. Type:

```
Relay mode:     "RELAY-CLNT", "RELAY-SRVR"
Server mode:    "ASSIGN", "EXTEND", "RELEASE",
                    "DECLINE", "INFORM", "NAK" "ICMP-ECHO."
```

3. Transaction time: absolute time in seconds (unix time)
4. Lease time:

```
Relay mode:     Always 0.
Server mode:    0 for ICMP-ECHO events, absolute time in
                    seconds (unix time)  otherwise
```

5. Source IP address: Dotted Internet form

   Relay mode: Relay interface IP on RELAY-CLNT, INADDR_ANY on RELAY-SRVR.
   Server mode: Client IP.

6. Destination IP address: Dotted Internet form

   Relay mode: Client IP on RELAY-CLNT, Server IP on RELAY-SRVR.
   Server mode: Server IP.

7. Client Identifier: Hex representation (0-9, A-F)

   Relay mode: MAC address
   Server mode: BOOTP - MAC address; DHCP - client id

8. Vendor Class identifier (white space converted to periods (.)).

   Relay mode: Always "N/A"
   Server mode: Vendor class ID tokenized by converting white space characters to periods (.)

9. MAC address: Hex representation (0-9, A-F)

Relay mode: MAC address
Server mode: MAC address

The format of this record is subject to change between releases.

Transactions are logged to the console if daemon is in debug mode (`-d`).

Logging transactions impact daemon performance.

It is suggested that you periodically rotate the DHCP transaction log file to keep it from growing until it fills the filesystem. This can be done in a fashion similar to that used for the general system message log `/var/adm/messages` and is best accomplished using the facilities provided by `logadm(1M)`.

`-n`

Disable automatic duplicate IP address detection. When this option is specified, the DHCP server does not attempt to verify that an IP address it is about to offer a client is not in use. By default, the DHCP server pings an IP address before offering it to a DHCP/BOOTP client, to verify that the address is not in use by another machine.

`-o` *DHCP_offer_time*

Specifies the number of seconds the DHCP server should cache the offers it has extended to discovering DHCP clients. The default setting is `10` seconds. On slow network media, this value can be increased to compensate for slow network performance. This option affects only DHCP server mode.

`-r` *IP_address* | *hostname, . . .*

This option enables BOOTP relay agent mode. The option argument specifies a comma-separated list of IP addresses or hostnames of DHCP or BOOTP servers to which the relay agent is to forward BOOTP requests. When the daemon is started in this mode, any DHCP tables are ignored, and the daemon simply acts as a BOOTP relay agent.

A BOOTP relay agent listens to UDP port 68, and forwards BOOTP request packets received on this port to the destinations specified on the command line. It supports the `BROADCAST` flag described in RFC 1542. A BOOTP relay

agent can run on any machine that has knowledge of local routers, and thus does not have to be an Internet gateway machine.

Note that the proper entries must be made to the `netmasks` database so that the DHCP server being served by the BOOTP relay agents can identify the subnet mask of the foreign BOOTP/DHCP client's network. See `netmasks(4)` for the format and use of this database.

`-t` *dhcptab_rescan_interval* — Specifies the interval in minutes that the DHCP server should use to schedule the automatic rereading of the `dhcptab` information. Typically, you would use this option if the changes to the `dhcptab` are relatively frequent. Once the contents of the `dhcptab` have stabilized, you can turn off this option to avoid needless reinitialization of the server.

`-v` — Verbose mode. The daemon displays more messages than in the default mode. Note that verbose mode can reduce daemon efficiency due to the time taken to display messages. Messages are displayed to the current TTY if the debugging option is used; otherwise, messages are logged to the `syslogd` facility. This option can be used in both DHCP/BOOTP server mode and BOOTP relay agent mode.

Examples

EXAMPLE 1 Starting a DHCP Server in BOOTP Compatibility Mode

The following command starts a DHCP server in BOOTP compatibility mode, permitting the server to automatically allocate permanent IP addresses to BOOTP clients which are not registered in the server's table; limits the server's attention to incoming datagrams on network devices `le2` and `tr0`; drops BOOTP packets whose hop count exceeds 2; configures the DHCP server to cache extended DHCP offers for 15 seconds; and schedules `dhcptab` rescans to occur every 10 minutes:

```
# in.dhcpd -i le2,tr0 -h 2 -o 15 -t 10 -b automatic
```

EXAMPLE 2 Starting the Daemon in BOOTP Relay Agent Mode

The following command starts the daemon in BOOTP relay agent mode, registering the hosts `bladerunner` and `10.0.0.5` as relay destinations, with debugging and verbose modes enabled, and drops BOOTP packets whose hop count exceeds 5:

```
# in.dhcpd -d -v -h 5 -r bladerunner,10.0.0.5
```

Files

- `/etc/inet/dhcpsvc.conf`
- `/etc/init/hosts`
- `/usr/lib/inet/dhcp/nsu/rfc2136.so.1`

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	service/network/dhcp
Interface Stability	Obsolete

See Also svcs(1), cron(1M), dhcpmgr(1M), dhtadm(1M), inetadm(1M), inetd(1M), logadm(1M), pntadm(1M), svcadm(1M), syslogd(1M), syslog(3C), dhcpsvc.conf(4), dhcp_network(4), dhcptab(4), ethers(4), hosts(4), netmasks(4), nsswitch.conf(4), attributes(5), dhcp(5), smf(5)

System Administration Guide: IP Services

Alexander, S., and R. Droms, *DHCP Options and BOOTP Vendor Extensions*, RFC 2132, Silicon Graphics, Inc., Bucknell University, March 1997.

Droms, R., *Interoperation Between DHCP and BOOTP*, RFC 1534, Bucknell University, October 1993.

Droms, R., *Dynamic Host Configuration Protocol*, RFC 2131, Bucknell University, March 1997.

Wimer, W., *Clarifications and Extensions for the Bootstrap Protocol*, RFC 1542, Carnegie Mellon University, October 1993.

Notes The in.dhcpd service is managed by the service management facility, smf(5), under the service identifier:

```
svc:/network/dhcp-server
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using svcadm(1M). Responsibility for initiating and restarting this service is delegated to inetd(1M). Use inetadm(1M) to make configuration changes and to view configuration information for this service. The service's status can be queried using the svcs(1) command.

Name in.discardd – UDP or TCP discard protocol service

Synopsis

```
in.discardd

FMRI svc:/internet/discard:default
```

Description FMRI stands for Fault Management Resource Identifier. It is used to identify resources managed by the Fault Manager. See `fmd(1M)` and `smf(5)`.

The `in.discardd` service provides the server-side of the discard protocol. This protocol is used for debugging and bandwidth measurement and is available on both TCP and UDP transports through port 9.

The `in.discardd` service is an `inetd(1M)` `smf(5)` delegated service. The `in.discardd` detects which transport is requested by examining the socket it is passed by the `inetd` daemon.

The discard service simply throws away any data it receives from the client.

TCP-based service — Once a connection is established, the `in.discardd` discards any data received. No response is generated. The connection remains open until the client terminates it.

UDP-based service — The `in.discardd` listens for UDP datagrams. When a datagram is received, the server discards it. No response is sent.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	service/network/legacy-network-services
Interface Stability	Committed

See Also `inetd(1M)`, `attributes(5)`, `smf(5)`

RFC 863

Name in.echod – UDP or TCP echo protocol service daemon

Synopsis

```
in.echod

FMRI svc:/internet/echo:default
```

Description FMRI stands for Fault Management Resource Identifier. It is used to identify resources managed by the Fault Manager. See `fmd(1M)` and `smf(5)`.

The `in.echod` service provides the server-side of the echo protocol. This protocol is used for debugging and bandwidth measurement and is available on both TCP and UDP transports, through port 7.

The `in.echod` service is an `inetd(1M)` `smf(5)` delegated service. The `in.echod` detects which transport is requested by examining the socket it is passed by the `inetd` daemon.

TCP-based service — Once a connection is established, the `in.echod` echoes any data received from the client back to the client. The server echoes data until the client program terminates the connection.

UDP-based service — The `in.echod` listens for UDP datagrams. When a datagram is received, the server creates a UDP datagram containing the data it received and sends it to the client.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	service/network/legacy-network-services
Interface Stability	Committed

See Also `inetd(1M)`, `attributes(5)`, `smf(5)`

RFC 862

Name inetadm – observe or configure inetd-controlled services

Synopsis

```
inetadm

inetadm -?

inetadm -p

inetadm -l {FMRI | pattern}

inetadm -e {FMRI | pattern}

inetadm -d {FMRI | pattern}

inetadm -m {FMRI | pattern}... {name=value}...

inetadm -M {name=value}...
```

Description The `inetadm` utility provides the following capabilities for `inetd`-managed SMF services:

- Provides a list of all such services installed.
- Lists the services' properties and values.
- Allows enabling and disabling of services.
- Allows modification of the services'property values, as well as the default values provided by `inetd`.

See `smf(5)` for a description of an SMF service.

With no arguments, `inetadm` lists all services under `inetd(1M)` control, including such attributes as their current run state and whether or not they are enabled.

Options For options taking one or more FMRI operands (see `smf(5)` for a description of an FMRI), if the operand specifies a service (instead of a service instance), and that service has only a single instance, `inetadm` operates on that instance.

If a service name is supplied and it contains more than one instances or a pattern is supplied and and it matches more than one instance, a warning message is displayed and that operand is ignored.

For those options taking *name=value* parameters, a description of each of the possible names and the allowed values is found in the `inetd(1M)` man page.

The following options are supported:

`-?` Display a usage message.

`-p` Lists all default `inet` service property values provided by `inetd` in the form of *name=value* pairs. If the value is of boolean type, it is listed as TRUE or FALSE.

`-l` {*FMRI* | *pattern*}...	List all properties for the specified service instances as *name=value* pairs. In addition, if the property value is inherited from the default value provided by `inetd`, the *name=value* pair is identified by the token (default). Property inheritance occurs when properties do not have a specified service instance default.
`-e` {*FMRI* | *pattern*}...	Enable the specified service instances.
`-d` {*FMRI* | *pattern*}...	Disable the specified service instances.
`-m` {*FMRI* | *pattern*}...{*name=value*}...	Change the values of the specified properties of the identified service instances. Properties are specified as whitespace-separated *name=value* pairs. To remove an instance-specific value and accept the default value for a property, simply specify the property without a value, for example, `name=` .
`-M` {*name=value*}...	Change the values of the specified `inetd` default properties. Properties are specified as whitespace-separated *name=value* pairs.

Examples

EXAMPLE 1 Displaying Properties for a Service

The following command displays the properties for the `spray` service.

```
# inetadm -l network/rpc/spray:default
SCOPE    NAME=VALUE
         name="sprayd"
         endpoint_type="tli"
         proto="datagram_v"
         isrpc=TRUE
         rpc_low_version=1
         rpc_high_version=1
         wait=TRUE
         exec="/usr/lib/netsvc/spray/rpc.sprayd"
         user="root"
default  bind_addr=""
default  bind_fail_max=-1
default  bind_fail_interval=-1
default  max_con_rate=-1
default  max_copies=-1
default  con_rate_offline=-1
default  failrate_cnt=40
default  failrate_interval=60
default  inherit_env=TRUE
default  tcp_trace=FALSE
```

EXAMPLE 1 Displaying Properties for a Service *(Continued)*

```
default  tcp_wrappers=FALSE
default  connection_backlog=10
```

EXAMPLE 2 Displaying Default Properties

The following command displays default properties.

```
# inetadm -p
NAME=VALUE
bind_addr=""
bind_fail_max=-1
bind_fail_interval=-1
max_con_rate=-1
max_copies=-1
con_rate_offline=-1
failrate_cnt=40
failrate_interval=60
inherit_env=TRUE
tcp_trace=FALSE
tcp_wrappers=FALSE
default  connection_backlog=10
```

EXAMPLE 3 Changing Property Values for a Service

The following command changes `rpc_high_version` to `3` and `tcp_trace` to `TRUE` for the `spray` service.

```
# inetadm -m network/rpc/spray:default \
         rpc_high_version=3 tcp_trace=TRUE
# inetadm -l network/rpc/spray:default
SCOPE    NAME=VALUE
         name="sprayd"
         endpoint_type="tli"
         proto="datagram_v"
         isrpc=TRUE
         rpc_low_version=1
         rpc_high_version=3
         wait=TRUE
         exec="/usr/lib/netsvc/spray/rpc.sprayd"
         user="root"
default  bind_addr=""
default  bind_fail_max=-1
default  bind_fail_interval=-1
default  max_con_rate=-1
default  max_copies=-1
default  con_rate_offline=-1
default  failrate_cnt=40
```

EXAMPLE 3 Changing Property Values for a Service *(Continued)*

```
default  failrate_interval=60
default  inherit_env=TRUE
         tcp_trace=TRUE
default  tcp_wrappers=FALSE
default  connection_backlog=10
```

Exit Status The following exit values are returned:

`0` Operation completed successfully.

`1` A fatal error occurred. An accompanying error message will provide further information.

`2` Invalid arguments were supplied, such as an ambiguous service FMRI or pattern.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os
Interface Stability	Committed

See Also `inetd(1M)`, `svccfg(1M)`, `attributes(5)`, `smf(5)`

Name inetconv – convert `inetd.conf` entries into smf service manifests, import them into smf repository

Synopsis

```
inetconv -?

inetconv [-f] [-n] [-i srcfile] [-o destdir]

inetconv -e [-n] [-i srcfile]
```

Description The `inetconv` utility converts a file containing records of `inetd.conf(4)` into `smf(5)` service manifests, and then import those manifests into the `smf` repository. Once the `inetd.conf` file has been converted, the only way to change aspects of an inet service is to use the `inetadm(1M)` utility.

There is a one-to-one correspondence between a service line in the input file and the manifest generated. By default, the manifests are named using the following template:

<*svcname*>-<*proto*>.xml

The <*svcname*> token is replaced by the service's name and the <*proto*> token by the service's protocol. Any slash (/) characters that exist in the source line for the service name or protocol are replaced with underscores (_).

The service line is recorded as a property of the converted service.

During the conversion process, if a service line is found to be malformed or to be for an internal `inetd` service, no manifest is generated and that service line is skipped.

The input file is left untouched by the conversion process.

Options The following options are supported:

`-?` Display a usage message.

`-e` Enable `smf` services which are listed in the input file.

`-f` If a service manifest of the same name as the one to be generated is found in the destination directory, `inetconv` will overwrite that manifest if this option is specified. Otherwise, an error message is generated and the conversion of that service is not performed.

`-i` *srcfile* Permits the specification of an alternate input file *srcfile*. If this option is not specified, then the `inetd.conf(4)` file is used as input.

`-n` Turns off the auto-import of the manifests generated during the conversion process. Later, if you want to import a generated manifest into the `smf(5)` repository, you can do so through the use of the `svccfg(1M)` utility.

If the `-e` option is specified, the `-n` option only displays the `smf` services that would be enabled.

`-o` Permits the specification of an alternate destination directory *destdir* for the generated manifests. If this option is not specified, then the manifests are placed in `/lib/svc/manifest/network/rpc`, if the service is a RPC service, or `/lib/svc/manifest/network` otherwise.

Examples

EXAMPLE 1 Generating smf Manifests from `inetd.conf`

The following command generates `smf(5)` manifests from `inetd.conf(4)` and places them in `/var/tmp`, overwriting any preexisting manifests of the same name, and then imports them into the `smf` repository.

```
# inetconv -f -o /var/tmp
100232/10 -> /var/tmp/100232_10-rpc_udp.xml
Importing 100232_10-rpc_udp.xml ...Done
telnet -> /var/tmp/telnet-tcp6.xml
Importing telnet-tcp6.xml ...Done
```

EXAMPLE 2 Generating Manifests from an Alternate Input File

The following command specifies a different input file and does not load the resulting manifests into the `smf` repository.

```
# inetconv -n -i /export/test/inet.svcs -o /var/tmp
100232/10 -> /var/tmp/100232_10-rpc_udp.xml
telnet -> /var/tmp/telnet-tcp6.xml
```

Exit Status

The following exit values are returned:

`0` Operation completed successfully (no errors).

`1` Invalid options specified.

`2` One or more service lines are malformed, and thus no manifest(s) were generated for them.

`3` An error occurred importing one or more of the generated manifests.

`4` A system error occurred.

Files

`/lib/svc/manifest/network/{rpc}/<svcname>-<proto>.xml`
Default output manifest file name.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os
Interface Stability	Committed

See Also inetadm(1M), inetd(1M), svccfg(1M), inetd.conf(4), attributes(5), smf(5)

Name inetd – Solaris Management Facility delegated restarter for inet services

Synopsis

```
inetd [configuration-file] start | stop | refresh

svc:/network/inetd:default
```

Description `inetd` is the delegated restarter for internet services for the Service Management Facility (SMF). Its basic responsibilities are to manage service states in response to administrative requests, system failures, and service failures; and, when appropriate, to listen for network requests for services.

Services are no longer managed by editing the inetd configuration file, `inetd.conf(4)`. Instead, you use `inetconv(1M)` to convert the configuration file content into SMF format services, then manage these services using `inetadm(1M)` and `svcadm(1M)`. Once a service has been converted by `inetconv`, any changes to the legacy data in the `inetd` config file will not become effective. However, `inetd` does alert the administrator when it notices change in the configuration file. See the start description under the "inetd Methods" section for further information.

Also note that the current `inetd` cannot be run from outside the SMF. This means it cannot be run from the command line, as was supported by the previous `inetd`. If you attempt to do this, a message is sent to stderr displaying mappings between the options supported by the previous `inetd` to the SMF version of `inetd`.

`inetd` listens for connections on behalf of all services that are in either the `online` or `degraded` state. A service enters one of these states when the service is enabled by the user and `inetd` manages to listen on its behalf. A listen attempt can fail if another server (whether standalone or a third-party internet service) is already listening on the same port. When this occurs, `inetd` logs this condition and continues trying to bind to the port at configured intervals a configured number of times. See the property `bind_fail_max` under "Service Properties," below, for more details.

The configuration of all `inetd`'s managed SMF services is read when it is started. It is reread when `inetd` is refreshed, which occurs in response to an SMF request, or when it receives a SIGHUP signal. See the `refresh` description under "inetd Methods" for the behavior on configuration refresh.

You can use the `inetadm(1M)` or `svccfg(1M)` utilities to make configuration changes to Internet services within the SMF repository. `inetadm` has the advantage over `svccfg` in that it provides an Internet/RPC service context.

Service States As part of its service management duties, `inetd` implements a state machine for each of its managed services. The states in this machine are made up of the `smf(5)` set of states. The semantics of these states are as follows:

`uninitialized`
: `inetd` has yet to process this service.

`online`
The service is handling new network requests and might have existing connections active.

`degraded`
The service has entered this state because it was able to listen and process requests for some, but not all, of the protocols specified for the service, having exhausted its listen retries. Existing network connections might be active.

`offline`
Connections might be active, but no new requests are being handled. This is a transient state. A service might be `offline` for any of the following reasons:

- The service's dependencies are unmet. When its dependencies become met the service's state will be re-evaluated.
- The service has exceeded its configured connection rate limit, `max_con_rate`. The service's state is re-evaluated when its connection offline timer, `con_rate_offline`, expires.
- The service has reached its allowed number of active connections, `max_copies`. The service's state is re-evaluated when the number of active connections drops below `max_copies`.
- `inetd` failed to listen on behalf of the service on all its protocols. As mentioned above, `inetd` retries up to a configured maximum number of times, at configured intervals.The service's state is re-evaluated when either a listen attempt is successful or the retry limit is reached.

`disabled`
The service has been turned off by an administrator, is not accepting new connections, and has none active. Administrator intervention is required to exit this state.

`maintenance`
A service is in this state because it is either malfunctioning and needs adminstrator attention or because an administrator has requested it.

Events constituting malfunctioning include: `inetd`'s inability to listen on behalf on any of the service's protocols before exceeding the service's bind retry limit, non-start methods returning with non-success return values, and the service exceeding its failure rate.

You request the maintenance state to perform maintenance on the service, such as applying a patch. No new requests are handled in this state, but existing connections might be active. Administrator intervention is required to exit this state.

Use `inetadm(1M)` to obtain the current state of a managed service.

Service Methods

As part of certain state transitions `inetd` will execute, if supplied, one of a set of methods provided by the service. The set of supported methods are:

`inetd_start`
Executed to handle a request for an `online` or `degraded` service. Since there is no separate state to distinguish a service with active connections, this method is not executed as part of a state transition.

`inetd_offline`
Executed when a service is taken from the `online` or `degraded` state to the `offline` state. For a `wait`-type service that at the time of execution is performing its own listening, this method should result in it ceasing listening. This method will be executed before the `disable` method in the case an `online`/`degraded` service is disabled. This method is required to be implemented for a `wait`-type service.

`inetd_online`
Executed when a service transitions from the `offline` state to the `online` state. This method allows a service author to carry out some preparation prior to a service starting to handle requests.

`inetd_disable`
Executed when a service transitions from the `offline` state to the `disabled` state. It should result in any active connections for a service being terminated.

`inetd_refresh`
Executed when both of the following conditions are met:

- `inetd` is refreshed, by means of the framework or a SIGHUP, or a request comes in to refresh the service, and
- the service is currently in the `online` state and there are no configuration changes that would result in the service needing to be taken `offline` and brought back again.

The only compulsory method is the `inetd_start` method. In the absence of any of the others, `inetd` runs no method but behaves as if one was run successfully.

Service Properties

Configuration for SMF–managed services is stored in the SMF repository. The configuration is made up of the basic configuration of a service, the configuration for each of the service's methods, and the default configuration applicable to all `inetd`-managed services.

For details on viewing and modifying the configuration of a service and the defaults, refer to `inetadm(1M)`.

The basic configuration of a service is stored in a property group named `inetd` in the service. The properties comprising the basic configuration are as follows:

`bind_addr`
The address of the network interface to which the service should be bound. An empty string value causes the service to accept connections on any network interface.

`bind_fail_interval`
The time interval in seconds between a failed bind attempt and a retry. The values `0` and `-1` specify that no retries are attempted and the first failure is handled the same as exceeding `bind_fail_max`.

`bind_fail_max`
The maximum number of times `inetd` retries binding to a service's associated port before giving up. The value `-1` specifies that no retry limit is imposed. If none of the service's protocols were bound to before any imposed limit is reached, the service goes to the `maintenance` state; otherwise, if not all of the protocols were bound to, the service goes to the `degraded` state.

`con_rate_offline`
The time in seconds a service will remain offline if it exceeds its configured maximum connection rate, `max_con_rate`. The values `0` and `-1` specify that connection rate limiting is disabled.

`connection_backlog`
The backlog queue size. Represents a limit on the number of incoming client requests that can be queued at the listening endpoints for servers.

`endpoint_type`
The type of the socket used by the service or the value `tli` to signify a TLI-based service. Valid socket type values are: `stream`, `dgram`, `raw`, `seqpacket`.

`failrate_cnt`
The count portion of the service's failure rate limit. The failure rate limit applies to `wait`-type services and is reached when *count* instances of the service are started within a given time. Exceeding the rate results in the service being transitioned to the `maintenance` state. This is different from the behavior of the previous `inetd`, which continued to retry every 10 minutes, indefinitely. The `failrate_cnt` check accounts for badly behaving servers that fail before consuming the service request and which would otherwise be continually restarted, taxing system resources. Failure rate is equivalent to the `-r` option of the previous `inetd`. The values `0` and `-1` specify that this feature is disabled.

`failrate_interval`
The time portion in seconds of the service's failure rate. The values `0` and `-1` specify that the failure rate limit feature is disabled.

`inherit_env`
If true, pass `inetd`'s environment on to the service's start method. Regardless of this setting, `inetd` will set the variables `SMF_FMRI`, `SMF_METHOD`, and `SMF_RESTARTER` in the start method's environment, as well as any environment variables set in the method context. These variables are described in `smf_method(5)`.

`isrpc`
If true, this is an RPC service.

`max_con_rate`
The maximum allowed connection rate, in connections per second, for a `nowait`-type service. The values `0` and `-1` specify that that connection rate limiting is disabled.

`max_copies`
The maximum number of copies of a `nowait` service that can run concurrently. The values `0` and `-1` specify that copies limiting is disabled.

`name`
Can be set to one of the following values:

- a service name understood by `getservbyname(3SOCKET)`;
- if `isrpc` is set to `true`, a service name understood by `getrpcbyname(3NSL)`;
- if `isrpc` is set to `true`, a valid RPC program number.

`proto`
In the case of socket-based services, this is a list of protocols supported by the service. Valid protocols are: `tcp`, `tcp6`, `tcp6only`, `udp`, `udp6`, and `udp6only`. In the case of TLI services, this is a list of netids recognized by `getnetconfigent(3NSL)` supported by the service, plus the values `tcp6only` and `udp6only`. RPC/TLI services also support nettypes in this list, and `inetd` first tries to interpret the list member as a nettype for these service types. The values `tcp6only` and `udp6only` are new to `inetd`; these values request that `inetd` listen only for and pass on true `IPv6` requests (not IPv4 mapped ones). See "Configuring Protocols for Sockets-Based Services," below.

`rpc_low_version`
Lowest supported RPC version. Required when `isrpc` is set to `true`.

`rpc_high_version`
Highest supported RPC version. Required when `isrpc` is set to `true`.

`tcp_keepalive`
If true, enable the periodic transmission of messages on a connected socket. If the connected party fails to respond to these messages, the connection is considered broken. This applies only to services with `endpoint_type` set to `streams` and wait set to `false`.

`tcp_trace`
If true, and this is a `nowait`-type service, `inetd` logs the client's IP address and TCP port number, along with the name of the service, for each incoming connection, using the `syslog(3C)` facility. `inetd` uses the `syslog` facility `code` daemon and `notice` priority level. See `syslog.conf(4)` for a description of `syslog` codes and severity levels. This logging is separate from the logging done by the TCP wrappers facility.

`tcp_trace` is equivalent to the previous `inetd`'s `-t` option (and the `/etc/default/inetd` property `ENABLE_CONNECTION_LOGGING`).

`tcp_wrappers`
If `true`, enable TCP wrappers access control. This applies only to services with `endpoint_type` set to `streams` and `wait` set to `false`. The `syslog` facility `code` daemon is used to log allowed connections (using the `notice` severity level) and denied traffic (using

the warning severity level). See syslog.conf(4) for a description of syslog codes and severity levels. The Interface Stability of the TCP wrappers facility and its configuration files is Volatile. As the TCP wrappers facility is not controlled by Sun, intra-release incompatibilities are not uncommon. See attributes(5).

For more information about configuring TCP wrappers, you can refer to the tcpd(1M) and hosts_access(4) man pages, which are delivered as part of the Solaris operating system at /usr/sfw/man. These pages are not part of the standard Solaris man pages, available at /usr/man.

tcp_wrappers is equivalent to the previous inetd's /etc/default/inetd property ENABLE_TCPWRAPPERS.

wait

If true this is a wait-type service, otherwise it is a nowait-type service. A wait-type service has the following characteristics:

- Its inetd_start method will take over listening duties on the service's bound endpoint when it is executed.
- inetd will wait for it to exit after it is executed before it resumes listening duties.

Datagram servers must be configured as being of type wait, as they are always invoked with the original datagram endpoint that will participate in delivering the service bound to the specified service. They do not have separate “listening” and “accepting” sockets. Connection-oriented services, such as TCP stream services can be designed to be either of type wait or nowait.

A number of the basic properties are optional for a service. In their absence, their values are taken from the set of default values present in the defaults property group in the inetd service. These properties, with their seed values, are listed below. Note that these values are configurable through inetadm(1M).

```
bind_fail_interval  -1
bind_fail_max       -1
con_rate_offline    -1
connection_backlog  10
failrate_count      40
failrate_time       60
inherit_env         true
max_con_rate        -1
max_copies          -1
tcp_keepalive       false
tcp_trace           false
tcp_wrappers        false
```

Each method specified for a service will have its configuration stored in the SMF repository, within a property group of the same name as the method. The set of properties allowable for

these methods includes those specified for the services managed by svc.startd(1M). (See svc.startd(1M) for further details.) Additionally, for the inetd_start method, you can set the arg0 property.

The arg0 property allows external wrapper programs to be used with inetd services. Specifically, it allows the first argument, argv[0], of the service's start method to be something other than the path of the server program.

In the case where you want to use an external wrapper program and pass arguments to the service's daemon, the arguments should be incorporated as arguments to the wrapper program in the exec property. For example:

```
exec='/path/to/wrapper/prog service_daemon_args'
arg0='/path/to/service/daemon'
```

In addition to the special method tokens mentioned in smf_method(5), inetd also supports the :kill_process token for wait-type services. This results in behavior identical to that if the :kill token were supplied, except that the kill signal is sent only to the parent process of the wait-type service's start method, not to all members of its encompassing process contract (see process(4)).

Configuring Protocols for Sockets-Based Services

When configuring inetd for a sockets-based service, you have the choice, depending on what is supported by the service, of the alternatives described under the proto property, above. The following are guidelines for which proto values to use:

- For a service that supports only IPv4: tcp and udp
- For a service that supports only IPv6: tcp6only and udp6only
- For a service that supports both IPv4 and IPv6:
  - Obsolete and not recommended: tcp6 and udp6
  - Recommended: use two separate entries that differ only in the proto field. One entry has tcp and the other has tcp6only, or udp plus udp6only.

See EXAMPLES for an example of a configuration of a service that supports both IPv4 and IPv6.

inetd Methods

inetd provides the methods listed below for consumption by the master restarter, svc.startd(1M).

start

Causes inetd to start providing service. This results in inetd beginning to handle smf requests for its managed services and network requests for those services that are in either the online or degraded state.

In addition, inetd also checks if the inetd.conf(4)–format configuration file it is monitoring has changed since the last inetconv(1M) conversion was carried out. If it has, then a message telling the administrator to re-run inetconv to effect the changes made is logged in syslog.

`stop`
Causes `inetd` to stop providing service. At this point, `inetd` transitions each of its services that are not in either the `maintenance` or `disabled` states to the `offline` state, running any appropriate methods in the process.

`refresh`
Results in a refresh being performed for each of its managed services and the `inetd.conf(4)` format configuration file being checked for change, as in the `start` method. When a service is refreshed, its behavior depends on its current state:

- if it is in the `maintenance` or `disabled` states, no action is performed because the configuration will be read and consumed when the service leaves the state;
- if it is in the `offline` state, the configuration will be read and any changes consumed immediately;
- if it is in the `online` or `degraded` state and the configuration has changed such that a re-binding is necessary to conform to it, then the service will be transitioned to the `offline` state and back again, using the new configuration for the bind;
- if it is in the `online` state and a re-binding is not necessary, then the `inetd_refresh` method of the service, if provided, will be run to allow `online` `wait`–type services to consume any other changes.

Options No options are supported.

Operands *configuration-file*
Specifies an alternate location for the legacy service file (`inetd.conf(4)`).

`start|stop|refresh`
Specifies which of `inetd`'s methods should be run.

Examples **EXAMPLE 1** Configuring a Service that Supports Both IPv4 and IPv6

The following commands illustrate the existence of services that support both IPv4 and IPv6 and assign `proto` properties to those services.

```
example# svcs -a | grep mysvc
online         15:48:29 svc:/network/mysvc:dgram4
online         15:48:29 svc:/network/mysvc:dgram6
online         15:51:47 svc:/network/mysvc:stream4
online         15:52:10 svc:/network/mysvc:stream6

# inetadm -M network/rpc/mysvc:dgram4 proto=udp
# inetadm -M network/rpc/mysvc:dgram6 proto=udp6only
# inetadm -M network/rpc/mysvc:stream4 proto=tcp
# inetadm -M network/rpc/mysvc:stream6 proto=tcp6only
```

See `svcs(1)` and `inetadm(1M)` for descriptions of those commands.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os
Interface Stability	Committed

See Also fmd(1M), inetadm(1M), inetconv(1M), svcadm(1M), svccfg(1M), svcs(1), svc.startd(1M), syslog(3C), getnetconfigent(3NSL), getrpcbyname(3NSL), getservbyname(3SOCKET), inetd.conf(4), process(4), syslog.conf(4), attributes(5), smf(5), smf_method(5)

Notes The inetd daemon performs the same function as, but is implemented significantly differently from, the daemon of the same name in Solaris 9 and prior Solaris operating system releases. In the current Solaris release, inetd is part of the Solaris Management Facility (see smf(5)) and will run only within that facility.

The /etc/default/inetd file has been deprecated. The functionality represented by the properties ENABLE_CONNECTION_LOGGING and ENABLE_TCP_WRAPPERS are now available as the tcp_trace and tcp_wrappers properties, respectively. These properties are described above, under “Service Properties”.

Name in.fingerd, fingerd – remote user information server

Synopsis

```
/usr/sbin/in.fingerd [-s]
```

Description `fingerd` implements the server side of the Name/Finger protocol, specified in *RFC 742*. The Name/Finger protocol provides a remote interface to programs which display information on system status and individual users. The protocol imposes little structure on the format of the exchange between client and server. The client provides a single command line to the finger server which returns a printable reply.

`fingerd` waits for connections on TCP port 79. Once connected, it reads a single command line terminated by RETURN-LINEFEED and passes the arguments to `finger(1)`, prepended with `-s`. `fingerd` closes its connections as soon as the output is finished.

Options `fingerd` supports the following option.

`-s` Enable secure mode. Deny forwarding of queries to other remote hosts.

Files

`/var/adm/utmpx`	User and accounting information.
`/etc/passwd`	System password file.
`/var/adm/lastlog`	Last login times.
`$HOME/.plan`	User's plans.
`$HOME/.project`	User's projects.

Usage `fingerd` and `in.fingerd` are IPv6–enabled. See `ip6(7P)`.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	service/network/network-servers

See Also `finger(1)`, `svcs(1)`, `inetadm(1M)`, `inetd(1M)`, `svcadm(1M)`, `attributes(5)`, `smf(5)`, `ip6(7P)`

Harrenstien, Ken, *RFC 742, NAME/FINGER*, Network Information Center, SRI International, Menlo Park, Calif., December 1977.

Notes Connecting directly to the server from a TIP or an equally narrow-minded TELNET-protocol user program can result in meaningless attempts at option negotiation being sent to the server, which foul up the command line interpretation. `fingerd` should be taught to filter out IAC's and perhaps even respond negatively (IAC does not) to all option commands received.

The `in.fingerd` service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/network/finger:default
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using svcadm(1M). Responsibility for initiating and restarting this service is delegated to inetd(1M). Use inetadm(1M) to make configuration changes and to view configuration information for this service. The service's status can be queried using the svcs(1) command.

Name infocmp – compare or print out terminfo descriptions

Synopsis

```
/usr/bin/infocmp [-d] [-c] [-n] [-I] [-L] [-C] [-r] [-u]
     [-s | d | i | l | c] [-v] [-V] [-1] [-w width]
     [-A  directory] [-B directory] [termname]...
```

Description `infocmp` compares a binary `terminfo` entry with other terminfo entries, rewrites a `terminfo` description to take advantage of the `use=` terminfo field, or prints out a `terminfo` description from the binary file (`term`) in a variety of formats. It displays boolean fields first, then numeric fields, followed by the string fields. If no options are specified and zero, or one *termname* is specified, the `-I` option is assumed. If more than one *termname* is specified, the `-d` option is assumed.

Options The `-d` , `-c` , and `-n` options can be used for comparisons. `infocmp` compares the `terminfo` description of the first terminal *termname* with each of the descriptions given by the entries for the other terminal's *termname*. If a capability is defined for only one of the terminals, the value returned will depend on the type of the capability: `F` for boolean variables, –`1` for integer variables, and `NULL` for string variables.

`-d` Produce a list of each capability that is different between two entries. This option is useful to show the difference between two entries, created by different people, for the same or similar terminals.

`-c` Produce a list of each capability that is common between two entries. Capabilities that are not set are ignored. This option can be used as a quick check to see if the `-u` option is worth using.

`-n` Produce a list of each capability that is in neither entry. If no *termname* is given, the environment variable `TERM` will be used for both of the *termname*s. This can be used as a quick check to see if anything was left out of a description.

The `-I` , `-L` , and `-C` options will produce a source listing for each terminal named.

`-I` Use the `terminfo` names.

`-L` Use the long C variable name listed in < `term.h` >.

`-C` Use the `termcap` names. The source produced by the `-C` option may be used directly as a `termcap` entry, but not all of the parameterized strings may be changed to the `termcap` format. `infocmp` will attempt to convert most of the parameterized information, but anything not converted will be plainly marked in the output and commented out. These should be edited by hand.

`-r` When using `-C` , put out all capabilities in `termcap` form.

If no *termname* is given, the environment variable `TERM` will be used for the terminal name.

All padding information for strings will be collected together and placed at the beginning of the string where `termcap` expects it. Mandatory padding (padding information with a trailing '/') will become optional.

All termcap variables no longer supported by terminfo , but are derivable from other terminfo variables, will be displayed. Not all terminfo capabilities will be translated; only those variables which were part of termcap will normally be displayed. Specifying the -r option will take off this restriction, allowing all capabilities to be displayed in termcap form.

Note that because padding is collected to the beginning of the capability, not all capabilities are displayed. Mandatory padding is not supported. Because termcap strings are not as flexible, it is not always possible to convert a terminfo string capability into an equivalent termcap format. A subsequent conversion of the termcap file back into terminfo format will not necessarily reproduce the original terminfo source.

Some common terminfo parameter sequences, their termcap equivalents, and some terminal types which commonly have such sequences, are:

```
terminfo     termcap      Representative Terminals
%p1%c    %.    adm
%p1%d    %d    hp, ANSI standard, vt100
%p1%'x'%+%c    %+x    concept
%i    %i    ANSI standard, vt100
%p1%?%'x'%>%t%p1%'y'%+%;    %>xy    concept
%p2 is printed before %p1    %r    hp
```

-u Produce a terminfo source description of the first terminal *termname* which is relative to the sum of the descriptions given by the entries for the other terminals' *termname*s. It does this by analyzing the differences between the first *termname* and the other *termnames* and producing a description with use= fields for the other terminals. In this manner, it is possible to retrofit generic terminfo entries into a terminal's description. Or, if two similar terminals exist, but were coded at different times, or by different people so that each description is a full description, using infocmp will show what can be done to change one description to be relative to the other.

A capability is displayed with an at-sign (@) if it no longer exists in the first *termname*, but one of the other *termname* entries contains a value for it. A capability's value is displayed if the value in the first *termname* is not found in any of the other *termname* entries, or if the first of the other *termname* entries that has this capability gives a different value for that capability.

The order of the other *termname* entries is significant. Since the terminfo compiler tic does a left-to-right scan of the capabilities, specifying two use= entries that contain differing entries for the same capabilities will produce different results, depending on the order in which the entries are given. infocmp will flag any such inconsistencies between the other *termname* entries as they are found.

Alternatively, specifying a capability *after* a use= entry that contains, it will cause the second specification to be ignored. Using infocmp to recreate a description can be a useful check to make sure that everything was specified correctly in the original source description.

Another error that does not cause incorrect compiled files, but will slow down the compilation time, is specifying superfluous `use=` fields. `infocmp` will flag any superfluous `use=` fields.

`-s` Sorts the fields within each type according to the argument below:

- `d` Leave fields in the order that they are stored in the `terminfo` database.
- `i` Sort by `terminfo` name.
- `l` Sort by the long C variable name.
- `c` Sort by the `termcap` name.

If the `-s` option is not given, the fields are sorted alphabetically by the `terminfo` name within each type, except in the case of the `-C` or the `-L` options, which cause the sorting to be done by the `termcap` name or the long C variable name, respectively.

`-v` Print out tracing information on standard error as the program runs.

`-V` Print out the version of the program in use on standard error and exit.

`–1` Print the fields one to a line. Otherwise, the fields are printed several to a line to a maximum width of 60 characters.

`-w`*width* Changes the output to *width* characters.

The location of the compiled `terminfo` database is taken from the environment variable `TERMINFO` . If the variable is not defined, or the terminal is not found in that location, the system `terminfo` database, usually in `/usr/share/lib/terminfo`, is used. The options `-A` and `-B` may be used to override this location.

`-A` *directory* Set `TERMINFO` for the first *termname*.

`-B` *directory* Set `TERMINFO` for the other *termname*s. With this, it is possible to compare descriptions for a terminal with the same name located in two different databases. This is useful for comparing descriptions for the same terminal created by different people.

Files `/usr/share/lib/terminfo/?/*` Compiled terminal description database.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also `captoinfo(1M)`, `tic(1M)`, `curses(3CURSES)`, `terminfo(4)`, `attributes(5)`

Name in.iked – daemon for the Internet Key Exchange (IKE)

Synopsis

```
/usr/lib/inet/in.iked [-d] [-f filename] [-p level]

/usr/lib/inet/in.iked -c [-f filename]
```

Description `in.iked` performs automated key management for IPsec using the Internet Key Exchange (IKE) protocol.

`in.iked` implements the following:

- IKE authentication with either pre-shared keys, DSS signatures, RSA signatures, or RSA encryption.
- Diffie-Hellman key derivation using either `768`, `1024`, `1536`, `2048`, `3072`, or `4096`-bit public key moduli, or `256`, `384`, or `521`-bit elliptic curve moduli.
- Authentication protection with cipher choices of AES, DES, Blowfish, or 3DES, and hash choices of either `HMAC-MD5` or `HMAC-SHA-1`. Encryption in `in.iked` is limited to the IKE authentication and key exchange. See `ipsecesp(7P)` for information regarding IPsec protection choices.

`in.iked` is managed by the following `smf(5)` service:

```
svc:/network/ipsec/ike
```

This service is delivered disabled because the configuration file needs to be created before the service can be enabled. See `ike.config(4)` for the format of this file.

See "Service Management Facility" for information on managing the `smf(5)` service.

`in.iked` listens for incoming IKE requests from the network and for requests for outbound traffic using the `PF_KEY` socket. See `pf_key(7P)`.

`in.iked` has two support programs that are used for IKE administration and diagnosis: `ikeadm(1M)` and `ikecert(1M)`.

The `ikeadm(1M)` command can read the `/etc/inet/ike/config` file as a `rule`, then pass the configuration information to the running `in.iked` daemon using a doors interface.

```
example# ikeadm read rule /etc/inet/ike/config
```

Refreshing the `ike` `smf(5)` service provided to manage the `in.iked` daemon sends a `SIGHUP` signal to the `in.iked` daemon, which will (re)read `/etc/inet/ike/config` and reload the certificate database.

The preceding two commands have the same effect, that is, to update the running IKE daemon with the latest configuration. See "Service Management Facility" for more details on managing the `in.iked` daemon.

When Trusted Extensions are enabled (see labeld(1M)), in.iked can be used in the global zone to negotiate labeled security associations. On labeled systems using in.iked, UDP ports 500 and 4500 must be configured as multi-level ports for the global zone in the tnzonecfg file (see tnzonecfg(4), part of the *Solaris Trusted Extensions Reference Manual*). See ike.config(4) for more information on configuring in.iked to be label-aware.

Service Management Facility

The IKE daemon (in.iked) is managed by the service management facility, smf(5). The following group of services manage the components of IPsec:

```
svc:/network/ipsec/ipsecalgs   (See ipsecalgs(1M))
svc:/network/ipsec/policy      (See ipsecconf(1M))
svc:/network/ipsec/manual-key  (See ipseckey(1M))
svc:/network/ipsec/ike         (see ike.config(4))
```

The manual-key and ike services are delivered disabled because the system administrator must create configuration files for each service, as described in the respective man pages listed above.

The correct administrative procedure is to create the configuration file for each service, then enable each service using svcadm(1M).

The ike service has a dependency on the ipsecalgs and policy services. These services should be enabled before the ike service. Failure to do so results in the ike service entering maintenance mode.

If the configuration needs to be changed, edit the configuration file then refresh the service, as follows:

```
example# svcadm refresh ike
```

The following properties are defined for the ike service:

config/admin_privilege
: Defines the level that ikeadm(1M) invocations can change or observe the running in.iked. The acceptable values for this property are the same as those for the -p option. See the description of -p in OPTIONS.

config/config_file
: Defines the configuration file to use. The default value is /etc/inet/ike/config. See ike.config(4) for the format of this file. This property has the same effect as the -f flag. See the description of -f in OPTIONS.

config/debug_level
: Defines the amount of debug output that is written to the debug_logfile file, described below. The default value for this is op or operator. This property controls the recording of information on events such as re-reading the configuration file. Acceptable value for debug_level are listed in the ikeadm(1M) man page. The value all is equivalent to the -d flag. See the description of -d in OPTIONS.

`config/debug_logfile`
Defines where debug output should be written. The messages written here are from debug code within `in.iked`. Startup error messages are recorded by the `smf(5)` framework and recorded in a service-specific log file. Use any of the following commands to examine the `logfile` property:

```
example# svcs -l ike
example# svcprop ike
example# svccfg -s ike listprop
```

The values for these log file properties might be different, in which case both files should be inspected for errors.

`config/ignore_errors`
A boolean value that controls `in.iked`'s behavior should the configuration file have syntax errors. The default value is `false`, which causes `in.iked` to enter maintenance mode if the configuration is invalid.

Setting this value to `true` causes the IKE service to stay online, but correct operation requires the administrator to configure the running daemon with `ikeadm(1M)`. This option is provided for compatibility with previous releases.

These properties can be modified using `svccfg(1M)` by users who have been assigned the following authorization:

```
solaris.smf.value.ipsec
```

PKCS#11 token objects can be unlocked or locked by using `ikeadm` token login and `ikeadm` token logout, respectively. Availability of private keying material stored on these PKCS#11 token objects can be observed with: `ikeadm dump certcache`. The following authorizations allow users to log into and out of PKCS#11 token objects:

```
solaris.network.ipsec.ike.token.login
solaris.network.ipsec.ike.token.logout
```

See `auths(1)`, `ikeadm(1M)`, `user_attr(4)`, `rbac(5)`.

The service needs to be refreshed using `svcadm(1M)` before a new property value is effective. General, non-modifiable properties can be viewed with the `svcprop(1)` command.

```
# svccfg -s ipsec/ike setprop config/config_file = \
/new/config_file
# svcadm refresh ike
```

Administrative actions on this service, such as enabling, disabling, refreshing, and requesting restart can be performed using `svcadm(1M)`. A user who has been assigned the authorization shown below can perform these actions:

```
solaris.smf.manage.ipsec
```

The service's status can be queried using the `svcs(1)` command.

The `in.iked` daemon is designed to be run under `smf(5)` management. While the `in.iked` command can be run from the command line, this is discouraged. If the `in.iked` command is to be run from the command line, the `ike` `smf(5)` service should be disabled first. See `svcadm(1M)`.

Options The following options are supported:

`-c` Check the syntax of a configuration file.

`-d` Use debug mode. The process stays attached to the controlling terminal and produces large amounts of debugging output. This option is deprecated. See "Service Management Facility" for more details.

`-f` *filename* Use *filename* instead of `/etc/inet/ike/config`. See `ike.config(4)` for the format of this file. This option is deprecated. See "Service Management Facility" for more details.

`-p` *level* Specify privilege level (*level*). This option sets how much `ikeadm(1M)` invocations can change or observe about the running `in.iked`.

Valid *levels* are:

0 Base level

1 Access to preshared key info

2 Access to keying material

If `-p` is not specified, *level* defaults to `0`.

This option is deprecated. See "Service Management Facility" for more details.

Security This program has sensitive private keying information in its image. Care should be taken with any core dumps or system dumps of a running `in.iked` daemon, as these files contain sensitive keying information. Use the `coreadm(1M)` command to limit any corefiles produced by `in.iked`.

Files

`/etc/inet/ike/config`	Default configuration file.
`/etc/inet/secret/ike.privatekeys/*`	Private keys. A private key *must* have a matching public-key certificate with the same filename in `/etc/inet/ike/publickeys/`.
`/etc/inet/ike/publickeys/*`	Public-key certificates. The names are only important with regard to matching private key names.
`/etc/inet/ike/crls/*`	Public key certificate revocation lists.
`/etc/inet/secret/ike.preshared`	IKE pre-shared secrets for Phase I authentication.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also svcs(1), coreadm(1M), ikeadm(1M), ikecert(1M), labeld(1M), svccfg(1M), svcadm(1M), ike.config(4), attributes(5), smf(5), ipsecesp(7P), pf_key(7P)

See tnzonecfg(4), part of the *Solaris Trusted Extensions Reference Manual*.

Harkins, Dan and Carrel, Dave. *RFC 2409, Internet Key Exchange (IKE)*. Network Working Group. November 1998.

Maughan, Douglas, Schertler, M., Schneider, M., Turner, J. *RFC 2408, Internet Security Association and Key Management Protocol (ISAKMP)*. Network Working Group. November 1998.

Piper, Derrell, *RFC 2407, The Internet IP Security Domain of Interpretation for ISAKMP*. Network Working Group. November 1998.

Fu, D.; Solinos, J., *RFC 4753, ECP Groups for IKE and IKEv2*. Network Working Group. January 2007.

Lepinski, M.; Kent, S., *RFC 5114, Additional Diffie-Hellman Groups for Use with IETF Standards*. Network Working Group. January 2008.

Name init – process control initialization

Synopsis `/usr/sbin/init [0123456abcQqSs]`

Description `init` is the default primordial user process. (Options given to the kernel during boot may result in the invocation of an alternative primordial user process, as described on `kernel(1M)`). `init` initiates the core components of the service management facility, `svc.configd(1M)` and `svc.startd(1M)`, and restarts these components if they fail. For backwards compatibility, `init` also starts and restarts general processes according to `/etc/inittab`, as desribed below.

The run levels and system booting descriptions given below are provided for compatibility purposes only, and otherwise made obsolete by the service management facility, `smf(5)`.

init Failure If `init` exits for any reason other than system shutdown, it will be restarted with process-ID 1.

Run Level Defined At any given time, the system is in one of eight possible run levels. A run level is a software configuration under which only a selected group of processes exists. Processes spawned by `init` for each of these run levels are defined in `/etc/inittab`. `init` can be in one of eight run levels, `0`–`6` and `S` or `s` (`S` and `s` are identical). The run level changes when a privileged user runs `/usr/sbin/init`.

init and System Booting When the system is booted, `init` is invoked and the following occurs. First, it reads the properties for the `svc:/system/environment:init` service. Among these properties are values for locale-related environments, such as `LANG` or `LC_CTYPE`. `init` then looks in `/etc/inittab` for the `initdefault` entry (see `inittab(4)`). If the `initdefault` entry:

exists
: `init` usually uses the run level specified in that entry as the initial run level to enter only if the options/milestone property has not been specified for `svc.startd(1M)`.

does not exist
: The service management facility, `smf(5)`, examines its configuration specified in `svc.startd(1M)`, and enters the milestone specified by the options/milestone property.

The `initdefault` entry in `/etc/inittab` corresponds to the following run levels:

`S` or `s`
: `init` goes to the single-user state. In this state, the system console device (`/dev/console`) is opened for reading and writing and the command `/usr/sbin/su`, (see `su(1M)`), is invoked. Use `init` to change the run level of the system. Note that if the shell is terminated (using an end-of-file), `init` only re-initializes to the single-user state if `/etc/inittab` does not exist.

`0-6`
: `init` enters the corresponding run level. Run levels `0`, `5`, and `6` are reserved states for shutting the system down. Run levels `2`, `3`, and `4` are available as multi-user operating states.

If this is the first time since power up that `init` has entered a run level other than single-user state, `init` first scans `/etc/inittab` for `boot` and `bootwait` entries (see `inittab(4)`). These entries are performed before any other processing of `/etc/inittab` takes place, providing

that the run level entered matches that of the entry. In this way any special initialization of the operating system, such as mounting file systems, can take place before users are allowed onto the system. `init` then scans `/etc/inittab` and executes all other entries that are to be processed for that run level.

To spawn each process in `/etc/inittab`, `init` reads each entry and for each entry that should be respawned, it forks a child process. After it has spawned all of the processes specified by `/etc/inittab`, `init` waits for one of its descendant processes to die, a `powerfail` signal, or a signal from another `init` process to change the system's run level. When one of these conditions occurs, `init` re-examines `/etc/inittab`.

inittab Additions

New entries can be added to `/etc/inittab` at any time; however, `init` still waits for one of the above three conditions to occur before re-examining `/etc/inittab`. To get around this, `init Q` or `init q` command wakes `init` to re-examine `/etc/inittab` immediately.

When `init` comes up at boot time and whenever the system changes from the single-user state to another run state, `init` sets the `ioctl(2)` states of the console to those modes saved in the file `/etc/ioctl.syscon`. `init` writes this file whenever the single-user state is entered.

Run Level Changes

When a run level change request is made, `init` or a designate sends the warning signal (`SIGTERM`) to all processes that are undefined in the target run level. A minimum interval of five seconds is observed before `init` or its designate forcibly terminates these processes by sending a kill signal (`SIGKILL`). Additionally, init informs `svc.startd(1M)` that the run level is changing. `svc.startd(1M)` then restricts the system to the set of services which the milestone corresponding to the run-level change depends on.

When `init` receives a signal telling it that a process it spawned has died, it records the fact and the reason it died in `/var/adm/utmpx` and `/var/adm/wtmpx` if it exists (see `who(1)`). A history of the processes spawned is kept in `/var/adm/wtmpx`.

If `init` receives a `powerfail` signal (`SIGPWR`) it scans `/etc/inittab` for special entries of the type `powerfail` and `powerwait`. These entries are invoked (if the run levels permit) before any further processing takes place. In this way `init` can perform various cleanup and recording functions during the powerdown of the operating system.

Setting Environment Variables

You can set default values for environment variables, for such items as timezone and character formatting, in the list of properties for the `svc:/system/environment:init` service.

Security

`init` uses `pam(3PAM)` for session management. The PAM configuration policy, configured in either `/etc/pam.conf` or per-service files in `/etc/pam.d/`, specifies the session management module to be used for `init`. Here is a partial `pam.conf` file with entries for `init` using the UNIX session management module.

```
init    session    required    pam_unix_session.so.1
```

The equivalent PAM configuration using `/etc/pam.d/` would be the following entry in `/etc/pam.d/init`:

```
session required    pam_unix_session.so.1
```

If there are no entries for the init service in `/etc/pam.conf` and no `/etc/pam.d/init` file exists, then the entries for the "other" service in `/etc/pam.conf` will be used. If there are not any entries in `/etc/pam.conf` for the "other" service, then the entries in `/etc/pam.d/other` will be used.

Options `0`

Go into firmware.

`1`

Put the system in system administrator mode. All local file systems are mounted. Only a small set of essential kernel processes are left running. This mode is for administrative tasks such as installing optional utility packages. All files are accessible and no users are logged in on the system.

This request corresponds to a request for `smf(5)` to restrict the system milestone to svc:/milestone/single-user:default.

`2`

Put the system in multi-user mode. All multi-user environment terminal processes and daemons are spawned. This state is commonly referred to as the multi-user state.

This request corresponds to a request for `smf(5)` to restrict the system milestone to svc:/milestone/multi-user:default.

`3`

Extend multi-user mode by making local resources available over the network.

This request corresponds to a request for `smf(5)` to restrict the system milestone to svc:/milestone/multi-user-server:default.

`4`

Is available to be defined as an alternative multi-user environment configuration. It is not necessary for system operation and is usually not used.

`5`

Shut the machine down so that it is safe to remove the power. Have the machine remove power, if possible.

`6`

Stop the operating system and reboot to the state defined by the `initdefault` entry in `/etc/inittab`.

The service `svc:/system/boot-config:default` is enabled by default. When the `config/fastreboot_default` property is set to `true`, `init 6` will bypass certain firmware initialization and test steps, depending on the specific capabilities of the system.

a,b,c
: Process only those /etc/inittab entries having the a, b, or c run level set. These are pseudo-states, which may be defined to run certain commands, but which do not cause the current run level to change.

Q,q
: Re-examine /etc/inittab.

S, s
: Enter single-user mode. This is the only run level that doesn't require the existence of a properly formatted /etc/inittab file. If this file does not exist, then by default, the only legal run level that init can enter is the single-user mode. When in single-user mode, the filesystems required for basic system operation will be mounted. When the system comes down to single-user mode, these file systems will remain mounted (even if provided by a remote file server), and any other local filesystems will also be left mounted. During the transition down to single-user mode, all processes started by init or init.d scripts that should only be running in multi-user mode are killed. In addition, any process that has a utmpx entry will be killed. This last condition insures that all port monitors started by the SAC are killed and all services started by these port monitors, including ttymon login services, are killed.

  This request corresponds to a request for smf(5) to restrict the system milestone to svc:/milestone/single-user:default.

Files

/dev/console
: System console device.

/etc/default/init
: This file is Obsolete and might be removed in a future release. Instead of obtaining values from this file, the init process reads properties for the svc:/system/environment:init service. To make changes that were formerly made by editing /etc/default/init, an administrator with the System Administrator or System Configuration rights profile can set the corresponding properties of the init service instance and refresh the instance.

  This read-only file contains environment variables and their default values. The variables are:

  TZ
  : Always set to localtime. To set the system timezone, an administrator must set the timezone/localtime property in timezone:default SMF service.

  CMASK
  : The mask (see umask(1)) that init uses and that every process inherits from the init process. If not set, init uses the mask it inherits from the kernel. Note that init always attempts to apply a umask of 022 before creating a file, regardless of the setting of CMASK

  LC_CTYPE
  : Character characterization information

`LC_MESSAGES`
Message translation

`LC_MONETARY`
Monetary formatting information

`LC_NUMERIC`
Numeric formatting information

`LC_TIME`
Time formatting information

`LC_ALL`
If set, all other `LC_*` environmental variables take-on this value.

`LANG`
If `LC_ALL` is not set, and any particular `LC_*` is also not set, the value of `LANG` is used for that particular environmental variable.

`/etc/inittab`
Controls process dispatching by `init`.

`/etc/ioctl.syscon`
ioctl states of the console, as saved by `init` when single-user state is entered.

`/etc/svc/volatile/init.state`
`init` state necessary to recover from failure.

`/var/adm/utmpx`
User access and administration information.

`/var/adm/wtmpx`
History of user access and administration information.

`/var/run/initpipe`
A named pipe used for internal communication.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also `login(1)`, `sh(1)`, `stty(1)`, `who(1)`, `kernel(1M)`, `shutdown(1M)`, `su(1M)`, `svc.configd(1M)`, `svc.startd(1M)`, `ttymon(1M)`, `ioctl(2)`, `kill(2)`, `ctime(3C)`, `pam(3PAM)`, `init.d(4)`, `inittab(4)`, `pam.conf(4)`, `TIMEZONE(4)`, `utmpx(4)`, `attributes(5)`, `pam_unix_session(5)`, `smf(5)`, `termio(7I)`

Diagnostics If `init` finds that it is respawning an entry from `/etc/inittab` more than ten times in two minutes, it assumes that there is an error in the command string in the entry and generates an error message on the system console. It then refuses to respawn this entry until either five minutes has elapsed or it receives a signal from a user-spawned `init` command. This prevents `init` from eating up system resources when someone makes a typographical error in the `inittab` file, or a program is removed that is referenced in `/etc/inittab`.

Notes `init` can be run only by a privileged user.

The `S` or `s` state must not be used indiscriminately in `/etc/inittab`. When modifying this file, it is best to avoid adding this state to any line other than `initdefault`.

If a default state is not specified in the `initdefault` entry in `/etc/inittab`, state `6` is entered. Consequently, the system will loop by going to firmware and rebooting continuously.

If the `utmpx` file cannot be created when booting the system, the system will boot to state "`s`" regardless of the state specified in the `initdefault` entry in `/etc/inittab`. This can occur if the `/var` file system is not accessible.

When a system transitions down to the `S` or `s` state, the `/etc/nologin` file (see `nologin(4)`) is created. Upon subsequent transition to run level 2, this file is removed.

`init` uses `/var/run/initpipe`, a named pipe, for internal communication.

Name init.sma – start and stop the snmpd daemon

Synopsis

```
/etc/init.d/init.sma start | stop | restart | status
```

Description The `init.sma` utility is run automatically during installation and each time the system is rebooted. This utility manages the `snmpd`.

Options The following options are supported:

`start` Starts the `snmpd` daemon.

`stop` Stops the `snmpd` daemon.

`restart` Stops then starts the `snmpd` daemon.

`status` Reports the `snmpd` daemon's status.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsmagt
Interface Stability	Uncommitted

See Also `attributes(5)`

Name init.wbem – start and stop the CIM Boot Manager

Synopsis

```
svccfg svc:/application/management/wbem start | stop
```

Description The `init.wbem` utility is run automatically during installation, each time the system is rebooted, and by means of the `svccfg(1M)` command. This method manipulates the CIM Object Manager (CIMOM). `init.wbem` can be used to start, stop, or retrieve status from the server.

CIM Object Manager

The CIM Object Manager manages CIM objects on a WBEM-enabled system. A CIM object is a computer representation, or model, of a managed resource, such as a printer, disk drive, or CPU. CIM objects are stored internally as Java classes.

When a WBEM client application accesses information about a CIM object, the CIM Object Manager contacts either the appropriate provider for that object or the CIM Object Manager Repository. Providers are classes that communicate with managed objects to access data.

When a WBEM client application requests data from a managed resource that is not available from the CIM Object Manager Repository, the CIM Object Manager forwards the request to the provider for that managed resource. The provider dynamically retrieves the information.

At startup, the CIM Object Manager performs the following functions:

- Listens for RMI connections on RMI port 5987 and for XML/HTTP connections on HTTP port 5988.
- Sets up a connection to the CIM Object Manager Repository.
- Waits for incoming requests.

During normal operations, the CIM Object Manager performs the following functions:

- Performs security checks to authenticate user login and authorization to access namespaces.
- Performs syntactical and semantic checking of CIM data operations to ensure that they comply with the latest CIM Specification.
- Routes requests to the appropriate provider or to the CIM Object Manager Repository.
- Delivers data from providers and from the CIM Object Manager Repository to WBEM client applications.

A WBEM client application contacts the CIM Object Manager to establish a connection when it needs to perform WBEM operations, such as creating a CIM class or updating a CIM instance. When a WBEM client application connects to a CIM Object Manager, it gets a reference to the CIM Object Manager, which it then uses to request services and operations.

System Booting

The `init.wbem` script is installed in the `/etc/init.d` directory.

Options The `init.wbem` function is implemented as a service management facility (SMF) method and is controlled by means of the `svcadm(1M)` command using the fault management resource identifier (FMRI) `svc:/application/management/wbem`.

The following options are supported through `svcadm(1M)`:

`start` Starts the CIMOM on the local host.

`stop` Stops the CIMOM.

`status` Gets the status of the CIMOM.

The SMF property `options/tcp_listen` is used to control whether CIMOM and the Solaris Management console server will respond to requests from remote systems.

The specification:

```
svc:/application/management/wbem/options/tcp_listen = true
```

...allows remote access and `false` disallows remote access. `false` is the default value.

Examples **EXAMPLE 1** Allowing Access to Remote Systems

The following commands enable CIMOM to allow access from remote systems.

```
# svccfg -s \
svc:/application/management/wbem setprop options/tcp_listen = true
# svcadm refresh svc:/application/management/wbem
```

Notes When the `init.wbem` script is invoked by SMF, it does not run the CIMOM directly. The server process is in Java and is too heavyweight to be run immediately at system boot time. Instead, three lightweight processes listen on three different ports that the CIMOM normally uses. This acts similarly to `inetd(1M)`.

Because Java programs cannot inherit file descriptors as other programs can, there is a small time period from when the first connection is made until the server is fully operational where client connections may be dropped. WBEM clients are immune to this, as they will retry until the server comes online.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWwbco

See Also `inetd(1M)`, `mofcomp(1M)`, `svccfg(1M)`, `svcadm(1M)`, `wbemadmin(1M)`, `wbemlogviewer(1M)`, `attributes(5)`, `smf(5)`

Name inityp2l – create NIS (YP) to LDAP configuration files

Synopsis

```
/usr/lib/netsvc/yp/inityp2l [-m mapping_file_name]
     [-c config_file_name]
```

Description The `inityp2l` utility assists with creation of the `NISLDAPmapping` and `ypserv` files. See `NISLDAPmapping(4)` and`ypserv(4)`. `inityp2l` examines the NIS maps on a system. and through a dialogue with the user, determines which NIS to (and from) LDAP mappings are required. A `NISLDAPmapping` file is then created based on this information. The utility asks users about their LDAP server configuration and a `ypserv` file is created based on this information.

The `inityp2l` utility handles mappings for standard NIS maps and the `auto.*` series of maps. If requested, it creates default mappings for custom maps, with each map entry represented as a single DIT string. `inityp2l` does not handle full custom mapping, but if requested, `inityp2l` will insert comments into the `NISLDAPmapping` file that indicate where these should be added.

To write to the `NISLDAPmapping` or `ypserv` files is potentially dangerous. `inityp2l` warns the user and asks for confirmation before:

1. it overwrites either file
2. it writes to the default `NISLDAPmapping` file location, if this file did not previously exist. This is important because the existence of a file in this location causes NIS components to work NIS to LDAP (N2L) mode when next restarted, rather than to traditional NIS mode.

`inityp2l` assists with rapid creation of a simple N2L configuration files. It is not a general purpose tool for the management of these files. An advanced user who would like to maintain the files or use custom mappings should examine the output of `inityp2l` and customize it by using a standard text editor.

Options `inityp2l` supports the following options:

`-c` Specify the name of the generated `ypserv` file. The default location is described in Files.

`-m` Specify the name of the generated `NISLDAPmapping` file. The default is described in Files.

Files

`/var/yp`	The directory to be searched for candidate domains (`/var/yp/*`) and NIS maps (`/var/yp/*/*`)
`/var/yp/NISLDAPmapping`	The default location for the generated `NISLDAPmapping` file
`/etc/default/ypserv`	The default location for the generated `ypserv` file

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	service/network/nis
Interface Stability	Obsolete

See Also NISLDAPmapping(4), ypserv(4), attributes(5)

Name in.lpd – BSD print protocol adaptor

Synopsis

```
/usr/lib/print/in.lpd
```

Description `in.lpd` implements the network listening service for the BSD print protocol specified in RFC 1179. The BSD print protocol provides a remote interface for systems to interact with a local spooling system. The protocol defines five standard requests from the client to the server: starting queue processing, transferring print jobs, retrieving terse status, retrieving verbose status, and canceling print jobs.

The `in.lpd` service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/network/lp
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. Responsibility for initiating and restarting this service is delegated to `inetd(1M)`. Use `inetadm(1M)` to make configuration changes and to view configuration information for this service. The service's status can be queried using the `svcs(1)` command.

`in.lpd` uses the `config/log_from_remote` property to allow or disallow remote access. The default value of this property, `localhost`, disallows remote access.

`inetd` waits for connections on TCP port 515. Upon receipt of a connect request, `in.lpd` is started to service the connection. Once the request has been filled, `in.lpd` closes the connection and exits.

Examples **EXAMPLE 1** Allowing Remote Access

The following command allows remote access to `in.lpd`.

```
# inetadm -m svc:/application/print/rfc1179:default bind_addr=""
```

Exit Status The following exit values are returned:

`0` Successful completion.

non-zero An error occurred.

Files `/usr/lib/print/bsd-adaptor/bsd_*.so*` Spooler translation modules.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	print/lp/print-client-commands

See Also svcs(1), inetadm(1M), inetd(1M), svcadm(1M), attributes(5), smf(5)

Name in.mpathd – IP multipathing daemon

Synopsis `/usr/lib/inet/in.mpathd`

Description The `in.mpathd` daemon performs failure and repair detection for IP interfaces that have been placed into an IPMP group (or optionally, for all IP interfaces on the system). It also controls which IP interfaces in an IPMP group are "active" (being used by the system to send or receive IP data traffic) in a manner that is consistent with the administrator's configured policy.

The `in.mpathd` daemon can detect IP interface failure and repair through three methods: by monitoring the `IFF_RUNNING` flag for each IP interface (link-state based failure detection), by sending and receiving ICMP probes on each IP interface (ICMP probe-based failure detection), or by transitive probing.

If transitive probing is enabled, IP interfaces are split into two classes: those that are eligible to receive inbound IP data traffic (see `INBOUND` in `ipmpstat(1M)`) and those that are not. Interfaces that are eligible to receive inbound IP data traffic detect failures by sending and receiving ICMP probes. Those that are not eligible to receive IP data traffic detect failures by exchanging link-layer ("transitive") probes with interfaces that are eligible.

By default, only link-state based failure detection is enabled. This requires the driver to support link-status notification. ICMP probe-based failure detection must be enabled through the configuration of one or more test addresses, which are described below. Transitive probing can be enabled by modifying the value of the SMF property (shown below) to `true`:

```
svc:/network/ipmp/config/transitive-probing
```

See `EXAMPLES` for more information on how to modify the value of the `transitive-probing` property.

Both ICMP and transitive probe-based failure detection methods test the entire IP interface send and receive path. The `ipmpstat(1M)` utility can be used to check which failure detection methods are enabled.

If `transitive-probing` is set to `true`, and no test addresses are configured for a given IPMP group, then transitive probing will be used. If it is set to `false` (default value), then transitive probing will not be used under any circumstance.

If only link-state based failure detection is enabled, then the health of the interface is determined solely from the state of the `IFF_RUNNING` flag. If probes have been enabled, the interface is considered failed if either link-state or probes indicate a failure, and repaired once the failure detection method has indicated the failure has been corrected. Although all interfaces in a group need not be configured with the same failure detection methods, transitive probing will be disabled on any interface of a group that has at least one IP test address.

As mentioned above, to perform ICMP probe-based failure detection, `in.mpathd` requires a test address on each IP interface for the purpose of sending and receiving probes. Any address created on an underlying interface with `ipadm(1M)` is automatically used as a test address. The

system will automatically set the NOFAILOVER flag on such addresses. Each address may be configured statically or acquired by means of DHCP. To find targets, in.mpathd first consults the routing table for routes on the same subnet, and uses the specified next-hop. If no routes match, it sends all-hosts ICMP probes and selects a subset of the systems that respond. Thus, for probe-based failure detection to operate, there must be at least one neighbor on each subnet that responds to ICMP echo request probes. The ipmpstat(1M) utility can be used to display both the current probe target information and the status of sent probes.

Both IPv4 and IPv6 are supported. If an IP interface is plumbed for IPv4 and an IPv4 test address is configured then in.mpathd will start sending ICMPv4 probes over that IP interface. Similarly, if an IP interface is plumbed for IPv6 and an IPv6 test address is configured, then in.mpathd will start sending ICMPv6 probes over that IP interface. However, note that in.mpathd will ignore IPv6 test addresses that are not link-local. If both IPv4 and IPv6 are plumbed, it is sufficient to configure only one of the two, that is, either an IPv4 test address or an IPv6 test address. If both IPv4 and IPv6 test addresses are configured, in.mpathd probes using both ICMPv4 and ICMPv6.

As mentioned above, in.mpathd also controls which IP interfaces in an IPMP group are "active" (used by the system to send and receive IP data traffic). Specifically, in.mpathd tracks the administrative configuration of each IPMP group and attempts to keep the number of active IP interfaces in each group consistent with that configuration. Therefore, if an active IP interface fails, in.mpathd will activate an INACTIVE interface in the group, provided one exists (it will prefer INACTIVE interfaces that are also marked STANDBY). Likewise, if an IP interface repairs and the resulting repair leaves the IPMP group with more active interfaces than the administrative configuration specifies, in.mpathd will deactivate one of the interfaces (preferably one marked STANDBY), except when the FAILBACK variable is used, as described below. Similar adjustments will be made by in.mpathd when offlining IP interfaces (for instance, in response to if_mpadm(1M)).

The in.mpathd daemon accesses three variable values in /etc/default/mpathd: FAILURE_DETECTION_TIME, FAILBACK and TRACK_INTERFACES_ONLY_WITH_GROUPS.

The FAILURE_DETECTION_TIME variable specifies the probe-based failure detection time. The shorter the failure detection time, the more probe traffic. The default value of FAILURE_DETECTION_TIME is 10 seconds. This means that IP interface failure will be detected by in.mpathd within 10 seconds. The IP interface repair detection time is always twice the value of FAILURE_DETECTION_TIME. Note that failures and repairs detected by link-based failure detection are acted on immediately, though in.mpathd may ignore link state changes if it suspects that the link state is flapping due to defective hardware; see DIAGNOSTICS.

By default, in.mpathd limits failure and repair detection to IP interfaces that are configured as part of a named IPMP group. Setting TRACK_INTERFACES_ONLY_WITH_GROUPS to no enables failure and repair detection on all IP interfaces, even if they are not part of a named IPMP group. IP interfaces that are tracked but not part of a named IPMP group are considered to be part of the "anonymous" IPMP group. In addition to having no name, this IPMP group is special in that its IP interfaces are not equivalent and thus cannot take over for one another in

the event of an IP interface failure. That is, the anonymous IPMP group can only be used for failure and repair detection, and provides no high-availability or load-spreading.

As described above, when `in.mpathd` detects that an IP interface has repaired, it activates it so that it will again be used to send and receive IP data traffic. However, if `FAILBACK` is set to `no`, then the IP interface will only be activated if no other active IP interfaces in the group remain. However, the interface may subsequently be activated if another IP interface in the group fails.

SMF Management

The `in.mpathd` daemon service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/network/ipmp:default
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`.

Examples

EXAMPLE 1 Enabling Fault Detection by Transitive Probing

The following example shows the sequence of SMF commands used to enable fault detection by transitive probing.

```
# svccfg -s svc:/network/ipmp setprop config/transitive-probing=true
# svcadm refresh svc:/network/ipmp:default
```

Files

`/etc/default/mpathd` Contains default values used by the `in.mpathd` daemon.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also

`if_mpadm(1M)`, `ifconfig(1M)`, `ipmpstat(1M)`, `svcadm(1M)`, `attributes(5)`, `smf(5)`, `icmp(7P)`, `icmp6(7P)`,

System Administration Guide: IP Services

Diagnostics

`IP interface` *interface_name* `has a hardware address which is not unique in group` *group_name*`; offlining`

Description: For probe-based failure detection, load-spreading, and other code IPMP features to work properly, each IP interface in an IPMP group must have a unique hardware address. If this requirement is not met, `in.mpathd` will automatically offline all but one of the IP interfaces with duplicate hardware addresses.

`IP interface` *interface_name* `now has a unique hardware address in group` *group_name*`; onlining`

Description: The previously-detected duplicate hardware address is now unique, and therefore `in.mpathd` has brought *interface_name* back online.

`Test address` *address* `is not unique in group; disabling probe-based failure detection on` *interface_name*

Description: For `in.mpathd` to perform probe-based failure detection, each test address in the group must be unique.

`No test address configured on interface` *interface_name* `disabling probe-based failure detection on it`

Description: For `in.mpathd` to perform probe-based failure detection on an IP interface, it must be configured with a test address: IPv4, IPv6, or both.

`IP` *interface_name* `in group` *group_name* `is not plumbed for IPv[4|6], affecting IPv[4|6] connectivity`

Description: All IP interfaces in a multipathing group must be homogeneously plumbed. For example, if one IP interface is plumbed for IPv4, then all IP interfaces in the group must be plumbed for IPv4, or IPv4 packets will not be able to be reliably sent and received. The STREAMS modules pushed on all IP interfaces must also be identical.

`The link has come up on` *interface_name* `more than 2 times in the last minute; disabling repair until it stabilizes.`

Description: To limit the impact of interfaces with intermittent hardware (such as a bad cable), `in.mpathd` will not consider an IP interface with a frequently changing link state as repaired until the link state stabilizes.

`Invalid failure detection time of` *time*`, assuming default 10000 ms`

Description: An invalid value was encountered for `FAILURE_DETECTION_TIME` in the `/etc/default/mpathd` file.

`Too small failure detection time of` *time*`, assuming minimum of 100 ms`

Description: The minimum value that can be specified for `FAILURE_DETECTION_TIME` is currently 100 milliseconds.

`Invalid value for FAILBACK` *value*

Description: Valid values for the boolean variable `FAILBACK` are `yes` or `no`.

`Invalid value for TRACK_INTERFACES_ONLY_WITH_GROUPS` *value*

Description: Valid values for the boolean variable `TRACK_INTERFACES_ONLY_WITH_GROUPS` are `yes` or `no`.

`Cannot meet requested failure detection time of` *time* `ms on (inet[6]` *interface_name*`) new failure detection time for group` *group_name* `is` *time* `ms`

Description: The round trip time for ICMP probes is higher than necessary to maintain the current failure detection time. The network is probably congested or the probe targets are loaded. `in.mpathd` automatically increases the failure detection time to whatever it can achieve under these conditions.

`Improved failure detection time` *time* `ms on (inet[6]` *interface_name*`) for group` *group_name*

Description: The round trip time for ICMP probes has now decreased and `in.mpathd` has lowered the failure detection time correspondingly.

`IP interface failure detected on` *interface_name*

Description: `in.mpathd` has detected a failure on *interface_name*, and has set the `IFF_FAILED` flag on *interface_name*, ensuring that it will not be used for IP data traffic.

`IP interface repair detected on` *interface_name*

Description: `in.mpathd` has detected a repair on *interface_name*, and has cleared the `IFF_FAILED` flag. Depending on the administrative configuration, the *interface_name* may again be used for IP data traffic.

`All IP interfaces in group` *group* `are now unusable`

Description: `in.mpathd` has determined that none of the IP interfaces in *group* can be used for IP data traffic, breaking network connectivity for the group.

`At least 1 IP interface (`*interface_name*`) in group` *group* `is now usable`

Description: `in.mpathd` has determined that at least one of the IP interfaces in *group* can again be used for IP data traffic, restoring network connectivity for the group.

`The link has gone down on` *interface_name*

Description: `in.mpathd` has detected that the `IFF_RUNNING` flag for *interface_name* has been cleared, indicating that the link has gone down.

`The link has come up on` *interface_name*

Description: `in.mpathd` has detected that the `IFF_RUNNING` flag for *interface_name* has been set, indicating that the link has come up.

Name in.ndpd – daemon for IPv6 autoconfiguration

Synopsis `/usr/lib/inet/in.ndpd [-adt] [-f` *config_file*`]`

Description `in.ndpd` provides both the host and router autoconfiguration components of Neighbor Discovery for IPv6 and Stateless and Stateful Address Autoconfiguration for IPv6. In particular, `in.ndpd` implements:

- router discovery;
- prefix discovery;
- parameter discovery;
- invocation of stateful address autoconfiguration;
- stateless address autoconfiguration; and
- privacy extensions for stateless address autoconfiguration.

Other aspects of Neighbor Discovery are implemented by `ip6(7P)`, including:

- address resolution;
- neighbor unreachability detection; and
- redirect.

The duplicate address detection function is implemented by the system kernel.

`in.ndpd` is managed by the service management facility (SMF), by means of the service identifier:

```
svc:/network/routing/ndp:default
```

If the `/etc/inet/ndpd.conf` file does not exist or does not set the variable `AdvSendAdvertisements` to true for a network interface, then `in.ndpd` will make the node a host for that interface, that is, sending router solicitation messages and then using router advertisement messages it receives to autoconfigure the node. Note that `in.ndpd` only autoconfigures the addresses of global or site-local scope from the prefix advertisement.

If `AdvSendAdvertisements` is set to true for an interface, then `in.ndpd` will perform router functions on that interface, that is, sending router advertisement messages to autoconfigure the attached hosts, but not use any advertisements it receives for autoconfiguration. However, when sending advertisements, `in.ndpd` will use the advertisements it sends itself to autoconfigure its prefixes.

Stateless autoconfiguration requires no manual configuration of hosts, minimal (if any) configuration of routers, and no additional servers. The stateless mechanism enables a host to generate its own addresses and uses local information as well as non-local information that is advertised by routers to generate the addresses. `in.ndpd` will plumb logical interfaces for each of these addresses.

Stateful autoconfiguration involves the `dhcpagent(1M)` daemon and the use of the DHCPv6 protocol. The `dhcpagent` daemon is responsible for plumbing the logical interfaces for the acquired addresses, maintaining the leases, and handling duplicate addresses. `in.ndpd` starts

the dhcpagent daemon automatically and signals when DHCPv6 should be started. in.ndpd also detects when dhcpagent configures the logical interfaces, and sets the appropriate prefix length on each according to received Routing Advertisement messages. in.ndpd will not stop dhcpagent; use ifconfig(1M) to control dhcpagent if necessary.

Temporary addresses that are autoconfigured for an interface can also be implemented. A temporary address token is enabled for one or more interfaces on a host. However, unlike standard, autoconfigured IPv6 addresses, a temporary address consists of the site prefix and a randomly generated 64 bit number. This random number becomes the interface ID segment of the IPv6 address. A link-local address is not generated with the temporary address as the interface ID.

If the kernel detects a duplicate temporary address, in.ndpd will automatically choose another.

Routers advertise all prefixes that have been assigned on the link. IPv6 hosts use Neighbor Discovery to obtain a subnet prefix from a local router. Hosts automatically create IPv6 addresses by combining the subnet prefix with an interface IDs that is generated from an interface's MAC address. In the absence of routers, a host can generate only link-local addresses. Link-local addresses can only be used for communication with nodes on the same link.

For information on how to enable IPv6 address autoconfiguration, see *System Administration Guide: IP Services*.

Options Supported options and equivalent SMF service properties are listed below. SMF service properties are set using a command of the form:

```
# routeadm -m ndp:default key=value
```

`-a`
Turn off stateless and stateful address auto configuration. When set, the daemon does not autoconfigure any addresses and does not renumber any addresses. This option does the same thing as the following lines in ndpd.conf(4):

```
ifdefault StatefulAddrConf off
ifdefault StatelessAddrConf off
```

Use of this option is equivalent to setting the stateless_addr_conf property to false.

`-d`
Turn on large amounts of debugging output on stdout. When set, the program runs in the foreground and stays attached to the controlling terminal. Use of this option is equivalent to setting the debug property to true.

`-f` *config_file*
Use *config_file* for configuration information instead of the default /etc/inet/ndpd.conf. Use of this option is equivalent to setting the config_file property to the configuration file to be used.

`-t`

Turn on tracing (printing) of all sent and received packets to `stdout`. When set, the program runs in the foreground and stays attached to the controlling terminal. As such, this option cannot be run under the SMF.

Files

`/etc/inet/ndpd.conf` Configuration file. This file is not necessary on a host, but it is required on a router to enable `in.ndpd` to advertise autoconfiguration information to the hosts.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also

`dhcpagent(1M)`, `ifconfig(1M)`, `routeadm(1M)`, `svcadm(1M)`, `ndpd.conf(4)`, `attributes(5)`, `icmp6(7P)`,`ip6(7P)`, `attributes(5)`, `smf(5)`

System Administration Guide: IP Services

Narten, T., Nordmark, E., Simpson, W.*RFC 2461, Neighbor Discovery for IP Version 6 (IPv6)*. The Internet Society. December 1998.

Thomson, S., Narten, T. *RFC 2462, IPv6 Stateless Address Autoconfiguration*. The Internet Society. December 1998.

Narten, T., and Draves, R. RFC 3041, Privacy Extensions for Stateless Address Autoconfiguration in IPv6. The Internet Society. January 2001.

Diagnostics

Receipt of a `SIGHUP` signal will make `in.ndpd` restart and reread `/etc/inet/ndpd.conf`.

Notes

The `in.ndpd` daemon service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/network/routing/ndp:default
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)` or `routeadm(1M)`.

Name in.rarpd, rarpd – DARPA Reverse Address Resolution Protocol server

Synopsis

```
/usr/sbin/in.rarpd [-d] -a

/usr/sbin/in.rarpd [-d] device unit
```

Description `in.rarpd` starts a daemon that responds to Reverse Address Resolution Protocol (RARP) requests. The daemon forks a copy of itself that runs in background. It must be run as root.

RARP is used by machines at boot time to discover their Internet Protocol (IP) address. The booting machine provides its Ethernet address in a RARP request message. Using the `ethers` and `hosts` databases, `in.rarpd` maps this Ethernet address into the corresponding IP address which it returns to the booting machine in an RARP reply message. The booting machine must be listed in both databases for `in.rarpd` to locate its IP address. `in.rarpd` issues no reply when it fails to locate an IP address.

`in.rarpd` uses the STREAMS-based Data Link Provider Interface (DLPI) message set to communicate directly with the datalink device driver.

Options The following options are supported:

`-a` Get the list of available network interfaces from IP using the SIOCGIFADDR ioctl and start a RARP daemon process on each interface returned.

`-d` Print assorted debugging messages while executing.

Examples

EXAMPLE 1 Starting An `in.rarpd` Daemon For Each Network Interface Name Returned From /dev/ip:

The following command starts an `in.rarpd` for each network interface name returned from `/dev/ip`:

```
example# /usr/sbin/in.rarpd -a
```

EXAMPLE 2 Starting An `in.rarpd` Daemon On The Device /dev/le With The Device Instance Number 0

The following command starts one `in.rarpd` on the device `/dev/le` with the device instance number `0`.

```
example# /usr/sbin/in.rarpd le 0
```

Files

`/etc/ethers` File or other source, as specified by `nsswitch.conf(4)`.

`/etc/hosts` File or other source, as specified by `nsswitch.conf(4)`.

`/tftpboot`

`/dev/ip`

`/dev/arp`

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/boot/network

See Also svcs(1), boot(1M), ifconfig(1M), svcadm(1M), ethers(4), hosts(4), netconfig(4), nsswitch.conf(4),attributes(5), smf(5), dlpi(7P)

Finlayson, R., Mann, T., Mogul, J., and Theimer, M., *RFC 903, A Reverse Address Resolution Protocol*, Network Information Center, SRI International, June 1984.

Notes The in.rarpd service is managed by the service management facility, smf(5), under the service identifier:

```
svc:/network/rarp
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using svcadm(1M). The service's status can be queried using the svcs(1) command.

Name in.rdisc, rdisc – network router discovery daemon

Synopsis

```
/usr/sbin/in.rdisc  [-a] [-f] [-s] [send-address] [receive-address]

/usr/sbin/in.rdisc  -r [-p preference] [-T interval]
 [send-address] [receive-address]
```

Description `in.rdisc` remains part of the software distribution of the Solaris Operating Environment. It is, however, not used by default. `in.routed(1M)` includes the functionality provided by `in.rdisc`. See `routeadm(1M)` for details of how to specify the IPV4 routing daemon.

`in.rdisc` implements the ICMP router discovery protocol. The first form of the command is used on hosts and the second form is used on routers.

`in.rdisc` can be invoked in either the first form (host mode) or second form (router mode).

On a host, `in.rdisc` populates the network routing tables with default routes. On a router, advertises the router to all the hosts.

`in.rdisc` is managed by the service management facility (SMF), by means of the service identifier:

```
svc:/network/routing/rdisc:default
```

Host (First Form) On a host, `in.rdisc` listens on the `ALL_HOSTS` (224.0.0.1) multicast address for `ROUTER_ADVERTISE` messages from routers. The received messages are handled by first ignoring those listed router addresses with which the host does not share a network. Among the remaining addresses, the ones with the highest preference are selected as default routers and a default route is entered in the kernel routing table for each one of them.

Optionally, `in.rdisc` can avoid waiting for routers to announce themselves by sending out a few `ROUTER_SOLICITATION` messages to the `ALL_ROUTERS` (224.0.0.2) multicast address when it is started.

A timer is associated with each router address. The address will no longer be considered for inclusion in the routing tables if the timer expires before a new *advertise* message is received from the router. The address will also be excluded from consideration if the host receives an *advertise* message with the preference being maximally negative or with a lifetime of zero.

Router (Second Form) When `in.rdisc` is started on a router, it uses the `SIOCGIFCONF` `ioctl(2)` to find the interfaces configured into the system and it starts listening on the `ALL_ROUTERS` multicast address on all the interfaces that support multicast. It sends out *advertise* messages to the `ALL_HOSTS` multicast address advertising all its `IP` addresses. A few initial *advertise* messages are sent out during the first 30 seconds and after that it will transmit *advertise* messages approximately every 600 seconds.

When `in.rdisc` receives a *solicitation* message, it sends an *advertise* message to the host that sent the *solicitation* message.

When `in.rdisc` is terminated by a signal, it sends out an *advertise* message with the preference being maximally negative.

Options Supported options and equivalent SMF service properties are listed below. SMF service properties are set using a command of the form:

```
# routeadm -m rdisc:default key=value
```

`-a` Accept all routers independent of the preference they have in their *advertise* messages. Normally, `in.rdisc` only accepts (and enters in the kernel routing tables) the router or routers with the highest preference. Use of this option is equivalent to setting the `accept_all` property to true.

`-f` Run `in.rdisc` forever even if no routers are found. Normally, `in.rdisc` gives up if it has not received any *advertise* message after soliciting three times, in which case it exits with a non-zero exit code. If `-f` is not specified in the first form then `-s` must be specified. For SMF execution, this option is required.

`-r` Act as a router, rather than a host. Use of this option is equivalent to setting the `act_as_router` property to true.

`-s` Send three *solicitation* messages initially to quickly discover the routers when the system is booted. When `-s` is specified, `in.rdisc` exits with a non-zero exit code if it can not find any routers. This can be overridden with the `-f` option. This option is not compatible with SMF execution and is not supported for the `rdisc` service.

`-p` *preference* Set the preference transmitted in the *solicitation* messages. The default is zero. Use of this option is equivalent to setting the `preference` property.

`-T` *interval* Set the interval between transmitting the *advertise* messages. The default time is 600 seconds. Use of this option is equivalent to setting the `transmit_interval` property.

The `send-address` and `receive-address` daemon options can be set by means of the `send_address` and `receive_address` properties.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/network/routing

See Also `in.routed(1M)`, `routeadm(1M)`, `svcadm(1M)`, `ioctl(2)`, `gateways(4)`, `attributes(5)`, `smf(5)`, `icmp(7P)`, `inet(7P)`

Deering, S.E., editor, *ICMP Router Discovery Messages*, RFC 1256, Network Information Center, SRI International, Menlo Park, California, September 1991.

Name in.rexecd, rexecd – remote execution server

Synopsis

```
in.rexecd
```

Description `in.rexecd` is the server for the `rexec(3SOCKET)` routine. The server provides remote execution facilities with authentication based on user names and passwords. It is invoked automatically as needed by `inetd(1M)`, and then executes the following protocol:

1. The server reads characters from the socket up to a null (`\0`) byte. The resultant string is interpreted as an ASCII number, base 10.
2. If the number received in step 1 is non-zero, it is interpreted as the port number of a secondary stream to be used for the `stderr`. A second connection is then created to the specified port on the client's machine.
3. A null terminated user name of at most 16 characters is retrieved on the initial socket.
4. A null terminated password of at most 16 characters is retrieved on the initial socket.
5. A null terminated command to be passed to a shell is retrieved on the initial socket. The length of the command is limited by the upper bound on the size of the system's argument list.
6. `rexecd` then validates the user as is done at login time and, if the authentication was successful, changes to the user's home directory, and establishes the user and group protections of the user. If any of these steps fail the connection is aborted and a diagnostic message is returned.
7. A null byte is returned on the connection associated with the `stderr` and the command line is passed to the normal login shell of the user. The shell inherits the network connections established by `rexecd`.

Usage `in.rexecd` and `rexecd` are IPv6–enabled. See `ip6(7P)`.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	service/network/network-servers

See Also `svcs(1)`, `inetd(1M)`, `inetadm(1M)`, `svcadm(1M)`, `rexec(3SOCKET)`, `attributes(5)`, `smf(5)`, `ip6(7P)`

Diagnostics All diagnostic messages are returned on the connection associated with the `stderr`, after which any network connections are closed. An error is indicated by a leading byte with a value of 1 (0 is returned in step 7 above upon successful completion of all the steps prior to the command execution).

`username too long` The name is longer than 16 characters.

`password too long` The password is longer than 16 characters.

`command too long`	The command line passed exceeds the size of the argument list (as configured into the system).
`Login incorrect.`	No password file entry for the user name existed.
`No remote directory.`	The `chdir` command to the home directory failed.
`Try again.`	A `fork` by the server failed.
`/usr/bin/sh: ...`	The user's login shell could not be started.

Notes The `in.rexecd` service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/network/rexec:default
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. Responsibility for initiating and restarting this service is delegated to `inetd(1M)`. Use `inetadm(1M)` to make configuration changes and to view configuration information for this service. The service's status can be queried using the `svcs(1)` command.

Name in.ripngd – network routing daemon for IPv6

Synopsis
```
/usr/lib/inet/in.ripngd  [-s] [-q] [-t] [-p n] [-P] [-v ]
  [logfile]
```

Description `in.ripngd` is the IPv6 equivalent of `in.routed(1M)`. It is invoked at boot time to manage the network routing tables. The routing daemon uses the Routing Information Protocol for IPv6.

`in.ripngd` is managed by the service management facility (SMF), by means of the service identifier:

```
svc:/network/routing/ripng:default
```

In normal operation, `in.ripngd` listens on the `udp(7P)` socket port 521 for routing information packets. If the host is an internetwork router, it periodically supplies copies of its routing tables to any directly connected hosts and networks.

When `in.ripngd` is started, it uses the `SIOCGLIFCONF` `ioctl(2)` to find those directly connected IPv6 interfaces configured into the system and marked "up"; the software loopback interface is ignored. If multiple interfaces are present, it is assumed the host will forward packets between networks. `in.ripngd` then multicasts a request packet on each IPv6 interface and enters a loop, listening for request and response packets from other hosts.

When a request packet is received, `in.ripngd` formulates a reply based on the information maintained in its internal tables. The response packet contains a list of known routes. With each route is a number specifying the number of bits in the prefix. The prefix is the number of bits in the high order part of an address that indicate the subnet or network that the route describes. Each route reported also has a "*hop count*" metric. A count of 16 or greater is considered "infinity." The metric associated with each route returned provides a metric relative to the sender.

The request packets received by `in.ripngd` are used to update the routing tables if one of the following conditions is satisfied:

- No routing table entry exists for the destination network or host, and the metric indicates the destination is "reachable", that is, the *hop count* is not infinite.
- The source host of the packet is the same as the router in the existing routing table entry. That is, updated information is being received from the very internetwork router through which packets for the destination are being routed.
- The existing entry in the routing table has not been updated for a period of time, defined to be 90 seconds, and the route is at least as cost-effective as the current route.
- The new route describes a shorter route to the destination than the one currently stored in the routing tables; this is determined by comparing the metric of the new route against the one stored in the table.

When an update is applied, `in.ripngd` records the change in its internal tables and generates a response packet to all directly connected hosts and networks. To allow possible unstable situations to settle, `in.ripngd` waits a short period of time (no more than 30 seconds) before modifying the kernel's routing tables.

In addition to processing incoming packets, `in.ripngd` also periodically checks the routing table entries. If an entry has not been updated for 3 minutes, the entry's metric is set to infinity and marked for deletion. Deletions are delayed an additional 60 seconds to insure the invalidation is propagated throughout the internet.

Hosts acting as internetwork routers gratuitously supply their routing tables every 30 seconds to all directly connected hosts and networks.

Options `in.ripngd` supports the options listed below. Listed with the options are the equivalent SMF property values. These are set for the `ripng:default` service with a command of the form:

```
# routeadm -m ripng:default key=value
```

`-p` *n* Send and receive the routing packets from other routers using the UDP port number *n*. Use of this option is equivalent to setting the `udp_port` property.

`-P` Do not use poison reverse. Use of this option is equivalent to setting the `poison_reverse` property to false.

`-q` Do not supply routing information. Use of this option is equivalent to setting the `quiet_mode` property to true.

`-s` Force `in.ripngd` to supply routing information whether it is acting as an internetwork router or not. Use of this option is equivalent to setting the `supply_routes` property to true.

`-t` Print all packets sent or received to standard output. `in.ripngd` will not divorce itself from the controlling terminal. Accordingly, interrupts from the keyboard will kill the process. Not supported by the `ripng` service.

`-v` Print all changes made to the routing tables to standard output with a timestamp. Use of this option is equivalent to setting the `verbose` property to true.

Any other argument supplied to this option is interpreted as the name of the file in which the actions of `in.ripngd`, as specified by this option or by `-t`, should be logged instead of being sent to standard output.

The logfile can be specified for the `ripng` service by means of the `log_file` property.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/network/routing

See Also `in.routed`(1M), `routeadm`(1M), `svcadm`(1M), `ioctl`(2), `attributes`(5), `smf`(5), `udp`(7P)

G. Malkin, R. Minnear, *RFC 2080, RIPng for IPv6*, January 1997.

Notes The kernel's routing tables may not correspond to those of `in.ripngd` for short periods of time while processes that utilize existing routes exit; the only remedy for this is to place the routing process in the kernel.

`in.ripngd` currently does not support all of the functionality of `in.routed`(1M). Future releases may support more if appropriate.

`in.ripngd` initially obtains a routing table by examining the interfaces configured on a machine. It then sends a request on all directly connected networks for more routing information. `in.ripngd` does not recognize or use any routing information already established on the machine prior to startup. With the exception of interface changes, `in.ripngd` does not see any routing table changes that have been done by other programs on the machine, for example, routes added, deleted or flushed by means of the `route`(1M) command. Therefore, these types of changes should not be done while `in.ripngd` is running. Rather, shut down `in.ripngd`, make the changes required, and then restart `in.ripngd`.

Name in.rlogind, rlogind – remote login server

Synopsis

```
/usr/sbin/in.rlogind [-k5eExXciPp] [-s tos] [-S keytab]
     [-M realm]
```

Description `in.rlogind` is the server for the `rlogin(1)` program. The server provides a remote login facility with authentication based on Kerberos V5 or privileged port numbers.

`in.rlogind` is invoked by `inetd(1M)` when a remote login connection is established. When Kerberos V5 authentication is required (see option `-k` below), the authentication sequence is as follows:

- Check Kerberos V5 authentication.
- Check authorization according to the rules in `krb5_auth_rules(5)`.
- Prompt for a password if any checks fail and the PAM configuration (see `pam.conf(4)`) is configured to do so.

In order for Kerberos authentication to work, a `host/`<*FQDN*> Kerberos principal must exist for each Fully Qualified Domain Name associated with the `in.rlogind` server. Each of these `host/`<*FQDN*> principals must have a `keytab` entry in the `/etc/krb5/krb5.keytab` file on the `in.rlogind` server. An example principal might be:

```
host/bigmachine.eng.example.com
```

See `kadmin(1M)` or `gkadmin(1M)` for instructions on adding a principal to a `krb5.keytab` file. See *Oracle Solaris 11.1 Administration: Security Services* for a discussion of Kerberos authentication.

If Kerberos V5 authentication is not enabled, then the authentication procedure follows the standard `rlogin` protocol:

- The server checks the client's source port. If the port is not in the range 512-1023, the server aborts the connection.
- The server checks the client's source address. If an entry for the client exists in both `/etc/hosts` and `/etc/hosts.equiv`, a user logging in from the client is not prompted for a password. If the address is associated with a host for which no corresponding entry exists in `/etc/hosts`, the user is prompted for a password, regardless of whether or not an entry for the client is present in `/etc/hosts.equiv`. See `hosts(4)` and `hosts.equiv(4)`.

Once the source port and address have been checked, `in.rlogind` allocates a pseudo-terminal and manipulates file descriptors so that the slave half of the pseudo-terminal becomes the `stdin`, `stdout`, and `stderr` for a login process. The login process is an instance of the `login(1)` program, invoked with the `-r`.

The login process then proceeds with the `pam(3PAM)` authentication process. See `SECURITY` below. If automatic authentication fails, it reprompts the user to login.

The parent of the login process manipulates the master side of the pseudo-terminal, operating as an intermediary between the login process and the client instance of the `rlogin` program. In normal operation, a packet protocol is invoked to provide Ctrl-S and Ctrl-Q type facilities and propagate interrupt signals to the remote programs. The login process propagates the client terminal's baud rate and terminal type, as found in the environment variable, `TERM`.

Options The following options are supported:

`-5` Same as `-k`, for backwards compatibility.

`-c` Requires Kerberos V5 clients to present a cryptographic checksum of initial connection information like the name of the user that the client is trying to access in the initial authenticator. This checksum provides additionl security by preventing an attacker from changing the initial connection information. This option is mutually exclusive with the `-i` option.

`-e` Creates an encrypted session.

`-E` Same as `-e`, for backwards compatibility.

`-i` Ignores authenticator checksums if provided. This option ignores authenticator checksums presented by current Kerberos clients to protect initial connection information. Option `-i` is the opposite of option `-c`.

`-k` Allows Kerberos V5 authentication with the `.k5login` access control file to be trusted. If this authentication system is used by the client and the authorization check is passed, then the user is allowed to log in.

`-M` *realm* Uses the indicated Kerberos V5 realm. By default, the daemon will determine its realm from the settings in the `krb5.conf(4)` file.

`-p` Prompts for authentication only if other authentication checks fail.

`-P` Prompts for a password in addition to other authentication methods.

`-s` *tos* Sets the `IP TOS` option.

`-S` *keytab* Sets the `KRB5` keytab file to use. The `/etc/krb5/krb5.keytab` file is used by default.

`-x` Same as `-e`, for backwards compatibility.

`-X` Same as `-e`, for backwards compatibility.

Usage `rlogind` and `in.rlogind` are IPv6–enabled. See `ip6(7P)`. `IPv6` is not currently supported with Kerberos V5 authentication.

Typically, Kerberized `rlogin` service runs on port 543 (klogin) and Kerberized, encrypted `rlogin` service runs on port 2105 (eklogin). The corresponding FMRI entries are:

```
svc:/network/login:klogin (rlogin with kerberos)
svc:/network/login:eklogin (rlogin with kerberos and encryption)
```

Security `in.rlogind` uses pam(3PAM) for authentication, account management, and session management. The PAM configuration policy, configured in `/etc/pam.conf` or per-service files in `/etc/pam.d/`, specifies the modules to be used for `in.rlogind`. Here is a partial `pam.conf` file with entries for the `rlogin` command using the `rhosts` and UNIX authentication modules, and the UNIX account, session management, and password management modules.

```
rlogin          auth sufficient     pam_rhosts_auth.so.1
rlogin          auth requisite      pam_authtok_get.so.1
rlogin          auth required       pam_dhkeys.so.1
rlogin          auth required       pam_unix_auth.so.1

rlogin          account required    pam_unix_roles.so.1
rlogin          account required    pam_unix_projects.so.1
rlogin          account required    pam_unix_account.so.1

rlogin          session required    pam_unix_session.so.1
```

The equivalent PAM configuration using `/etc/pam.d/` would be the following entries in `/etc/pam.d/rlogin`:

```
auth sufficient    pam_rhosts_auth.so.1
auth requisite     pam_authtok_get.so.1
auth required      pam_dhkeys.so.1
auth required      pam_unix_auth.so.1

account required   pam_unix_roles.so.1
account required   pam_unix_projects.so.1
account required   pam_unix_account.so.1

session required   pam_unix_session.so.1
```

With this configuration, the server checks the client's source address. If an entry for the client exists in both `/etc/hosts` and `/etc/hosts.equiv`, a user logging in from the client is not prompted for a password. If the address is associated with a host for which no corresponding entry exists in `/etc/hosts`, the user is prompted for a password, regardless of whether or not an entry for the client is present in `/etc/hosts.equiv`. See `hosts(4)` and `hosts.equiv(4)`.

When running a Kerberized rlogin service (with or without the encryption option), the pam service name that should be used is "`krlogin`".

If there are no entries for the `rlogin` service, then the entries for the "other" service will be used. If multiple authentication modules are listed, then the user may be prompted for multiple passwords. Removing the `pam_rhosts_auth.so.1` entry will disable the `/etc/hosts.equiv` and `~/.rhosts` authentication protocol and the user would always be forced to type the password. The *sufficient* flag indicates that authentication through the `pam_rhosts_auth.so.1` module is sufficient to authenticate the user. Only if this authentication fails is the next authentication module used.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	service/network/network-servers

See Also `login(1)`, `svcs(1)`, `rlogin(1)`, `gkadmin(1M)`, `in.rshd(1M)`, `inetadm(1M)`, `inetd(1M)`, `kadmin(1M)`, `svcadm(1M)`, `pam(3PAM)`, `hosts(4)`, `hosts.equiv(4)`, `krb5.conf(4)`, `pam.conf(4)`, `attributes(5)`, `environ(5)`, `krb5_auth_rules(5)`, `pam_authtok_check(5)`, `pam_authtok_get(5)`, `pam_authtok_store(5)`, `pam_dhkeys(5)`, `pam_passwd_auth(5)`, `pam_unix_account(5)`, `pam_unix_auth(5)`, `pam_unix_session(5)`, `smf(5)`

Oracle Solaris 11.1 Administration: Security Services

Diagnostics All diagnostic messages are returned on the connection associated with the `stderr`, after which any network connections are closed. An error is indicated by a leading byte with a value of 1.

`Hostname for your address unknown.`	No entry in the host name database existed for the client's machine.
`Try again.`	A *fork* by the server failed.
`/usr/bin/sh:...`	The user's login shell could not be started.

Notes The authentication procedure used here assumes the integrity of each client machine and the connecting medium. This is insecure, but it is useful in an "open" environment.

A facility to allow all data exchanges to be encrypted should be present.

The `in.rlogind` service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/network/login:rlogin (rlogin)
svc:/network/login:klogin (rlogin with kerberos)
svc:/network/login:eklogin (rlogin with kerberos and encryption)
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. Responsibility for initiating and restarting this service is

delegated to inetd(1M). Use inetadm(1M) to make configuration changes and to view configuration information for this service. The service's status can be queried using the svcs(1) command.

Name in.routed, routed – network routing daemon

Synopsis

```
/usr/sbin/in.routed [-AdghmnqsStVz] [-T tracefile [-v]]
 [-F net[/mask ][,metric]] [-P params]
```

Description The daemon `in.routed`, often referred to as `routed`, is invoked at boot time to manage the network routing tables. It uses Routing Information Protocol, RIPv1 (RFC 1058), RIPv2 (RFC 2453), and Internet Router Discovery Protocol (RFC 1256) to maintain the kernel routing table. The RIPv1 protocol is based on the reference 4.3BSD daemon.

`in.routed` is managed by means of the service management facility (SMF), using the fault management resource identifier (FMRI):

```
svc:/network/routing/route:default
```

The daemon listens on a `udp` socket for the `route` service (see `services(4)`) for Routing Information Protocol packets. It also sends and receives multicast Router Discovery ICMP messages. If the host is a router, `in.routed` periodically supplies copies of its routing tables to any directly connected hosts and networks. It also advertises or solicits default routes using Router Discovery ICMP messages.

When started (or when a network interface is later turned on), `in.routed` uses an `AF_ROUTE` address family facility to find those directly connected interfaces configured into the system and marked "up". It adds necessary routes for the interfaces to the kernel routing table. Soon after being first started, and provided there is at least one interface on which RIP has not been disabled, `in.routed` deletes all pre-existing non-static routes in the kernel table. Static routes in the kernel table are preserved and included in RIP responses if they have a valid RIP metric (see `route(1M)`).

If more than one interface is present (not counting the loopback interface), it is assumed that the host should forward packets among the connected networks. After transmitting a RIP request and Router Discovery Advertisements or Solicitations on a new interface, the daemon enters a loop, listening for RIP request and response and Router Discovery packets from other hosts.

When a request packet is received, `in.routed` formulates a reply based on the information maintained in its internal tables. The response packet generated contains a list of known routes, each marked with a "hop count" metric (a count of 16 or greater is considered "infinite"). Advertised metrics reflect the metric associated with an interface (see `ifconfig(1M)`), so setting the metric on an interface is an effective way to steer traffic.

Responses do not include routes with a first hop on the requesting network, to implement in part split-horizon. Requests from query programs such as `rtquery(1M)` are answered with the complete table.

The routing table maintained by the daemon includes space for several gateways for each destination to speed recovery from a failing router. RIP response packets received are used to

update the routing tables, provided they are from one of the several currently recognized gateways or advertise a better metric than at least one of the existing gateways.

When an update is applied, `in.routed` records the change in its own tables and updates the kernel routing table if the best route to the destination changes. The change in the kernel routing table is reflected in the next batch of response packets sent. If the next response is not scheduled for a while, a flash update response containing only recently changed routes is sent.

In addition to processing incoming packets, in.routed also periodically checks the routing table entries. If an entry has not been updated for 3 minutes, the entry's metric is set to infinity and marked for deletion. Deletions are delayed until the route has been advertised with an infnite metric to insure the invalidation is propagated throughout the local internet. This is a form of poison reverse.

Routes in the kernel table that are added or changed as a result of ICMP Redirect messages are deleted after a while to minimize black-holes. When a TCP connection suffers a timeout, the kernel tells `in.routed`, which deletes all redirected routes through the gateway involved, advances the age of all RIP routes through the gateway to allow an alternate to be chosen, and advances of the age of any relevant Router Discovery Protocol default routes.

Hosts acting as internetwork routers gratuitously supply their routing tables every 30 seconds to all directly connected hosts and networks. These RIP responses are sent to the broadcast address on nets that support broadcasting, to the destination address on point-to-point links, and to the router's own address on other networks. If RIPv2 is enabled, multicast packets are sent on interfaces that support multicasting.

If no response is received on a remote interface, if there are errors while sending responses, or if there are more errors than input or output (see `netstat(1M)`), then the cable or some other part of the interface is assumed to be disconnected or broken, and routes are adjusted appropriately.

The Internet Router Discovery Protocol is handled similarly. When the daemon is supplying RIP routes, it also listens for Router Discovery Solicitations and sends Advertisements. When it is quiet and listening to other RIP routers, it sends Solicitations and listens for Advertisements. If it receives a good Advertisement and it is not multi-homed, it stops listening for broadcast or multicast RIP responses. It tracks several advertising routers to speed recovery when the currently chosen router dies. If all discovered routers disappear, the daemon resumes listening to RIP responses. It continues listening to RIP while using Router Discovery if multi-homed to ensure all interfaces are used.

The Router Discovery standard requires that advertisements have a default “lifetime” of 30 minutes. That means should something happen, a client can be without a good route for 30 minutes. It is a good idea to reduce the default to 45 seconds using `-P rdisc_interval=45` on the command line or `rdisc_interval=45` in the `/etc/gateways` file. See `gateways(4)`.

While using Router Discovery (which happens by default when the system has a single network interface and a Router Discover Advertisement is received), there is a single default route and a variable number of redirected host routes in the kernel table. On a host with more than one network interface, this default route will be via only one of the interfaces. Thus, multi-homed hosts running with `-q` might need the `no_rdisc` argument described below.

To support "legacy" systems that can handle neither RIPv2 nor Router Discovery, you can use the `pm_rdisc` parameter in the `/etc/gateways`. See `gateways(4)`.

By default, neither Router Discovery advertisements nor solicitations are sent over point-to-point links (for example, PPP). The Solaris OE uses a netmask of all ones (255.255.255.255) on point-to-point links.

`in.routed` supports the notion of "distant" passive or active gateways. When the daemon is started, it reads the file `/etc/gateways` to find such distant gateways that cannot be located using only information from a routing socket, to discover if some of the local gateways are passive, and to obtain other parameters. Gateways specified in this manner should be marked passive if they are not expected to exchange routing information, while gateways marked active should be willing to exchange RIP packets. Routes through passive gateways are installed in the kernel's routing tables once upon startup and are not included in transmitted RIP responses.

Distant active gateways are treated like network interfaces. RIP responses are sent to the distant active gateway. If no responses are received, the associated route is deleted from the kernel table and RIP responses are advertised via other interfaces. If the distant gateway resumes sending RIP responses, the associated route is restored.

Distant active gateways can be useful on media that do not support broadcasts or multicasts but otherwise act like classic shared media, such as some ATM networks. One can list all RIP routers reachable on the HIPPI or ATM network in `/etc/gateways` with a series of "host" lines. Note that it is usually desirable to use RIPv2 in such situations to avoid generating lists of inferred host routes.

Gateways marked external are also passive, but are not placed in the kernel routing table, nor are they included in routing updates. The function of external entries is to indicate that another routing process will install such a route if necessary, and that other routes to that destination should not be installed by `in.routed`. Such entries are required only when both routers might learn of routes to the same destination.

Options Listed below are available options. Any other argument supplied is interpreted as the name of a file in which the actions of `in.routed` should be logged. It is better to use `-T` (described below) instead of appending the name of the trace file to the command. Associated SMF properties for these options are described, and can be set by means of a command of the form:

```
# routeadm -m route:default name=value
```

`-A`

Do not ignore RIPv2 authentication if we do not care about RIPv2 authentication. This option is required for conformance with RFC 2453. However, it makes no sense and breaks using RIP as a discovery protocol to ignore all RIPv2 packets that carry authentication when this machine does not care about authentication. This option is equivalent to setting the `ignore_auth` property value to false.

`-d`

Do not run in the background. This option is meant for interactive use and is not usable under the SMF.

`-F net[/`*mask*`][,`*metric*`]`

Minimize routes in transmissions via interfaces with addresses that match *net* (network number)/*mask* (netmask), and synthesizes a default route to this machine with the *metric*. The intent is to reduce RIP traffic on slow, point-to-point links, such as PPP links, by replacing many large UDP packets of RIP information with a single, small packet containing a "fake" default route. If *metric* is absent, a value of 14 is assumed to limit the spread of the "fake" default route. This is a dangerous feature that, when used carelessly, can cause routing loops. Notice also that more than one interface can match the specified network number and mask. See also `-g`. Use of this option is equivalent to setting the `minimize_routes` property.

`-g`

Used on internetwork routers to offer a route to the "default" destination. It is equivalent to `-F 0/0,1` and is present mostly for historical reasons. A better choice is `-P pm_rdisc` on the command line or `pm_rdisc` in the `/etc/gateways` file. A larger metric will be used with the latter alternatives, reducing the spread of the potentially dangerous default route. The `-g` (or `-P`) option is typically used on a gateway to the Internet, or on a gateway that uses another routing protocol whose routes are not reported to other local routers. Note that because a metric of 1 is used, this feature is dangerous. Its use more often creates chaos with a routing loop than solves problems. Use of this option is equivalent to setting the `offer_default_route` property to true.

`-h`

Causes host or point-to-point routes not to be advertised, provided there is a network route going the same direction. That is a limited kind of aggregation. This option is useful on gateways to LANs that have other gateway machines connected with point-to-point links such as SLIP. Use of this option is equivalent to setting the `advertise_host_routes` property to false.

`-m`

Cause the machine to advertise a host or point-to-point route to its primary interface. It is useful on multi-homed machines such as NFS servers. This option should not be used except when the cost of the host routes it generates is justified by the popularity of the server. It is effective only when the machine is supplying routing information, because there is more than one interface. The `-m` option overrides the `-q` option to the limited extent

of advertising the host route. Use of this option is equivalent to setting the `advertise_host_routes_primary` property to true.

`-n`
Do not install routes in kernel. By default, routes are installed in the kernel. Use of this option is equivalent to setting the `install_routes` property to false.

`-P` *params*
Equivalent to adding the parameter line *params* to the `/etc/gateways` file. Can also be set by means of the `parameters` property.

`-q`
Opposite of the `-s` option. This is the default when only one interface is present. With this explicit option, the daemon is always in "quiet mode" for RIP and does not supply routing information to other computers. Use of this option is equivalent to setting the `quiet_mode` property to true.

`-s`
Force `in.routed` to supply routing information. This is the default if multiple network interfaces are present on which RIP or Router Discovery have not been disabled, and if global IPv4 forwarding is turned on (by means of `ipadm`(1M)). Use of this option is equivalent to setting the `supply_routes` property to true.

`-S`
If `in.routed` is not acting as an internetwork router, instead of entering the whole routing table in the kernel, it enters only a default route for each internetwork router. This reduces the memory requirements without losing any routing reliability. This option is provided for compatibility with the previous, RIPv1–only `in.routed`. Use of this option is generally discouraged. Use of this option is equivalent to setting the `default_routes_only` property to true.

`-t`
Runs in the foreground (as with `-d`) and logs the contents of the packets received (as with `-zz`). This is for compatibility with prior versions of Solaris and has no SMF equivalent.

`-T` *tracefile*
Increases the debugging level to at least 1 and causes debugging information to be appended to the trace file. Because of security concerns, do not to run `in.routed` routinely with tracing directed to a file. Use of this option is equivalent to setting the `log_file` property to `trace file path`.

`-v`
Enables debug. Similar to `-z`, except, where `-z` increments `trace_level`, `-v` sets `trace_level` to 1. Also, `-v` requires the `-T` option. Use of this option is equivalent to setting the `debug` property to true.

`-V`
Displays the version of the daemon.

`-z`
Increase the debugging level, which causes more information to be logged on the tracefile specified with `-T` or stdout. The debugging level can be increased or decreased with the `SIGUSR1` or `SIGUSR2` signals or with the `rtquery(1M)` command.

Files

`/etc/defaultrouter`	If this file is present and contains the address of a default router, the system startup script does not run `in.routed`. See `defaultrouter(4)`.
`/etc/gateways`	List of distant gateways and general configuration options for `in.routed`. See `gateways(4)`.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/network/routing

See Also `ipadm(1M)`, `route(1M)`, `routeadm(1M)`, `rtquery(1M)`, `svcadm(1M)`, `ioctl(2)`, `inet(3SOCKET)`, `defaultrouter(4)`, `gateways(4)`, `attributes(5)`, `icmp(7P)`, `inet(7P)`, `udp(7P)`

Internet Transport Protocols, XSIS 028112, Xerox System Integration Standard

Routing Information Protocol, v2 (RFC 2453, STD 0056, November 1998)

RIP-v2 MD5 Authentication (RFC 2082, January 1997)

Routing Information Protocol, v1 (RFC 1058, June 1988)

ICMP Router Discovery Messages (RFC 1256, September 1991)

Notes In keeping with its intended design, this daemon deviates from RFC 2453 in two notable ways:

- By default, `in.routed` does not discard authenticated RIPv2 messages when RIP authentication is not configured. There is little to gain from dropping authenticated packets when RIPv1 listeners will gladly process them. Using the `-A` option causes `in.routed` to conform to the RFC in this case.
- Unauthenticated RIP requests are never discarded, even when RIP authentication is configured. Forwarding tables are not secret and can be inferred through other means such as test traffic. RIP is also the most common router-discovery protocol, and hosts need to send queries that will be answered.

`in.routed` does not always detect unidirectional failures in network interfaces, for example, when the output side fails.

Name in.rshd, rshd – remote shell server

Synopsis

```
in.rshd [-k5eciU] [-s tos] [-S keytab] [-M realm]
     [-L env_var] host.port
```

Description `in.rshd` is the server for the `rsh(1)` program. The server provides remote execution facilities with authentication based on Kerberos V5 or privileged port numbers.

`in.rshd` is invoked by `inetd(1M)` each time a shell service is requested.

When Kerberos V5 authentication is required (this can be set with Kerberos-specific options listed below), the following protocol is initiated:

1. Check Kerberos V5 authentication.
2. Check authorization according to rules in `krb5_auth_rules(5)`.
3. A null byte is returned on the initial socket and the command line is passed to the normal login shell of the user. (The `PATH` variable is set to `/usr/bin`.) The shell inherits the network connections established by `in.rshd`.

In order for Kerberos authentication to work, a `host/`<*FQDN*> Kerberos principal must exist for each Fully Qualified Domain Name associated with the `in.rshd` server. Each of these `host/`<*FQDN*> principals must have a `keytab` entry in the `/etc/krb5/krb5.keytab` file on the `in.rshd` server. An example principal might be:

```
host/bigmachine.eng.example.com
```

See `kadmin(1M)` or `gkadmin(1M)` for instructions on adding a principal to a `krb5.keytab` file. See *Oracle Solaris 11.1 Administration: Security Services* for a discussion of Kerberos authentication.

If Kerberos V5 authentication is not enabled, then `in.rshd` executes the following protocol:

1. The server checks the client's source port. If the port is not in the range 512-1023, the server aborts the connection. The client's host address (in hex) and port number (in decimal) are the arguments passed to `in.rshd`.
2. The server reads characters from the socket up to a null (`\0`) byte. The resultant string is interpreted as an ASCII number, base 10.
3. If the number received in step 2 is non-zero, it is interpreted as the port number of a secondary stream to be used for the `stderr`. A second connection is then created to the specified port on the client's machine. The source port of this second connection is also in the range 512-1023.
4. A null-terminated user name of at most 16 characters is retrieved on the initial socket. This user name is interpreted as the user identity on the *client*'s machine.
5. A null terminated user name of at most 16 characters is retrieved on the initial socket. This user name is interpreted as a user identity to use on the *server*'s machine.

6. A null terminated command to be passed to a shell is retrieved on the initial socket. The length of the command is limited by the upper bound on the size of the system's argument list.
7. `in.rshd` then validates the user according to the following steps. The remote user name is looked up in the password file and a `chdir` is performed to the user's home directory. If the lookup fails, the connection is terminated. If the `chdir` fails, it does a `chdir` to `/` (root). If the user is not the superuser, (user ID 0), and if the pam_rhosts_auth PAM module is configured for authentication, the file `/etc/hosts.equiv` is consulted for a list of hosts considered "equivalent". If the client's host name is present in this file, the authentication is considered successful. See the SECURITY section below for a discussion of PAM authentication.

   If the lookup fails, or the user is the superuser, then the file `.rhosts` in the home directory of the remote user is checked for the machine name and identity of the user on the client's machine. If this lookup fails, the connection is terminated
8. A null byte is returned on the initial connection and the command line is passed to the normal login shell of the user. The `PATH` variable is set to `/usr/bin`. The shell inherits the network connections established by `in.rshd`.

Options The following options are supported:

`-5`	Same as `-k`, for backwards compatibility
`-c`	Requires Kerberos V5 clients to present a cryptographic checksum of initial connection information like the name of the user that the client is trying to access in the initial authenticator. This checksum provides additionl security by preventing an attacker from changing the initial connection information. This option is mutually exclusive with the `-i` option.
`-e`	Requires the client to encrypt the connection.
`-i`	Ignores authenticator checksums if provided. This option ignores authenticator checksums presented by current Kerberos clients to protect initial connection information. Option `-i` is the opposite of option `-c`.
`-k`	Allows Kerberos V5 authentication with the `.k5login` access control file to be trusted. If this authentication system is used by the client and the authorization check is passed, then the user is allowed to log in.
`-L` *env_var*	List of environment variables that need to be saved and passed along.
`-M` *realm*	Uses the indicated Kerberos V5 realm. By default, the daemon will determine its realm from the settings in the `krb5.conf(4)` file.
`-s` *tos*	Sets the `IP TOS` option.
`-S` *keytab*	Sets the `KRB5` keytab file to use. The`/etc/krb5/krb5.keytab` file is used by default.

`-U` Refuses connections that cannot be mapped to a name through the `getnameinfo(3SOCKET)` function.

Usage `rshd` and `in.rshd` are IPv6–enabled. See `ip6(7P)`. `IPv6` is not currently supported with Kerberos V5 authentication.

The Kerberized `rshd` service runs on port 544 (kshell). The corresponding FMRI entry is: :

```
svc:/network/shell:kshell (rshd with kerberos (ipv4 only))
```

Security `in.rshd` uses `pam(3PAM)` for authentication, account management, and session management. The PAM configuration policy, configured in `/etc/pam.conf` or per-service files in `/etc/pam.d/`, specifies the modules to be used for `in.rshd`. Here is a partial `pam.conf` file with entries for the `rsh` command using `rhosts` authentication, UNIX account management, and session management module.

rsh	auth	required	pam_rhosts_auth.so.1
rsh	account	required	pam_unix_roles.so.1
rsh	session	required	pam_unix_projects.so.1
rsh	session	required	pam_unix_account.so.1
rsh	session	required	pam_unix_session.so.1

The equivalent PAM configuration using `/etc/pam.d/` would be the following entries in `/etc/pam.d/rsh`:

```
auth         required   pam_rhosts_auth.so.1
account      required   pam_unix_roles.so.1
session      required   pam_unix_projects.so.1
session      required   pam_unix_account.so.1
session      required   pam_unix_session.so.1
```

If there are no entries for the `rsh` service in `/etc/pam.conf` and `/etc/pam.d/rsh` does not exist then the entries for the "other" service in `/etc/pam.conf` are used. If there are not any entries in `/etc/pam.conf` for the "other", then the entries in `/etc/pam.d/other` will be used. To maintain the authentication requirement for `in.rshd`, the `rsh` entry must always be configured with the `pam_rhosts_auth.so.1` module.

`in.rshd` can authenticate using Kerberos V5 authentication or `pam(3PAM)`. For Kerberized `rsh` service, the appropriate PAM service name is `krsh`.

Files

`/etc/hosts.equiv`	
`$HOME/.k5login`	File containing Kerberos principals that are allowed access.
`/etc/krb5/krb5.conf`	Kerberos configuration file.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	service/network/network-servers

See Also `rsh(1)`, `svcs(1)`, `gkadmin(1M)`, `inetadm(1M)`, `inetd(1M)`, `kadmin(1M)`, `svcadm(1M)`, `pam(3PAM)`, `getnameinfo(3SOCKET)`, `hosts(4)`, `krb5.conf(4)`, `pam.conf(4)`, `attributes(5)`, `environ(5)`, `krb5_auth_rules(5)`, `pam_authtok_check(5)`, `pam_authtok_get(5)`, `pam_authtok_store(5)`, `pam_dhkeys(5)`, `pam_passwd_auth(5)`, `pam_rhosts_auth(5)`, `pam_unix_account(5)`, `pam_unix_auth(5)`, `pam_unix_session(5)`, `smf(5)`, `ip6(7P)`

Oracle Solaris 11.1 Administration: Security Services

Diagnostics The following diagnostic messages are returned on the connection associated with `stderr`, after which any network connections are closed. An error is indicated by a leading byte with a value of 1 in step 8 above (`0` is returned above upon successful completion of all the steps prior to the command execution).

`locuser too long`	The name of the user on the client's machine is longer than 16 characters.
`remuser too long`	The name of the user on the remote machine is longer than 16 characters.
`command too long`	The command line passed exceeds the size of the argument list (as configured into the system).
`Hostname for your address unknown.`	No entry in the host name database existed for the client's machine.
`Login incorrect.`	No password file entry for the user name existed.
`Permission denied.`	The authentication procedure described above failed.
`Can't make pipe.`	The pipe needed for the `stderr` was not created.
`Try again.`	A *fork* by the server failed.

Notes The authentication procedure used here assumes the integrity of each client machine and the connecting medium. This is insecure, but it is useful in an "open" environment.

A facility to allow all data exchanges to be encrypted should be present.

The `in.rshd` service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/network/shell:default
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. Responsibility for initiating and restarting this service is delegated to `inetd(1M)`. Use `inetadm(1M)` to make configuration changes and to view configuration information for this service. The service's status can be queried using the `svcs(1)` command.

Name in.rwhod, rwhod – system status server

Synopsis `/usr/sbin/in.rwhod [-m [`*ttl*`]]`

Description `in.rwhod` is the server which maintains the database used by the `rwho(1)` and `ruptime(1)` programs. Its operation is predicated on the ability to broadcast or multicast messages on a network.

`in.rwhod` operates as both a producer and consumer of status information. As a producer of information it periodically queries the state of the system and constructs status messages which are broadcast or multicast on a network. As a consumer of information, it listens for other `in.rwhod` servers' status messages, validating them, then recording them in a collection of files located in the directory `/var/spool/rwho`.

The `rwho` server transmits and receives messages at the port indicated in the `rwho` service specification, see `services(4)`. The messages sent and received are defined in `/usr/include/protocols/rwhod.h` and are of the form:

```
struct  outmp {
        char    out_line[8];    /* tty name */
        char    out_name[8];    /* user id */
        long    out_time;       /* time on */
};
struct  whod {
        char    wd_vers;
        char    wd_type;
        char    wd_fill[2];
        int     wd_sendtime;
        int     wd_recvtime;
        char    wd_hostname[32];
        int     wd_loadav[3];
        int     wd_boottime;
        struct  whoent {
                struct  outmp we_utmp;
                int     we_idle;
    } wd_we[1024 / sizeof (struct whoent)];
 };
```

All fields are converted to network byte order prior to transmission. The load averages are as calculated by the `w(1)` program, and represent load averages over the 1, 5, and 15 minute intervals prior to a server's transmission. The host name included is that returned by the `uname(2)` system call. The array at the end of the message contains information about the users who are logged in to the sending machine. This information includes the contents of the `utmpx(4)` entry for each non-idle terminal line and a value indicating the time since a character was last received on the terminal line.

Messages received by the `rwho` server are discarded unless they originated at a `rwho` server's port. In addition, if the host's name, as specified in the message, contains any unprintable ASCII characters, the message is discarded. Valid messages received by `in.rwhod` are placed in files named `whod.hostname` in the directory `/var/spool/rwho`. These files contain only the most recent message, in the format described above.

Status messages are generated approximately once every 3 minutes.

Options The following options are supported:

`-m` [*ttl*] Use the rwho IP multicast address (224.0.1.3) when transmitting. Receive announcements both on this multicast address and on the IP broadcast address. If *ttl* is not specified `in.rwhod` multicasts on all interfaces but with the IP TimeToLive set to 1 (that is, packets are not forwarded by multicast routers.) If *ttl* is specified `in.rwhod` only transmits packets on one interface and setting the IP TimeToLive to the specified *ttl*.

Files `/var/spool/rwho/whod.*` information about other machines

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	service/network/network-servers

See Also `ruptime(1)`, `rwho(1)`, `w(1)`, `uname(2)`, `services(4)`, `utmpx(4)`, `attributes(5)`

Warnings This service can cause network performance problems when used by several hosts on the network. It is not run at most sites by default. If used, include the `-m` multicast option.

Notes This service takes up progressively more network bandwidth as the number of hosts on the local net increases. For large networks, the cost becomes prohibitive.

`in.rwhod` should relay status information between networks. People often interpret the server dying as a machine going down.

Name install – install commands

Synopsis

```
/usr/sbin/install -c dira [-m mode] [-u user] [-g group]
     [-o] [-s] file

/usr/sbin/install -f dirb [-m mode] [-u user] [-g group]
     [-o] [-s] file

/usr/sbin/install -n dirc [-m mode] [-u user] [-g group]
     [-o] [-s] file

/usr/sbin/install -d | -i [-m mode] [-u user] [-g group]
     [-o] [-s] dirx...

/usr/sbin/install [-m mode] [-u user] [-g group] [-o] [-s] file
     [dirx]...
```

Description `install` is most commonly used in "makefiles" (see `make(1S)`) to install a `file` in specific locations, or to create directories within a file system. Each `file` is installed by copying it into the appropriate directory.

`install` uses no special privileges to copy files from one place to another. The implications of this are:

- You must have permission to read the files to be installed.
- You must have permission to copy into the destination directory.
- You must have permission to change the modes on the final copy of the file if you want to use the `-m` option.
- You must be super-user if you want to specify the ownership of the installed file with the `-u` or `-g` options. If you are not the super-user, the installed file is owned by you, regardless of who owns the original.

Note that if the `ROOT` environment variable is set, each of the default directory paths are prefixed by its value (for example, `$ROOT/bin` and so on).

`install` prints messages telling the user exactly what files it is replacing or creating and where they are going.

If no options or directories (*dirx* . . .) are given, `install` searches a set of default directories (`/bin`, `/usr/bin`, `/etc`, `/lib`, and `/usr/lib`, in that order) for a file with the same name as `file`. When the first occurrence is found, `install` issues a message saying that it is overwriting that file with `file`, and proceeds to do so. If the file is not found, the program states this and exits.

If one or more directories (*dirx* . . .) are specified after `file`, those directories are searched before the default directories.

This version of `install` (`/usr/sbin/install`) is not compatible with the `install` binaries in many versions of Unix other than Solaris. For a higher degree of compatibility with other Unix versions, use `/usr/ucb/install`, which is described in the `install(1B)` man page.

Options The following options are supported:

`-c` *dira* Install `file` in the directory specified by *dira*, if `file` does not yet exist. If it is found, `install` issues a message saying that the file already exists, and exits without overwriting it.

`-f` *dirb* Force `file` to be installed in given directory, even if the file already exists. If the file being installed does not already exist, the mode and owner of the new file is set to `755` and `bin ,` respectively. If the file already exists, the mode and owner is that of the already existing file.

`-n` *dirc* If `file` is not found in any of the searched directories, it is put in the directory specified in *dirc*. The mode and owner of the new file is set to `755` and `bin`, respectively.

`-d` Create a directory. Missing parent directories are created as required as in `mkdir -p`. If the directory already exists, the owner, group and mode is set to the values given on the command line.

`-i` Ignore default directory list, searching only through the given directories (*dirx* . . .).

`-m` *mode* The mode of the new file is set to *mode*. Set to `0755` by default.

`-u` *user* The owner of the new file is set to *user*. Only available to the super-user. Set to `bin` by default.

`-g` *group* The group id of the new file is set to *group*. Only available to the super-user. Set to `bin` by default.

`-o` If `file` is found, save the "found" file by copying it to `OLDfile` in the directory in which it was found. This option is useful when installing a frequently used file such as `/bin/sh` or `/lib/saf/ttymon`, where the existing file cannot be removed.

`-s` Suppress printing of messages other than error messages.

Usage See `largefile(5)` for the description of the behavior of `install` when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also `chgrp(1)`, `chmod(1)`, `chown(1)`, `cp(1)`, `install(1B)`, `make(1S)`, `mkdir(1)`, `attributes(5)`, `largefile(5)`

Name installadm – Manages automated installations on a network

Synopsis

```
/usr/sbin/installadm [-h|--help]

installadm help [subcommand]

installadm create-service [-n|--service svcname]
    [-s|--source source]
    [-p|--publisher publisher=origin]
    [-a|--arch architecture]
    [-d|--imagepath imagepath] [-y|--noprompt]
    [-t|--aliasof aliasof]
    [-i|--ip-start start] [-c|--ip-count count]
    [-b|--boot-args property=value,...]
    [-B|--bootfile-server server]

installadm set-service -o|--option property=value svcname

installadm update-service [-p|--publisher publisher=origin]
    [-s|--source FMRI] svcname

installadm rename-service svcname newsvcname

installadm enable svcname

installadm disable svcname

installadm delete-service [-r|--autoremove] [-y|--noprompt]
    svcname

installadm list [-n|--service svcname] [-c|--client]
    [-m|--manifest] [-p|--profile]

installadm create-manifest -n|--service svcname
    -f|--file filename [-m|--manifest manifest]
    [-c|--criteria criteria=value|list|range... |
    -C|--criteria-file criteriafile] [-d|--default]

installadm update-manifest -n|--service svcname
    -f|--file filename [-m|--manifest manifest]

installadm delete-manifest -n|--service svcname
    -m|--manifest manifest

installadm create-profile -n|--service svcname
    -f|--file filename... [-p|--profile profile]
    [-c|--criteria criteria=value|list|range... |
    -C|--criteria-file criteriafile]

installadm update-profile -n|--service svcname
    -f|--file filename [-p|--profile profile]

installadm delete-profile -n|--service svcname
    -p|--profile profile...
```

```
installadm export -n|--service svcname
    -m|--manifest manifest... -p|--profile profile...
    [-o|--output pathname]

installadm validate -n|--service svcname
    -P|--profile-file filename... | -p|--profile profile...

installadm set-criteria -n|--service svcname
    -m|--manifest manifest -p|--profile profile...
    -c|--criteria criteria=value|list|range... |
    -C|--criteria-file criteriafile |
    -a|--append-criteria criteria=value|list|range...

installadm create-client -n|--service svcname
    [-b|--boot-args property=value,...] -e|--macaddr macaddr

installadm delete-client macaddr
```

Description The Automated Installer (AI) is used to automate the installation of the Oracle Solaris OS on one or more SPARC and x86 systems over a network.

The machine topography necessary to employ AI over the network is to have an install server, a DHCP server (this can be the same system as the install server), and the installation clients. On the install server, install services are set up to contain an AI boot image, which is provided to the clients in order for them to boot over the network, input specifications (AI manifests and derived manifests scripts), one of which will be selected for the client, and Service Management Facility (SMF) configuration profiles, zero or more of which will be selected for the client.

The AI boot image content is published as the package `install-image/solaris-auto-install`, and is installed by the `create-service` subcommand. The `create-service` subcommand is also able to accept and unpack an AI ISO file to create the AI boot image.

Install services are created with a default AI manifest, but customized manifests or derived manifests scripts (hereafter called "scripts") can be added to an install service by using the `create-manifest` subcommand. See *Installing Oracle Solaris 11.1 Systems* for information about how to create manifests and derived manifests scripts. The `create-manifest` subcommand also allows criteria to be specified, which are used to determine which manifest or script should be selected for an installation client. Criteria already associated with a manifest or script can be modified using the `set-criteria` subcommand.

Manifests can include information such as a target device, partition information, a list of packages, and other parameters. Scripts contain commands that query a running AI client system and build a custom manifest based on the information it finds. When AI is invoked with a script, AI runs that script as its first task, to generate a manifest.

When the client boots, a search is initiated for a manifest or script that matches the client's machine criteria. When a matching manifest or script is found, the client is installed with the

Oracle Solaris release according to the specifications in the matching manifest file, or to the specifications in the manifest file derived from the matching script. Each client can use only one manifest or script.

Each service has one default manifest or script. The default is used when the criteria of no other manifest or script matches the system being installed. Any manifest or script can be designated as the default. Any criteria associated with a default manifest or script become inactive and are not considered during manifest or script selection. If a different manifest or script is later made the default, the criteria of the former default manifest or script become active again. Manifests or scripts with no criteria associated with them can only be used as default manifests or scripts. Manifests or scripts without criteria become inactive when a different manifest or script is designated the default.

System configuration profiles are complementary to manifests and scripts in that they also contain specifications for an installation. In particular, profiles are used to specify configuration information such as user name, user password, time zone, host name, and IP address. Profiles can contain variables that are replaced at installation time with appropriate values for the client being installed. In this way, a single profile file can set different configuration parameters on different clients. See the "Examples" section.

System configuration profiles are processed by `smf`(5) and conform to document format `service_bundle`(4). See `sysconfig`(1M) and Chapter 11, "Configuring the Client System," in *Installing Oracle Solaris 11.1 Systems* for more information about system configuration profiles. Each client can use any number of system configuration profiles. A particular SMF property can be specified no more than once for each client system.

If you want a specific client to use a specific install service, you can associate that client with the service by using the `create-client` subcommand. You can also use `create-client` to modify an existing client.

The `installadm` utility can be used to accomplish the following tasks:

- Set up install services and aliases
- Update the net image of certain install services
- Set up installation images
- Set up or delete clients
- Add, update, or delete manifests and scripts
- Specify or modify criteria for a manifest or script
- Export manifests and scripts
- Add or delete system configuration profiles
- Validate profiles
- Specify or modify criteria for profiles
- Export profiles
- Enable or disable install services
- List install services
- List clients for an install service

- List manifests and scripts for an install service
- List profiles for an install service

Install Server Configuration Properties

The following properties of the `svc:/system/install/server:default` SMF service are used to configure the install server.

`all_services/networks`
A list of networks in CIDR format (for example, 192.168.56.0/24) to allow or disallow, depending on how the `all_services/exclude_networks` property is set.

Use this list of networks to specify which clients this install server serves. By default, the AI install server is configured to serve install clients on all networks that the server is connected to if the server is multihomed.

`all_services/exclude_networks`
A boolean value. If true, exclude networks specified by the `all_services/networks` property from being served by this install server. If false, include networks specified by the `all_services/networks` property.

`all_services/port`
Specifies the port that hosts the AI install services web server. By default, the web server is hosted on port 5555.

If you want to use a different port number from the default, customize the `port` property before you create any install services.

`all_services/default_imagepath_basedir`
Specifies the default location for images created by the `installadm create-service` command. Images are located at `all_services/default_imagepath_basedir/`*service_name*. The default value of this property is `/export/auto_install`.

`all_services/manage_dhcp`
A boolean value. If true, automatically update the local ISC DHCP configuration when client and service configurations are modified in the install server. If false, do not automatically maintain the ISC DHCP configuration.

Options

The `installadm` command has the following option:

`-h`, `--help`
Show the usage help message.

Sub-commands

The `installadm` command has the subcommands listed below. See also the "Examples" section below.

`installadm help [`*subcommand*`]`
Displays the syntax for the `installadm` utility.

subcommand
Displays the syntax for only the specified subcommand.

```
installadm create-service [-n|--service svcname]
[-s|--source source]
[-p|--publisher publisher=origin]
[-a|--arch architecture]
[-d|--imagepath imagepath] [-y|--noprompt]
[-t|--aliasof aliasof]
[-i|--ip-start start] [-c|--ip-count count]
[-b|--boot-args property=value,...]
[-B|--bootfile-server server]
```

This subcommand sets up a network boot image (net image) in the specified *imagepath* directory, and creates an install service that specifies how a client booted from the net image is installed.

The AI boot image content is published as the package `install-image/solaris-auto-install`. If the `-s` option is not specified, that package is installed from the first publisher in the system's publisher preference list that provides an instance of that package. The `-s` option accepts the `pkg` specification as a full FMRI or location of an image ISO file. The resulting net image is eventually located in *imagepath*. The net image enables client installations.

Note the following specifications:

- When the first install service of a given architecture is created on an install server, an alias of that service, `default-i386` or `default-sparc`, is automatically created. This default service is used for all installations to clients of that architecture that were not added to the install server explicitly with the `create-client` subcommand. To change the service aliased by the `default-`*arch* service, use the `set-service` subcommand. To update the `default-`*arch* service, use the `update-service` subcommand.

  If a `default-`*arch* alias is changed to a new install service and a local ISC DHCP configuration is found, this default alias boot file is set as the default DHCP server-wide boot file for that architecture.
- If you want a client to use a different install service than the default for that architecture, you must use the `create-client` subcommand to create a client-specific configuration.
- If the `-i` option and the `-c` option are used, and a DHCP server is not yet configured, an ISC DHCP server is configured.

  If an ISC DHCP server is already configured, that DHCP server is updated.

  Even when `-i` and `-c` arguments are provided and DHCP is configured, no binding exists between the install service being created and the IP range. When `-i` and `-c` are passed and the value of `all_services/manage_dhcp` is `true`, the IP range is set up, a new DHCP server is created if needed, and that DHCP server remains up and running for all install services and all clients to use. The network information provided to the DHCP server has no specific bearing on the service being created.

- If the IP range requested is not on a subnet that the install server has direct connectivity to and the install server is multihomed, the `-B` option is used to provide the address of the bootfile server (usually an IP address on this system). This should only be necessary when multiple IP addresses are configured on the install server and DHCP relays are employed. In all other configurations, the software can determine this automatically.

`-n|--service` *svcname*
Optional: Uses this install service name instead of a system-generated service name. The *svcname* can consist of alphanumeric characters, underscores (_), and hyphens (-). The first character of *svcname* cannot be a hyphen. The length of the *svcname* cannot exceed 63 characters.

If the `-n` option is not specified, a service name is generated automatically. The default name includes architecture and OS version information.

`-s|--source` *source*
Optional: Specifies the data source for the net image. This can be either of:

- The FMRI of an IPS AI net image package. This is the default. If the `-s` option is not specified, the newest available version of the `install-image/solaris-auto-install` package is used. The package is retrieved from the publisher specified by the `-p` option or from the first publisher in the install server's publisher preference list that provides an instance of the package.
- The path to an AI ISO image.

`-p|--publisher` *publisher=origin*
Optional: Only applies when the service is being created from an IPS package. Specifies the IPS package repository from where you want to retrieve the `install-image/solaris-auto-install` package. An example is `solaris=http://pkg.oracle.com/solaris/release/`.

If the `-p` option is not specified, the publisher used is the first publisher in the install server's publisher preference list that provides an instance of the package.

`-a|--arch` *architecture*
Optional: Only applies when the service is being created from an IPS package. Specifies the architecture of the clients to be installed with this service. The value can be either `i386` or `sparc`. The default is the architecture of the install server.

`-d|--imagepath` *imagepath*
Optional: Specifies the path at which to create the net image. If not specified, the image is created in a *svcname* directory at the location defined by the value of the `all_services/default_imagepath_basedir` property. For the default value of this property, see "Install Server Configuration Properties." A confirmation prompt is displayed unless `-y` is also specified.

`-y|--noprompt`
Optional: Suppresses any confirmation prompts and proceeds with service creation using the supplied options and any default values (see `-d`).

`-t|--aliasof` *aliasof*
Optional: This new service is an alternate name for the *aliasof* install service.

`-i|--ip-start` *start*
Optional: Specifies the starting IP address in a range to be added to the local DHCP configuration. The number of IP addresses is provided by the `-c` option. If a local ISC DHCP configuration does not exist, an ISC DHCP server is started if the value of `all_services/manage_dhcp` is `true`.

`-c|--ip-count` *count*
Optional: Sets up a total number of IP addresses in the DHCP configuration equal to the value of the *count*. The first IP address is the value of *start* that is provided by the `-i` option.

`-b|--boot-args` *property*=*value*`,...`
Optional: For x86 clients only. Sets a property value in the service-specific boot configuration file in the service image. Use this option to set boot properties that are specific to this service. This option can accept multiple comma-separated *property*=*value* pairs.

`-B|--bootfile-server` *server*
Optional: Used to provide the IP address of the boot server from which clients should request bootfiles. Only required if this IP address cannot be determined by other means.

`installadm set-service -o|--option` *property*=*value* *svcname*

`-o|--option` *property*=*value*
Specifies the property and value to set.

property=*value* can be:

- `aliasof=`*aliasof*

  Makes *svcname* an alias of the *aliasof* install service.

- `imagepath=`*newpath*

  Relocates the `imagepath` of an existing service.

- `default-manifest=`*manifest*

  Designates a particular manifest or derived manifests script that is already registered with the specified service to be the default manifest or script for that service. Use the `installadm list` command to show a list of manifests and scripts registered with this service.

  ```
  $ installadm list -n svcname -m
  ```

svcname
Required: Specifies the name of the install service whose property is being set.

`installadm update-service [-p|--publisher` *publisher=origin*`]`
`[-s|--source` *FMRI*`]` *svcname*
Updates the image associated with *svcname*, where *svcname* is an alias of a service that was created using an IPS AI net image package. A new service is created with the updated image, and *svcname* is aliased to the new service.

`-p|--publisher` *publisher=origin*
The IPS package repository from which to update the *svcname* image. An example value is `solaris=http://pkg.oracle.com/solaris/release/`.

If the `-p` option is not specified, the publisher used is the publisher that was used to create the image of the service for which *svcname* is an alias. The following `pkg publisher` command shows how to display the *svcname* publisher:

```
$ installadm list
Service Name     Alias Of          Status  Arch  Image Path
------------     --------          ------  ----  ----------
default-i386     solaris11_1-i386  on      i386  /export/auto_install/solaris11_1-i386
solaris11_1-i386 -                 on      i386  /export/auto_install/solaris11_1-i386
$ pkg -R /export/auto_install/solaris11_1-i386 publisher
PUBLISHER        TYPE     STATUS   URI
solaris          origin   online   http://pkg.oracle.com/solaris/release/
```

`-s|--source` *FMRI*
The FMRI of the net image package for the update.

If the `-s` option is not specified, the newest available version of the `install-image/solaris-auto-install` package is used from the publisher specified in the description of the `-p` option.

svcname
Required: Specifies the name of the install service being updated, which must be an alias of a service that was created using an IPS net image package.

`installadm rename-service` *svcname newsvcname*
Renames the install service *svcname* to *newsvcname*. The *newsvcname* can consist of alphanumeric characters, underscores (_), and hyphens (-). The first character of *newsvcname* cannot be a hyphen. The length of the *newsvcname* cannot exceed 63 characters.

`installadm enable` *svcname*
Enables the *svcname* install service.

`installadm disable` *svcname*
Disables the *svcname* install service.

`installadm delete-service [-r|--autoremove] [-y|--noprompt]` *svcname*

Deletes an install service.

- Deletes the manifests, profiles, client configuration files, and web server configuration for this install service.
- Deletes the image used to instantiate the service.
- If the following conditions exist, the bootfile associated with this service is removed from the ISC DHCP configuration:
  - The service is a default alias.
  - A local ISC DHCP configuration exists.
  - The `all_services/manage_dhcp` property value is `true`.

`-r|--autoremove`
If specified, any clients assigned to this service, and any services aliased to this service, are also removed.

`-y|--noprompt`
Suppresses any confirmation prompts and proceeds with service deletion.

svcname
Required: Specifies the install service name to delete.

`installadm list [-n|--service` *svcname*`] [-c|--client] [-m|--manifest] [-p|--profile]`

Lists all enabled install services on a server.

`-n|--service` *svcname*
Optional: Lists information about the specific install service on a local server.

- If the `-c` option is specified, lists the client information associated with the install service.
- If the `-m` option is specified, lists the manifests and derived manifests scripts associated with the install service.
- If the `-p` option is specified, lists the profiles associated with the install service.

`-c|--client`
Optional: Lists the clients of the install services on a local server.

`-m|--manifest`
Optional: Lists the manifests and derived manifests scripts associated with the install services on a local server, including criteria for each manifest. Criteria associated with the default manifest for the service are labeled as ignored. Inactive manifests are labeled. Inactive manifests have no associated criteria and are not the default manifest for that service.

When `-n` is not specified, displays all manifests and scripts for all services.

When `-n` is specified, displays all manifests and scripts for the given service.

`-p|--profile`
Optional: Lists the profiles associated with the install services on a local server, including criteria for each profile.

When `-n` is not specified, displays all profiles for all services.

When `-n` is specified, displays the profiles for the given service.

`installadm create-manifest -n|--service` *svcname*
`-f|--file` *filename* `[-m|--manifest` *manifest*`]`
`[-c|--criteria` *criteria*=*value*|*list*|*range*`... |`
`-C|--criteria-file` *criteriafile*`] [-d|--default]`
Creates a manifest or derived manifests script for a specific install service, thus making the manifest or script available on the network, independently from creating a service. A non-default manifest or script can be used (can be active) only when criteria are associated with it. Criteria can be entered on the command line (`-c`) or in a criteria XML file (`-C`). Any criteria specified along with the `-d` option are temporarily ignored until the manifest or script is no longer designated as the default.

The name of the manifest is determined in the following order:

1. The *manifest* name specified by the `-m` option, if present.
2. The value of the `ai_instance` `name` attribute, if present in the manifest.
3. The base name of the *filename*.

`-n|--service` *svcname*
Required: Specifies the name of the install service this manifest or script is to be associated with.

`-f|--file` *filename*
Required: Specifies the path name of the manifest or derived manifests script to add.

`-m|--manifest` *manifest*
Optional: Specifies the AI instance name of the manifest or derived manifests script. Sets the `name` attribute of the `ai_instance` element of the manifest to *manifest*. The manifest or script is referred to as *manifest* in subsequent `installadm` commands and `installadm list` output.

`-c|--criteria` *criteria*=*value*|*list*|*range*`...`
Optional: Specifies criteria to be associated with the added manifest or script. See the "Criteria" section below. When publishing a default manifest, criteria are registered but kept inactive until the manifest or script is no longer designated the default. The `-c` option can be specified multiple times.

`-C|--criteria-file` *criteriafile*
Optional: Specifies the path name of a criteria XML file containing criteria to be associated with the added manifest or script. When publishing a default manifest or script, criteria are registered but kept inactive until the manifest or script is no longer designated the default.

`-d|--default`
Optional: Specifies that this manifest or script is the new default manifest or script for the service. Any criteria specified are stored, but these criteria are ignored until a different manifest or script is made the default.

`installadm update-manifest -n|--service` *svcname*
`-f|--file` *filename* `[-m|--manifest` *manifest*`]`
Updates the specific manifest or derived manifests script from the *svcname* install service. Replaces the specified manifest or script with the contents of *filename*. Any criteria or default status remain with the manifest or script following the update.

The name of the manifest is determined in the following order:

1. The *manifest* specified by the `-m` option, if present.
2. The value of the `ai_instance name` attribute, if present in the changed manifest and if it matches the `ai_instance name` value of an existing manifest.
3. The base name of the *filename*, if it matches the `ai_instance name` attribute value in an existing manifest, or the name given by `installadm list` if it matches the name of an existing script.

`-n|--service` *svcname*
Required: Specifies the name of the install service of the manifest or script being updated.

`-f|--file` *filename*
Required: Specifies the path name of the replacement manifest or derived manifests script.

`-m|--manifest` *manifest*
Optional: Specifies the AI instance name of the replacement manifest or script.

`installadm delete-manifest -n|--service` *svcname*
`-m|--manifest` *manifest*
Deletes a manifest or derived manifests script that was published with a specific install service. A default manifest or script cannot be deleted.

`-n|--service` *svcname*
Required: Specifies the name of the install service of the manifest or script being deleted.

`-m|--manifest` *manifest*
Required: Specifies the AI instance name of a manifest or derived manifests script as output by `installadm list` with the `-n` option.

`installadm create-profile -n|--service` *svcname*
`-f|--file` *filename*`...` `[-p|--profile` *profile*`]`
`[-c|--criteria` *criteria*=*value*|*list*|*range*`...` `|`
`-C|--criteria-file` *criteriafile*`]`

Creates profiles for a specific install service. Criteria can optionally be associated with a profile by either entering them on the command line (`-c`) or in a criteria XML file (`-C`). Profiles created without criteria are associated with all clients of the service.

The name of the profile is determined in the following order:

1. The *profile* specified by the `-p` option, if present.
2. The base name of the *filename*.

Profile names must be unique for an AI service. If multiple `-f` options are used to create more than one profile with the same criteria, then the `-p` option is invalid and the names of the profiles are derived from their file names.

`-n|--service` *svcname*
Required: Specifies the name of the install service of the profile being created.

`-f|--file` *filename*`...`
Required: Specifies the path name of the file with which to add the profile. Multiple profiles can be specified.

`-p|--profile` *profile*
Optional: Specifies the name of the profile being created. Valid only for single profile creation.

`-c|--criteria` *criteria*=*value*|*list*|*range*`...`
Optional: Specifies criteria to be associated with the profiles. See the "Criteria" section below. Multiple `-c` options can be specified.

`-C|--criteria-file` *criteriafile*
Optional: Specifies the path name of a criteria XML file containing criteria to be associated with the specified profiles.

`installadm update-profile -n|--service` *svcname*
`-f|--file` *filename* `[-p|--profile` *profile*`]`

Updates the specified profile from the *svcname* install service. Replaces the specified profile with the contents of *filename*. Any criteria remain with the profile following the update.

The profile to be updated is determined in the following order:

1. The *profile* specified by the `-p` option, if present.
2. The base name of the *filename*.

`-n|--service` *svcname*
Required: Specifies the name of the install service of the profile being updated.

`-f|--file` *filename*
Required: Specifies the path name of the file to use to update the profile.

`-p|--profile` *profile*
Optional: Specifies the name of the profile being updated. Use this option if the name of the profile to update is different from the base name of the *filename*.

`installadm delete-profile -n|--service` *svcname*
`-p|--profile` *profile*`...`
Deletes the *profile* profile from the *svcname* install service.

`-n|--service` *svcname*
Required: Specifies the name of the install service of the profile being deleted.

`-p|--profile` *profile*`...`
Required: Specifies the name of the profile to delete. Multiple `-p` options can be specified.

`installadm export -n|--service` *svcname*
`-m|--manifest` *manifest*`... -p|--profile` *profile*`...`
`[-o|--output` *pathname*`]`
Displays (exports) the specified manifests, derived manifests scripts, and profiles belonging to a specified service. At least one manifest, script, or profile must be specified. Display goes to `stdout` unless the `-o` option redirects to a file or directory.

`-n|--service` *svcname*
Required: Specifies the install service associated with the manifest, script, or profile to export.

`-m|--manifest` *manifest*`...`
Specifies the AI instance name of a manifest or derived manifests script to export. Multiple `-m` options can be specified.

`-p|--profile` *profile*`...`
Specifies the name of a profile to export. Multiple `-p` options can be specified.

`-o|--output` *pathname*
Optional: Redirect output. The *pathname* must be a directory if multiple manifests, scripts, or profiles are requested. The *pathname* can be a file if only one manifest, script, or profile is requested.

`installadm validate -n|--service` *svcname*
`-P|--profile-file` *filename*`... | -p|--profile` *profile*`...`
Validates specified profiles. The `validate` subcommand can be used to either validate profiles in the database (`-p`) or to validate profiles while they are being developed before their entry into the database (`-P`).

`-n|--service` *svcname*
Required: Specifies the service with which the profiles are associated.

`-P|--profile-file` *filename*`...`
Specifies an external profile file to validate.

`-p|--profile` *profile*`...`
Specifies the name of the profile to validate.

`installadm set-criteria -n|--service` *svcname*
`-m|--manifest` *manifest* `-p|--profile` *profile*`...`
`-c|--criteria` *criteria*=*value*|*list*|*range*`... |`
`-C|--criteria-file` *criteriafile* `|`
`-a|--append-criteria` *criteria*=*value*|*list*|*range*`...`
Updates criteria of an already published manifest or derived manifests script, profile, or both. Criteria can be specified on the command line or in a criteria XML file. Criteria must be specified with one of the mutually exclusive options, `-a`, `-c`, or `-C`.

Valid criteria are described under the `create-manifest` subcommand.

`-n|--service` *svcname*
Required: Specifies the name of the install service associated with this manifest, script, or profile.

`-m|--manifest` *manifest*
Specifies the AI instance name of a manifest or derived manifests script.

`-p|--profile` *profile*`...`
Specifies the name of a profile. Any number of profiles can be specified.

`-c|--criteria` *criteria*=*value*|*list*|*range*`...`
Specifies criteria to replace all existing criteria for the manifest, script, or profile. See the "Criteria" section below.

`-C|--criteria-file` *criteriafile*
Specifies the path name of a criteria XML file containing criteria to replace all existing criteria for the manifest, script, or profile.

`-a|--append-criteria` *criteria*=*value*|*list*|*range*`...`
Specifies criteria to be appended to the existing criteria for the manifest, script, or profile. See the "Criteria" section below. If the *criteria* specified already exists, the *value*|*list*|*range* of that criteria is replaced by the specified *value*|*list*|*range*.

`installadm create-client -n|--service` *svcname*
`[-b|--boot-args` *property*=*value*`,...] -e|--macaddr` *macaddr*
Accomplishes optional setup tasks for a specified client, in order to provide custom client settings that vary from the default settings used by the `create-service` subcommand. Enables the user to specify a non-default service name and boot arguments for a client. Can also be used to modify an existing client.

If the following conditions exist, the client is configured in the ISC DHCP configuration:

- The client is an x86 system.
- A local ISC DHCP configuration exists.
- The `all_services/manage_dhcp` property value is `true`.

`-n|--service` *svcname*
Required: Specifies the install service for client installation.

`-b|--boot-args` *property=value*`,...`
Optional: For x86 clients only. Sets a property value in the client-specific boot configuration file in `/etc/netboot`. Use this option to set boot properties that are specific to this client. This option can accept multiple *property=value* pairs.

`-e|--macaddr` *macaddr*
Required: Specifies a MAC address for the client.

`installadm delete-client` *macaddr*
Deletes an existing client's specific service information that was previously set up using the `create-client` subcommand.

If the following conditions exist, the client is unconfigured in the ISC DHCP configuration:

- The client is an x86 system.
- A local ISC DHCP configuration exists.
- The `all_services/manage_dhcp` property value is `true`.

macaddr Required: Specifies the MAC address of the client to delete.

Criteria Manifests, derived manifests scripts, and profiles can be used to configure AI clients differently according to certain characteristics, or criteria. Only one manifest or script can be associated with a particular client. Any number of profiles can be associated with a particular client.

The criteria values are determined by the AI client during startup.

See the "Examples" section to see how to specify criteria on the command line. For information about creating a criteria file, see *Installing Oracle Solaris 11.1 Systems*.

Criteria	Description
`arch`	Architecture per `uname -m`.
`cpu`	CPU class per `uname -p`.
`hostname`	Assigned host name.
`ipv4`	IP version 4 network address.
`mac`	Hexadecimal MAC address with colon (:) separators.
`mem`	Memory size in MB per `prtconf`(1M).
`network`	IP version 4 network number.
`platform`	Platform name returned by `uname -i` for x86 systems and `prtconf -b` for SPARC systems.

Criteria	Description
zonename	Name of a zone per zones(5).

The ipv4, mac, mem, and network specifications can be expressed as ranged values separated by a hyphen (-). To specify no limit to one end of a range, use unbounded.

The arch, cpu, hostname, platform, and zonename specifications can be expressed as a quoted list of values separated by white space.

Examples

EXAMPLE 1 Set Up a New x86 Install Service From an ISO File

Set up an install server and an x86 install service for the first time. The command includes a starting IP address and total count of IP addresses, in order to configure the DHCP server.

```
$ pfexec installadm create-service -n sol-11_1-i386 \
-s /export/isos/sol-11_1-ai-x86.iso \
-i 172.0.0.10 -c 10 -d /export/images/sol-11_1-i386
```

The AI ISO image is at /export/isos/sol-11_1-ai-x86.iso. The command sets up a net image and an install service at /export/images/sol-11_1-i386 that is based on the AI ISO image. This net image enables client installations.

The starting IP address of 172.0.0.10 and ten IP addresses are added to the local ISC DHCP configuration. If a local ISC DHCP configuration does not exist, an ISC DHCP server is started.

Because this is the first x86 service created, the default-i386 service is automatically created and aliased to this service. The default-i386 alias is operational, and a client booted via PXE will boot and install from the default-i386 service.

EXAMPLE 2 Set Up a New SPARC Install Service From an ISO File

Set up a SPARC install service for the first time.

```
$ pfexec installadm create-service -n sol-11_1-sparc \
-s /export/isos/sol-11_1-ai-sparc.iso \
-d /export/images/sol-11_1-sparc
```

The AI ISO image is at /export/isos/sol-11_1-ai-sparc.iso. The command sets up a net image and an install service at /export/images/sol-11_1-sparc that is based on the AI ISO image. This net image enables client installations.

Because this is the first SPARC service created, the default-sparc service is automatically created and aliased to this service. The default-sparc alias is operational, and a SPARC client will boot and install from the default-sparc service.

EXAMPLE 3 Set Up an x86 Install Service From a Package Repository

If you do not specify a source for the net image, an IPS package is used.

```
$ pfexec installadm create-service -y
```

On an x86 install server, this command sets up an x86 net image and install service with a default name in a directory at the image location specified by the value of the `all_services/default_imagepath_basedir` property. For the default value of this property, see "Install Server Configuration Properties." The `-y` option confirms that the default location is acceptable. Since the architecture is not specified, the service created is of the same architecture as the install server. This command assumes that a package repository on the `pkg publisher` list for the install server contains the `install-image/solaris-auto-install` package.

To specify the creation of a SPARC service on this server, use the `-a` option.

To specify the publisher of the `solaris-auto-install` package, use the `-p` option. For example, use the following command to specify the `ai-image` publisher located at `http://example.company.com:4281` as the publisher of the `solaris-auto-install` package:

```
$ pfexec installadm create-service -y \
-p ai-image=http://example.company.com:4281
```

EXAMPLE 4 Associate a Client With an Install Service

Use the following sample command to associate a client with a specific install service. The install service must already exist.

```
$ pfexec installadm create-client -b "console=ttya" \
-e 0:e0:81:5d:bf:e0 -n sol-11_1-i386
```

In this example, the command creates a client-specific setup for the system with MAC address `0:e0:81:5d:bf:e0`. This client will use the install service previously set up, named `sol-11_1-i386`, and that service's associated net image. The command sets the boot property `console=ttya` in the client-specific boot configuration file in `/etc/netboot`.

EXAMPLE 5 Add a New Install Service Without Modifying the Default Service

Use the following sample command to add a new service named `sol-11-sparc`, retaining existing services, and leaving the existing default unchanged.

```
$ pfexec installadm create-service -n sol-11-sparc \
-s /export/isos/sol-11-1111-ai-sparc.iso \
-d /export/ai/sol-11-sparc
```

EXAMPLE 6 Update the default-i386 Service

Use the following sample command to update the `default-i386` alias service to be associated with the latest available image. The `installadm list` command shows the service before and after the command. The example assumes that an updated net image package is available from the publisher that was originally used to create the `default-i386` service alias.

EXAMPLE 6 Update the default-i386 Service *(Continued)*

```
$ installadm list
Service Name   Alias Of        Status  Arch  Image Path
------------   --------        ------  ----  ----------
default-i386   solaris11-i386  on      i386  /export/images/solaris11-i386
solaris11-i386 -               on      i386  /export/images/solaris11-i386
$ pfexec installadm update-service default-i386
...
Creating new i386 service: solaris11_1-i386
Aliasing default-i386 to solaris11_1-i386 ...
...
$ installadm list
Service Name     Alias Of          Status  Arch  Image Path
------------     --------          ------  ----  ----------
default-i386     solaris11_1-i386  on      i386  /export/images/solaris11_1-i386
solaris11-i386   -                 on      i386  /export/images/solaris11-i386
solaris11_1-i386 -                 on      i386  /export/images/solaris11_1-i386
```

EXAMPLE 7 Add a New Install Service and Update the default-sparc Service

Use the following two sample commands to add a new service named `my-sparc-service`, retaining existing services, and making the new service the default for SPARC clients.

```
$ pfexec installadm create-service -n solaris11_1-sparc \
-s /export/isos/sol-11_1-ai-sparc.iso \
-d /export/ai/solaris11_1-sparc
$ pfexec installadm set-service \
-o aliasof=solaris11_1-sparc default-sparc
```

EXAMPLE 8 Add a Custom Default AI Manifest to an Install Service

Use the following sample command to add a new manifest to the `sol-11_1-i386` install service, and make it the service's default manifest. The manifest data is in `my_default.xml`. Future `installadm` commands will refer to this manifest as `my_default`.

```
$ pfexec installadm create-manifest -d -f my_default.xml \
-m my_default -n sol-11_1-i386
```

EXAMPLE 9 Add a Derived Manifests Script to an Install Service

Use the following sample command to add a derived manifests script named `my_script` to an existing install service named `solaris11_1-i386`. Scripts are added in the same way that manifests are added.

```
$ pfexec installadm create-manifest -f my_script.py \
-m my_script -n solaris11_1-i386
```

See *Installing Oracle Solaris 11.1 Systems* for information about how to create derived manifests scripts.

EXAMPLE 10 Replace the Default AI Manifest for an Install Service

Use the following sample command to replace the default manifest for an existing install service, `sol-11_1-sparc`, with a custom manifest that has already been added to the service as `custom_manifest`. The manifest was added to the service by specifying `-m custom_manifest` to the `create-manifest` subcommand.

```
$ pfexec installadm set-service \
-o default-manifest=custom_manifest sol-11_1-sparc
```

EXAMPLE 11 List Install Services

Use the following sample command to list the install services on a local server.

```
$ installadm list
Service Name   Alias Of       Status  Arch   Image Path
------------   --------       ------  ----   ----------
default-i386   sol-11_1-i386  on      i386   /export/images/sol-11_1-i386
default-sparc  sol-11_1-sparc on      sparc  /export/images/sol-11_1-sparc
sol-11_1-i386  -              on      i386   /export/images/sol-11_1-i386
sol-11_1-sparc -              on      sparc  /export/images/sol-11_1-sparc
```

EXAMPLE 12 List Clients Associated With an Install Service

Use the following sample command to list the clients of a specific install service on a local server.

```
$ installadm list -c -n sol-11_1-i386
Service Name  Client Address     Arch  Image Path
------------  --------------     ----  ----------
sol-11_1-i386 01:C2:52:E6:4B:E1  i386  /export/images/sol-11_1-i386
```

EXAMPLE 13 List Manifests Associated With an Install Service

Use the following sample command to list the manifests and derived manifests scripts associated with a specific install service on a local server.

```
$ installadm list -m -n sol-11_1-sparc
Service/Manifest Name  Status   Criteria
---------------------  ------   --------
sol-11_1-sparc
   mem                          mem  = 4096 MB - unbounded
   custom_manifest     Default  (Ignored:
                                mem  = 2048 MB - 4095 MB)
   orig_default        Inactive None
```

This example shows the following output:

- A non-default manifest with criteria (`mem`)
- A default manifest with criteria that are ignored (`custom_manifest`)
- A non-default manifest (`orig_default`) that is marked inactive because it has no criteria

EXAMPLE 14 List Profiles

Use the following sample command to list the system configuration profiles for all install services on a local server.

```
$ installadm list -p
Service/Profile Name  Criteria
--------------------  --------
sol-11_1-i386
   sc_all-x86.xml     None

sol-11_1-sparc
   sc_all-sparc.xml   None
   sc_network.xml     network  = 10.0.0.0
                      ipv4     = 10.0.2.100 - 10.0.2.199
```

EXAMPLE 15 Add a Custom AI Manifest With No Name to an Install Service

Use the following sample command to add the manifest in `/export/my_manifest.xml` to `sol-11_1-i386` with a criterion of MAC address equaling `aa:bb:cc:dd:ee:ff`.

```
$ pfexec installadm create-manifest \
-f /export/my_manifest.xml -n sol-11_1-i386 \
-c mac="aa:bb:cc:dd:ee:ff"
```

In this example, the manifest does not contain a name attribute, so the manifest name is taken from the file name.

```
$ installadm list -m -n sol-11_1-i386
Service/Manifest Name  Status  Criteria
---------------------  ------  --------
sol-11_1-i386
   my_manifest.xml             mac  = AA:BB:CC:DD:EE:FF
   orig_default        Default None
```

EXAMPLE 16 Add a Custom AI Manifest With a Custom Name to an Install Service

Use the following sample command to add the manifest in `/export/my_manifest.xml` to `sol-11_1-i386` with the criterion of IPv4 range from 10.0.2.100 and 10.0.2.199.

```
$ pfexec installadm create-manifest \
-f /export/my_manifest.xml \
-n sol-11_1-i386 -m custom_name \
-c ipv4="10.0.2.100-10.0.2.199"
```

In this example, the manifest name is taken from the `-m` option.

```
$ installadm list -m -n sol-11_1-i386
Service/Manifest Name  Status  Criteria
---------------------  ------  --------
sol-11_1-i386
   custom_name                 ipv4 = 10.0.2.100 - 10.0.2.199
```

EXAMPLE 16 Add a Custom AI Manifest With a Custom Name to an Install Service *(Continued)*

```
   orig_default        Default None
```

EXAMPLE 17 Add a Custom AI Manifest With Name Specified In the Manifest

Use the following sample command to add the manifest in `/export/manifest3.xml` to `sol-11_1-i386` with criteria of 2048 MB memory or greater and an architecture of `i86pc`.

```
$ pfexec installadm create-manifest \
-f /export/manifest3.xml -n sol-11_1-i386 \
-c mem="2048-unbounded" -c arch=i86pc
```

In this example, the manifest name is taken from the `name` attribute of the `ai_instance` element in the manifest, as shown in the following partial manifest:

```
<auto_install>
    <ai_instance name="my_name" />
</auto_install>
```

```
$ installadm list -m -n sol-11_1-i386
Service/Manifest Name  Status  Criteria
---------------------  ------  --------
sol-11_1-i386
   my_name                     arch = i86pc
                               mem  = 2048 MB - unbounded
   orig_default        Default None
```

EXAMPLE 18 Add a System Configuration Profile To an Install Service

Use the following sample command to add the profile in `/export/profile4.xml` to `sol-11_1-i386` with criteria of any of the host names `myhost1`, `host3`, or `host6`.

```
$ pfexec installadm create-profile \
-f /export/profile4.xml -n sol-11_1-i386 -p profile4 \
-c hostname="myhost1 host3 host6"
$ installadm list -p -n sol-11_1-i386
Service/Profile Name  Criteria
--------------------  --------
sol-11_1-i386
   profile4           hostname = myhost1 host3 host6
```

EXAMPLE 19 Add a System Configuration Profile For All Clients

If you do not specify criteria, then the profile is used by all clients that use the specified install service. In the following example, the created profile is used by all clients that use the `sol-11_1-i386` service.

```
$ pfexec installadm create-profile -f /export/locale.xml \
-n sol-11_1-i386
$ installadm list -p -n sol-11_1-i386
```

EXAMPLE 19 Add a System Configuration Profile For All Clients *(Continued)*

```
Service/Profile Name  Criteria
--------------------  --------
sol-11_1-i386
   profile4.xml       hostname = myhost1 host3 host6
   locale.xml
```

EXAMPLE 20 Add a System Configuration Profile With Variables

A profile can use variables that are replaced with custom client configuration information at client installation time. Using such variables, a profile file can be reused for any number of different systems.

This example uses one system configuration profile file to assign each install client a unique host name. The `hostname.xml` file contains the following line:

```
<propval name="nodename" value="{{AI_HOSTNAME}}"/>
```

At installation time, `{{AI_HOSTNAME}}` is replaced with the actual host name of that system. For example, when `hostname.xml` is used to configure the client with host name `myhost1`, the `hostname.xml` profile contains the following line:

```
<propval name="nodename" value="myhost1"/>
```

For more information about using replacement tags with profiles, see "Using System Configuration Profile Templates" in *Installing Oracle Solaris 11.1 Systems*.

EXAMPLE 21 Add Criteria To an Existing Manifest

Use the following sample command to append the criterion of 4096 MB memory or greater to the criteria of `manifest2` of `sol-11_1-i386`.

```
$ pfexec installadm set-criteria -m manifest2 \
-n sol-11_1-i386 -a mem="4096-unbounded"
```

EXAMPLE 22 Replace the Criteria for an Existing Manifest

Use the following sample command to replace the criteria of `manifest2` of `sol-11_1-i386` with the criteria specified in the file `/tmp/criteria.xml`.

```
$ pfexec installadm set-criteria -m manifest2 \
-n sol-11_1-i386 -C /tmp/criteria.xml
```

See *Installing Oracle Solaris 11.1 Systems* for information about the contents of the criteria XML file.

EXAMPLE 23 Validate Profile Files Under Development

Use the following sample command to validate the profiles stored in the files `myprofdir/myprofile.xml` and `yourprofdir/yourprofile.xml` during their development.

EXAMPLE 23 Validate Profile Files Under Development *(Continued)*

```
$ pfexec installadm validate -P myprofdir/myprofile.xml \
-P yourprofdir/yourprofile.xml -n sol-11_1-i386
```

EXAMPLE 24 Export Profile Contents

Use the following sample command to export the profile `myprofile.xml` in the service `sol-11_1-i386`.

```
$ installadm export -p myprofile -n sol-11_1-i386
```

EXAMPLE 25 Replace the Contents of an Existing AI Manifest

Use the following sample command to update the manifest in service `sol-11_1-i386` that has the manifest name, or AI instance name, `spec` with the contents of the manifest in the file `/home/admin/new_spec.xml`.

```
$ pfexec installadm update-manifest -n sol-11_1-i386 \
-f /home/admin/new_spec.xml -m spec
```

EXAMPLE 26 Export and Update an Existing AI Manifest

Use the following sample commands to export the data of an existing manifest named `spec` in service `sol-11_1-i386`, and then update the manifest with modified content.

```
$ pfexec installadm export -n sol-11_1-i386 -m spec \
-o /home/admin/spec.xml
```

Make changes to `/home/admin/spec.xml`.

```
$ pfexec installadm update-manifest -n sol-11_1-i386 \
-f /home/admin/spec.xml -m spec
```

EXAMPLE 27 Export and Update an Existing Profile

Use the following sample commands to export the data of an existing profile named `prof1` in service `sol-11_1-i386`, and then update the profile with modified content.

```
$ pfexec installadm export -n sol-11_1-i386 -p prof1 \
-o /home/admin/prof1.xml
```

Make changes to `/home/admin/prof1.xml`.

```
$ pfexec installadm update-profile -n sol-11_1-i386 \
-f /home/admin/prof1.xml -p prof1
```

Exit Status

The following exit values are returned:

`0` The command was processed successfully.

`1` An error occurred.

`2` Invalid command line options were specified.

3 A service's version is not supported by installadm.

4 No changes were made - nothing to do.

Attributes See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	`install/installadm`
Interface Stability	Uncommitted

See Also `aimanifest`(1M), `sysconfig`(1M), `dhcp`(5), `smf`(5), `service_bundle`(4), `ai_manifest`(4), `environ`(5)

Part III, "Installing Using an Install Server," in *Installing Oracle Solaris 11.1 Systems*

Transitioning From Oracle Solaris 10 JumpStart to Oracle Solaris 11.1 Automated Installer

Name installboot – install bootblocks in a disk partition

Synopsis

```
installboot [-F zfs|ufs|hsfs] bootblk raw-disk-device
```

Description The boot(1M) program, `ufsboot`, is loaded from disk by the bootblock program which resides in the boot area of a disk partition. This program is filesystem-specific, and must match the type of filesystem on the disk to be booted.

The boot objects are platform-dependent and reside in the `/usr/platform/`*platform-name*`/lib/fs/`*file-system* directory. The platform name can be found using the `-i` option of uname(1). The filesystem type can be found using:

```
% fstyp raw-disk-device
```

See fstyp(1M).

The `installboot` utility is a SPARC only program. It is not supported on the x86 architecture. x86 users should use installgrub(1M) instead.

Options The following option is supported:

`-F zfs|ufs|hsfs`
: Specifies the file system type of the boot block to be installed. Required if you wish to specify `zfs` or `hsfs`. The default is `ufs`.

Operands

bootblk
: The name of the bootblock code.

raw-disk-device
: The name of the disk device onto which the bootblock code is to be installed; it must be a character device which is readable and writable. Naming conventions for a SCSI or IPI drive are `c?t?d?s?` and `c?d?s?` for an IDE drive.

Examples

EXAMPLE 1 Installing UFS Boot Block

To install a `ufs` boot block on slice `0` of target `0` on controller `1` of the platform where the command is being run, use:

```
# installboot /usr/platform/‘uname -i‘/lib/fs/ufs/bootblk \
    /dev/rdsk/c1t0d0s0
```

EXAMPLE 2 Installing ZFS Boot Block

To install a ZFS boot block on slice 0 of target 0 on controller 1 of the platform where the command is being run, use syntax such as the following:

```
# installboot -F zfs /usr/platform/‘uname -i‘/lib/fs/zfs/bootblk \
      /dev/rdsk/c1t1d0s0
```

Files `/usr/platform/`*platform-name*`/lib/fs/` Directory where boot objects reside.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also `kmdb(1)`, `od(1)`, `uname(1)`, `boot(1M)`, `fstyp(1M)`, `init(1M)`, `kernel(1M)`, `monitor(1M)`, `reboot(1M)`, `rpc.bootparamd(1M)`, `init.d(4)`, `attributes(5)`

Installing Oracle Solaris 11.1 Systems

Warnings The `installboot` utility fails if the *bootblk* or *openfirmware* files do not exist or if the raw disk device is not a character device.

Name installf – add a file to the software installation database

Synopsis

```
installf [-c class] [ [-M] -R root_path] [-V fs_file] pkginst pathname
     [ftype [major minor] [mode owner group]]

installf [-c class] [ [-M] -R root_path] [-V fs_file] pkginst –

installf -f [-c class] [ [-M] -R root_path] [-V fs_file] pkginst
```

Description `installf` informs the system that a pathname not listed in the `pkgmap(4)` file is being created or modified. It should be invoked before any file modifications have occurred.

When the second synopsis is used, the pathname descriptions will be read from standard input. These descriptions are the same as would be given in the first synopsis but the information is given in the form of a list. The descriptions should be in the form:

pathname [*ftype* [*major minor*] [*mode owner group*]]

After all files have been appropriately created and/or modified, `installf` should be invoked with the `-f` synopsis to indicate that installation is final. Links will be created at this time and, if attribute information for a pathname was not specified during the original invocation of `installf`, or was not already stored on the system, the current attribute values for the pathname will be stored. Otherwise, `installf` verifies that attribute values match those given on the command line, making corrections as necessary. In all cases, the current content information is calculated and stored appropriately.

Package commands are `largefile(5)`-aware. They handle files larger than 2 GB in the same way they handle smaller files. In their current implementations, `pkgadd(1M)`, `pkgtrans(1)` and other package commands can process a datastream of up to 4 GB.

Options

`-c` *class* — Class to which installed objects should be associated. Default class is `none`.

`-f` — Indicates that installation is complete. This option is used with the final invocation of `installf` (for all files of a given class).

`-M` — Instruct `installf` not to use the $*root_path*`/etc/vfstab` file for determining the client's mount points. This option assumes the mount points are correct on the server and it behaves consistently with Solaris 2.5 and earlier releases.

`-R` *root_path* — Define the full path name of a directory to use as the *root_path*. All files, including package system information files, are relocated to a directory tree starting in the specified *root_path*. The *root_path* can be specified when installing to a client from a server (for example, `/export/root/client1`).

`installf` inherits the value of the `PKG_INSTALL_ROOT` environment variable. (See ENVIRONMENT VARIABLES, below.) If `PKG_INSTALL_ROOT` is set, such as when the `-R` option is used with `pkgadd(1M)` or `pkgrm(1M)`

Note – The root file system of any non-global zones must not be referenced with the `-R` option. Doing so might damage the global zone's file system, might compromise the security of the global zone, and might damage the non-global zone's file system. See `zones(5)`.

`-V` *fs_file* Specify an alternative *fs_file* to map the client's file systems. For example, used in situations where the `$`*root_path*`/etc/vfstab` file is non-existent or unreliable.

Operands

pkginst Name of package instance with which the pathname should be associated.

pathname Pathname that is being created or modified.

ftype A one-character field that indicates the file type. Possible file types include:

- `b` block special device
- `c` character special device
- `d` directory
- `e` a file to be edited upon installation or removal
- `f` a standard executable or data file
- `l` linked file
- `p` named pipe
- `s` symbolic link
- `v` volatile file (one whose contents are expected to change)
- `x` an exclusive directory

major The major device number. The field is only specified for block or character special devices.

minor The minor device number. The field is only specified for block or character special devices.

mode The octal mode of the file (for example, 0664). A question mark (`?`) indicates that the mode will be left unchanged, implying that the file already exists on the target machine. This field is not used for linked or symbolically linked files.

owner The owner of the file (for example, `bin` or `root`). The field is limited to 14 characters in length. A question mark (`?`) indicates that the owner will be left unchanged, implying that the file already exists on the target machine. This field is not used for linked or symbolically linked files.

group The group to which the file belongs (for example, `bin` or `sys`). The field is limited to 14 characters in length. A question mark (`?`) indicates that the group will be left unchanged, implying that the file already exists on the target

machine. This field is not used for linked or symbolically linked files.

Examples

EXAMPLE 1 Basic Usage

The following example shows the use of `installf`, invoked from an optional pre-install or post-install script:

```
# create /dev/xt directory
# (needs to be done before drvinstall)
installf $PKGINST /dev/xt d 755 root sys ||
    exit 2
majno=‘/usr/sbin/drvinstall -m /etc/master.d/xt
     -d $BASEDIR/data/xt.o -v1.0‘ ||
    exit 2
i=00
while [ $i –lt $limit ]
do
    for j in 0 1 2 3 4 5 6 7
    do
        echo /dev/xt$i$j c $majno ‘expr $i ? 8 + $j‘
             644 root sys |
        echo /dev/xt$i$j=/dev/xt/$i$j
    done
    i=‘expr $i + 1‘
    [ $i –le 9 ] && i="0$i" #add leading zero
done | installf $PKGINST – || exit 2
# finalized installation, create links
installf -f $PKGINST || exit 2
```

Environment Variables

`installf` inherits the value of the following environment variable. This variable is set when `pkgadd(1M)` or `pkgrm(1M)` is invoked with the `-R` option.

`PKG_INSTALL_ROOT` — If present, defines the full path name of a directory to use as the system's `PKG_INSTALL_ROOT` path. All product and package information files are then looked for in the directory tree, starting with the specified `PKG_INSTALL_ROOT` path. If not present, the default system path of `/` is used.

Exit Status

`0` — Successful operation.

`>0` — An error occurred.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also pkginfo(1), pkgmk(1), pkgparam(1), pkgproto(1), pkgtrans(1), pkgadd(1M), pkgask(1M), pkgchk(1M), pkgrm(1M), removef(1M), pkgmap(4), space(4), attributes(5), largefile(5)

Application Packaging Developer's Guide

Notes When *ftype* is specified, all applicable fields, as shown below, must be defined:

ftype	Required Fields
p, x, d, f, v, or e	`mode owner group`
c or b	`major minor mode owner group`

The `installf` command will create directories, named pipes and special devices on the original invocation. Links are created when `installf` is invoked with the `-f` option to indicate installation is complete.

Links should be specified as *path1=path2*. *path1* indicates the destination and *path2* indicates the source file.

Files installed with `installf` will be placed in the class `none`, unless a class is defined with the command. Subsequently, they will be removed when the associated package is deleted. If this file should not be deleted at the same time as the package, be certain to assign it to a class which is ignored at removal time. If special action is required for the file before removal, a class must be defined with the command and an appropriate class action script delivered with the package.

When classes are used, `installf` must be used in one of the following forms:

```
installf -c class1 . . .
installf -f -c class1 . . .
installf -c class2 . . .
installf -f -c class2 . . .
```

Name installgrub – install GRUB in a disk partition

Synopsis `/usr/sbin/installgrub [-fm] stage1 stage2 raw-device`

Description The `installgrub` command is an x86–only program. GRUB stands for GRand Unified Bootloader. `installgrub` is deprecated, as it applies to the GRUB Legacy boot loader, which was the boot loader present in Oracle Solaris 11 11/11 and earlier revisions. To install the boot loader, see the `bootadm(1M)` `install-bootloader` subcommand.

`installgrub` installs GRUB Legacy stage 1 and stage 2 files on the boot area of a disk partition. If you specify the `-m` option, `installgrub` installs the stage 1 file onto first sector (the master boot sector [MBR]) of the disk.

Options The `installgrub` command accepts the following options:

`-f` Suppresses interaction when overwriting the master boot sector.

`-m` Installs GRUB *stage1* on the master boot sector interactively. You must use this option if Solaris is installed on an extended partition.

Operands The `installgrub` command accepts the following operands:

stage1 The name of the GRUB stage 1 file.

stage2 The name of the GRUB stage 2 file.

raw-device The name of the device onto which GRUB code is to be installed. It must be a character device that is readable and writable. For disk devices, specify the slice where the GRUB menu file is located. (For Solaris it is the root slice.)

Examples **EXAMPLE 1** Installing GRUB on a Hard Disk Slice

The following command installs GRUB on a system where the root slice is `c0d0s0`:

```
example# /usr/sbin/installgrub /boot/grub/stage1 \
         /boot/grub/stage2 /dev/rdsk/c0d0s0
```

Files `/boot/grub` Directory where GRUB files reside.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os
Interface Stability	Uncommitted

See Also `boot(1M)`, `bootadm(1M)`, `fdisk(1M)`, `fmthard(1M)`, `kernel(1M)`, `attributes(5)`

Warnings Installing GRUB on the master boot sector (`-m` option) overrides any boot manager currently installed on the machine. The system will always boot the GRUB in the Solaris partition regardless of which `fdisk` partition is active.

Do *not* use the `installgrub` command to install the boot loader on systems that have GRUB 2 installed, otherwise you can render the system unbootable. GRUB Legacy should be reinstalled with the installgrub command only *after* you have verified that the version of GRUB Legacy you are installing supports the ZFS pool version of your ZFS root pool, and that you no longer have any Solaris boot environments present that use GRUB2 as their boot loader.

Name in.stdiscover – Service Tag Discovery Daemon

Synopsis

```
/usr/lib/inet/in.stdiscover
```

Description The `in.stdiscover` daemon allows a mechanism for discovering the location of the Service Tag Listener. By default, the `in.stdiscover` daemon listens for discovery probes (using a minimal built-in protocol) on UDP port 6481.

The daemon is under control of the service management facility, `smf(5)`, under its `inetd` framework. It only runs upon demand and exits when no longer in use.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/management/service-tag
Interface Stability	Private

See Also `in.stlisten(1M)`, `stclient(1M)`

Name in.stlisten – Service Tag Listener

Synopsis

```
/usr/lib/inet/in.stlisten
```

Description The `in.stlisten` daemon allows a mechanism for discovering the location of the Service Tag. By default, the `in.stlisten` daemon listens for discovery probes (using a minimal built-in protocol) on TCP port 6481.

The daemon is under control of the service management facility, `smf(5)`, under its `inetd` framework. It only runs upon demand and exits when no longer in use.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/management/service-tag
Interface Stability	Private

See Also `in.stdiscover(1M)`, `stclient(1M)`, `svccfg(1M)`, `attributes(5)`, `environ(5)`, `smf(5)`

Notes In open networks where the participants may not always be trusted, it is recommended that you deploy this daemon with the `passphrase-encryption` option. In `smf(5)` environments, the following command can be used to set the *passphrase*:

```
< prepare a text file "passfile" containing the passphrase >
# chown svctag:daemon passfile
# chmod 600 passfile
svccfg -s svc:/network/stlisten \
        setprop servicetag/passphrase=passfile
```

where *passfile* is the path of a file containing the intended *passphrase*.

This *passphrase* can be subsequently cleared as follows:

```
svccfg -s svc:/network/stlisten \
        setprop servicetag/passphrase=""
```

Name in.talkd, talkd – server for talk program

Synopsis

```
in.talkd
```

Description `talkd` is a server used by the `talk(1)` program. It listens at the UDP port indicated in the "talk" service description; see `services(4)`. The actual conversation takes place on a TCP connection that is established by negotiation between the two machines involved.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	service/network/network-servers

See Also `svcs(1)`, `talk(1)`, `inetadm(1M)`, `inetd(1M)`, `svcadm(1M)`, `services(4)`, `attributes(5)`, `smf(5)`

Notes The protocol is architecture dependent.

The `in.talkd` service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/network/talk
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. Responsibility for initiating and restarting this service is delegated to `inetd(1M)`. Use `inetadm(1M)` to make configuration changes and to view configuration information for this service. The service's status can be queried using the `svcs(1)` command.

Name in.telnetd, telnetd – DARPA TELNET protocol server

Synopsis

```
/usr/sbin/in.telnetd [-a authmode] [-EXUh] [-s tos]
     [-S keytab] [-M realm]
```

Description `in.telnetd` is a server that supports the DARPA standard TELNET virtual terminal protocol. `in.telnetd` is normally invoked in the internet server (see `inetd(1M)`), for requests to connect to the TELNET port as indicated by the `/etc/services` file (see `services(4)`).

`in.telnetd` operates by allocating a pseudo-terminal device for a client, then creating a login process which has the slave side of the pseudo-terminal as its standard input, output, and error. `in.telnetd` manipulates the master side of the pseudo-terminal, implementing the TELNET protocol and passing characters between the remote client and the login process.

When a TELNET session starts up, `in.telnetd` sends TELNET options to the client side indicating a willingness to do *remote* `echo` of characters, and to *suppress go ahead*. The pseudo-terminal allocated to the client is configured to operate in "cooked" mode, and with XTABS, ICRNL and ONLCR enabled. See `termio(7I)`.

`in.telnetd` is willing to do: *echo*, *binary*, *suppress go ahead*, and *timing mark*. `in.telnetd` is willing to have the remote client do: *binary*, *terminal type*, *terminal size*, *logout option*, and *suppress go ahead*.

`in.telnetd` also allows environment variables to be passed, provided that the client negotiates this during the initial option negotiation. The DISPLAY environment variable may be sent this way, either by the TELNET general environment passing methods, or by means of the XDISPLOC TELNET option. DISPLAY can be passed in the environment option during the same negotiation where XDISPLOC is used. Note that if you use both methods, use the same value for both. Otherwise, the results may be unpredictable.

These options are specified in Internet standards *RFC 1096*, *RFC 1408*, *RFC 1510*, *RFC 1571*, *RFC 2941*, *RFC 2942*, *RFC 2946*, and *RFC 1572*. The following Informational draft is also supported: *RFC 2952*.

The banner printed by `in.telnetd` is configurable. The default is (more or less) equivalent to ‘`uname -sr`‘ and will be used if no banner is set in `/etc/default/telnetd`. To set the banner, add a line of the form

```
BANNER="..."
```

to `/etc/default/telnetd`. Nonempty banner strings are fed to shells for evaluation. The default banner may be obtained by

```
BANNER="\\r\\n\\r\\n‘uname -s‘ ‘uname -r‘\\r\\n\\r\\n"
```

and no banner will be printed if `/etc/default/telnetd` contains

```
BANNER=""
```

Options The following options are supported:

`-a` *authmode* This option may be used for specifying what mode should be used for authentication. There are several valid values for *authmode*:

- `valid` Only allows connections when the remote user can provide valid authentication information to identify the remote user, and is allowed access to the specified account without providing a password.
- `user` Only allows connections when the remote user can provide valid authentication information to identify the remote user. The `login(1)` command will provide any additional user verification needed if the remote user is not allowed automatic access to the specified account.
- `none` This is the default state. Authentication information is not required. If no or insufficient authentication information is provided, then the `login`(1) program provides the necessary user verification.
- `off` This disables the authentication code. All user verification happens through the `login`(1) program.

`-E` Disables encryption support negotiation.

`-h` Disables displaying host specific information before login has been completed.

`-M` *realm* Uses the indicated Kerberos V5 realm. By default, the daemon will determine its realm from the settings in the `krb5.conf(4)` file.

`-s` *tos* Sets the `IP TOS` option.

`-S` *keytab* Sets the `KRB5` keytab file to use. The `/etc/krb5/krb5.keytab` file is used by default.

`-U` Refuses connections that cannot be mapped to a name through the `getnameinfo(3SOCKET)` function.

`-X` Disables Kerberos V5 authentication support negotiation.

Usage `telnetd` and `in.telnetd` are IPv6–enabled. See `ip6(7P)`.

Security `in.telnetd` can authenticate using Kerberos V5 authentication, `pam(3PAM)`, or both. By default, the telnet server will accept valid Kerberos V5 authentication credentials from a `telnet` client that supports Kerberos. `in.telnetd` can also support an encrypted session from such a client if the client requests it.

The telnet protocol only uses single DES for session protection—clients request service tickets with single DES session keys. The KDC must know that host service principals that offer the telnet service support single DES, which, in practice, means that such principals must have single DES keys in the KDC database.

In order for Kerberos authentication to work, a host/<*FQDN*> Kerberos principal must exist for each Fully Qualified Domain Name associated with the telnetd server. Each of these host/<*FQDN*> principals must have a keytab entry in the /etc/krb5/krb5.keytab file on the telnetd server. An example principal might be:

```
host/bigmachine.eng.example.com
```

If Kerberized telnet(1) must be used, allow_weak_keys = true must be set in krb5.conf(4) on all systems involved (KDC, client, and server), as DES is hardcoded into the protocol.

See kadmin(1M) or gkadmin(1M) for instructions on adding a principal to a krb5.keytab file. See *Oracle Solaris 11.1 Administration: Security Services* for a discussion of Kerberos authentication.

in.telnetd uses pam(3PAM) for authentication, account management, session management, and password management. The PAM configuration policy, configured in /etc/pam.conf or per-service files in /etc/pam.d/, specifies the modules to be used for in.telnetd. Here is a partial pam.conf file with entries for the telnet command using the UNIX authentication, account management, session management, and password management modules.

```
telnet  auth requisite          pam_authtok_get.so.1
telnet  auth required           pam_dhkeys.so.1
telnet  auth required           pam_unix_auth.so.1

telnet  account requisite       pam_roles.so.1
telnet  account required        pam_projects.so.1
telnet  account required        pam_unix_account.so.1

telnet  session required        pam_unix_session.so.1

telnet  password required       pam_dhkeys.so.1
telent  password requisite      pam_authtok_get.so.1
telnet  password requisite      pam_authtok_check.so.1
telnet  password required       pam_authtok_store.so.1
```

The equivalent PAM configuration using /etc/pam.d/ would be the following entries in /etc/pam.d/telnet:

```
auth requisite          pam_authtok_get.so.1
auth required           pam_dhkeys.so.1
auth required           pam_unix_auth.so.1
account requisite       pam_roles.so.1
account required                pam_projects.so.1
```

```
account required                pam_unix_account.so.1

session required                pam_unix_session.so.1

password        required        pam_dhkeys.so.1
password        requisite       pam_authtok_get.so.1
password        requisite       pam_authtok_check.so.1
password        required        pam_authtok_store.so.1
```

If there are no entries for the `telnet` service in `/etc/pam.conf` and `/etc/pam.d/telnet` does not exist, then the entries for the "other" service in `/etc/pam.conf` will be used. If there are not any entries in `/etc/pam.conf` for the "other" service then the entries in `/etc/pam.d/other` will be used. If multiple authentication modules are listed, then the user may be prompted for multiple passwords.

For a Kerberized telnet service, the correct PAM service name is `ktelnet`.

Files `/etc/default/telnetd`

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	service/network/telnet

See Also `login(1)`, `svcs(1)`, `telnet(1)`, `gkadmin(1M)`, `inetadm(1M)`, `inetd(1M)`, `kadmin(1M)`, `svcadm(1M)`, `pam(3PAM)`, `getnameinfo(3SOCKET)`, `issue(4)`, `krb5.conf(4)`, `pam.conf(4)`, `services(4)`, `attributes(5)`, `pam_authtok_check(5)`, `pam_authtok_get(5)`, `pam_authtok_store(5)`, `pam_dhkeys(5)`, `pam_passwd_auth(5)`, `pam_unix_account(5)`, `pam_unix_auth(5)`, `pam_unix_session(5)`, `smf(5)`, `ip6(7P)`, `termio(7I)`

Oracle Solaris 11.1 Administration: Security Services

Alexander, S. *RFC 1572, TELNET Environment Option*. Network Information Center, SRI International, Menlo Park, Calif., January 1994.

Borman, Dave. *RFC 1408, TELNET Environment Option*. Network Information Center, SRI International, Menlo Park, Calif., January 1993.

Borman, Dave. *RFC 1571, TELNET Environment Option Interoperability Issues*. Network Information Center, SRI International, Menlo Park, Calif., January 1994.

Crispin, Mark. *RFC 727, TELNET Logout Option*. Network Information Center, SRI International, Menlo Park, Calif., April 1977.

Marcy, G. *RFC 1096, TELNET X Display Location Option*. Network Information Center, SRI International, Menlo Park, Calif., March 1989.

Postel, Jon, and Joyce Reynolds. *RFC 854, TELNET Protocol Specification*. Network Information Center, SRI International, Menlo Park, Calif., May 1983.

Waitzman, D. *RFC 1073, TELNET Window Size Option*. Network Information Center, SRI International, Menlo Park, Calif., October 1988.

Kohl, J., Neuman, C., *The Kerberos Network Authentication Service (V5), RFC 1510*. September 1993.

Ts'o, T. and J. Altman, *Telnet Authentication Option, RFC 2941*. September 2000.

Ts'o, T., *Telnet Authentication: Kerberos Version 5, RFC 2942*. September 2000.

Ts'o, T., *Telnet Data Encryption Option, RFC 2946*. September 2000.

Ts'o, T., *Telnet Encryption: DES 64 bit Cipher Feedback, RFC 2952*. September 2000.

Notes Some TELNET commands are only partially implemented.

Binary mode has no common interpretation except between similar operating systems.

The terminal type name received from the remote client is converted to lower case.

The *packet* interface to the pseudo-terminal should be used for more intelligent flushing of input and output queues.

`in.telnetd` never sends TELNET *go ahead* commands.

The `in.telnetd` service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/network/telnet
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. Responsibility for initiating and restarting this service is delegated to `inetd(1M)`. Use `inetadm(1M)` to make configuration changes and to view configuration information for this service. The service's status can be queried using the `svcs(1)` command.

Name in.tftpd, tftpd – Internet Trivial File Transfer Protocol server

Synopsis

```
in.tftpd [-d] [-T rexmtval] [-s] [homedir]
```

Description `tftpd` is a server that supports the Internet Trivial File Transfer Protocol (TFTP).

Before responding to a request, the server attempts to change its current directory to *homedir*; the default directory is `/tftpboot`.

The use of `tftp` does not require an account or password on the remote system. Due to the lack of authentication information, `in.tftpd` will allow only publicly readable files to be accessed. Files may be written only if they already exist and are publicly writable. Note that this extends the concept of "public" to include all users on all hosts that can be reached through the network. This may not be appropriate on all systems, and its implications should be considered before enabling this service.

`in.tftpd` runs with the user ID and group ID set to `[GU]ID_NOBODY` under the assumption that no files exist with that owner or group. However, nothing checks this assumption or enforces this restriction.

Options

`-d` Debug. When specified it sets the `SO_DEBUG` socket option.

`-s` Secure. When specified, the directory change to *homedir* must succeed. The daemon also changes its root directory to *homedir*.

`-T` *rexmtval* Specifies the value of the retransmission timeout in seconds. This also affects the maximum session timeout in that the latter is set to five times the retransmission timeout value.

Usage The `in.tftpd` server is IPv6–enabled. See `ip6(7P)`.

`in.tftpd` supports transfers of greater than 32 MB, per RFC 2348.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	service/network/tftp

See Also `svcs(1)`, `tftp(1)`, `inetadm(1M)`, `inetd(1M)`, `svcadm(1M)`, `netconfig(4)`, `attributes(5)`, `smf(5)`, `ip6(7P)`

Malkin, G. and Harkin, A. *RFC 2347, TFTP Option Extension*. The Internet Society. May 1998

Malkin, G. and Harkin, A. *RFC 2348, TFTP Blocksize Option*. The Internet Society. May 1998

Malkin, G. and Harkin, A. *RFC 2349, TFTP Timeout Interval and Transfer Size Options*. The Internet Society. May 1998

Sollins, K.R. *RFC 1350, The TFTP Protocol (Revision 2)*. Network Working Group. July 1992.

Notes The `tftpd` server only acknowledges the transfer size option that is sent with a `read` request when the octet transfer mode is specified.

The `in.tftpd.1m` service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/network/tftp/udp6:default
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. Responsibility for initiating and restarting this service is delegated to `inetd(1M)`. Use `inetadm(1M)` to make configuration changes and to view configuration information for this service. The service's status can be queried using the `svcs(1)` command.

Unlike most `smf(5)` services, a manifest for the `tftp` service is not included in the system. To create one and enable this service, the administrator should:

1. Edit `/etc/inet/inetd.conf` and uncomment the `tftp` entry.
2. Run `/usr/sbin/inetconv`.

After you run `inetconv`, the `svc:/network/tftp/udp6:default` service is created and enabled.

Name in.timed – UDP or TCP time protocol service daemon

Synopsis

```
in.timed

FMRI svc:/internet/time:default
```

Description FMRI stands for Fault Management Resource Identifier. It is used to identify resources managed by the Fault Manager. See `fmd(1M)` and `smf(5)`.

The `in.timed` service provides the server-side of the time protocol. The time server sends to requestors the time in seconds since midnight, January 1, 1900. The time protocol is available on both TCP and UDP transports through port 37.

The in.timed service is an `inetd(1M)` `smf(5)` delegated service. The `in.timed` detects which transport is requested by examining the socket it is passed by the `inetd` daemon.

TCP-based service — Once a connection is established, the `in.timed` sends the time as a 32-bit binary number and closes the connection. Any received data is ignored.

UDP-based service — The `in.timed` listens for UDP datagrams. When a datagram is received, the server generates a UDP datagram containing the time as a 32–bit binary number and sends it to the client. Any received data is ignored.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	service/network/legacy-network-services
Interface Stability	Committed

See Also `inetd(1M)`, `attributes(5)`, `smf(5)`

RFC 868

Name intrd – interrupt distribution daemon

Synopsis

```
intrd
```

Description The `intrd` daemon is started at boot time to monitor the assignments between interrupts and CPUs. If `intrd` decides that the current assignments are imbalanced and harmful to system performance, it will generate and implement new assignments.

Any notifications will be delivered via `syslogd(1M)`.

Because `intrd` dynamically monitors a system for optimal performance, it consumes a small amount of CPU time, even on an otherwise idle system. This behavior is normal.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/kernel/power
Interface Stability	Uncommitted

See Also `svcs(1)`, `svcadm(1M)`, `syslogd(1M)`, `attributes(5)`, `smf(5)`

Notes The interrupt distribution daemon is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/system/intrd:default
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. The service's status can be queried using the `svcs(1)` command.

Name intrstat – report interrupt statistics

Synopsis

```
/usr/sbin/intrstat [-c cpulist | -C processor_set_id] [-T u | d ]
     [-x opt[=val]] [interval [count]]
```

Description The `intrstat` utility gathers and displays run-time interrupt statistics. The output is a table of device names and CPU IDs, where each row of the table denotes a device, and each column of the table denotes a CPU. Each cell in the table contains both the raw number of interrupts for the given device on the given CPU, and the percentage of absolute time spent in that device's interrupt handler on that CPU.

The device name is given in the form of {*name*}#{*instance*}. The name is the normalized driver name, and typically corresponds to the name of the module implementing the driver. See `ddi_driver_name(9F)`. Many Sun-delivered drivers have their own manual pages. See `Intro(7)`.

If standard output is a terminal, the table contains as many columns of data as can fit within the terminal width. If standard output is not a terminal, the table contains at most four columns of data. By default, data is gathered and displayed for all CPUs. If the data cannot fit in a single table, it is printed across multiple tables. The set of CPUs for which data is displayed can be optionally specified with the `-c` or `-C` option.

By default, `intrstat` displays data once per second and runs indefinitely. Both of these behaviors can be optionally controlled with the interval and count parameters, respectively. See `OPERANDS`.

Because `intrstat` uses dynamic discovery, it reports only on devices that raise interrupts while the command is running. Any devices that are silent while `intrstat` is running are not displayed.

`intrstat` induces a small system-wide performance degradation. As a result, only the super-user can run `intrstat` by default. The *Solaris Dynamic Tracing Guide* explains how administrators can grant privileges to other users to permit them to run `intrstat`.

Options The following options are supported:

`-c` *cpulist*
Displays data for the CPUs specified by *cpulist*.

cpulist can be a single processor ID (for example, `4`), a range of processor IDs (for example, `4-6`), or a comma separated list of processor IDs or processor ID ranges (for example, `4,5,6` or `4,6-8`).

`-C` *processor_set_id*
Displays data for the CPUs in the processor set specified by *processor_set_id*.

`intrstat` modifies its output to always reflect the CPUs in the specified processor set. If a CPU is added to the set, `intrstat` modifies its output to include the added CPU. If a CPU is removed from the set, `intrstat` modifies its output to exclude the removed CPU. At most one processor set can be specified.

`-T u | d`
Display a time stamp.

Specify `u` for a printed representation of the internal representation of time. See `time(2)`. Specify `d` for standard date format. See `date(1)`.

`-x` *opt*[*=val*]
Enable or modify a DTrace runtime option or D compiler option. The list of options is found in the *Solaris Dynamic Tracing Guide*. A boolean option is enabled by specifying its name. Options with values are set by separating the option name and value with an equal sign (=)

Operands The following operands are supported:

count
Indicates the number of times `intrstat` will display its output before exiting.

interval
Indicates the number of seconds between displays of `intrstat` output.

Examples **EXAMPLE 1** Using `intrstat` Without Options

Without options, `intrstat` displays a table of trap types and CPUs. At most, four columns can fit in the default terminal width. If there are more than four CPUs, multiple tables are displayed.

The following example runs `intrstat` on a uniprocessor Intel IA/32-based laptop:

```
example# intrstat
               device |      cpu0 %tim
     -----------------+--------------
                ata#0 |       166  0.4
                ata#1 |         0  0.0
          audioi810#0 |         6  0.0
              i8042#0 |       281  0.7
               iprb#0 |         6  0.0
               uhci#1 |         6  0.0
               uhci#2 |         6  0.0

               device |      cpu0 %tim
     -----------------+--------------
                ata#0 |       161  0.5
                ata#1 |         0  0.0
          audioi810#0 |         6  0.0
              i8042#0 |       303  0.6
```

EXAMPLE 1 Using `intrstat` Without Options *(Continued)*

```
              iprb#0 |          6  0.0
              uhci#1 |          6  0.0
              uhci#2 |          6  0.0
     ...
```

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/dtrace
Interface Stability	See below.

The command-line syntax is Committed. The human-readable output is Uncommitted.

See Also `dtrace(1M)`, `trapstat(1M)`, `attributes(5)`, `Intro(7)`, `ddi_driver_name(9F)`

Solaris Dynamic Tracing Guide

Name in.uucpd, uucpd – UUCP server

Synopsis

```
/usr/sbin/in.uucpd [-n]
```

Description `in.uucpd` is the server for supporting UUCP connections over networks.

`in.uucpd` is invoked by `inetd(1M)` when a UUCP connection is established, that is, a connection to the port indicated in the "uucp" service specification, and executes the following protocol. See `services(4)`:

1. The server prompts with `login:`. The `uucico(1M)` process at the other end must supply a username.
2. Unless the username refers to an account without a password, the server then prompts with `Password:`. The `uucico` process at the other end must supply the password for that account.

If the username is not valid, or is valid but refers to an account that does not have `/usr/lib/uucp/uucico` as its login shell, or if the password is not the correct password for that account, the connection is dropped. Otherwise, `uucico` is run, with the user ID, group ID, group set, and home directory for that account, with the environment variables `USER` and `LOGNAME` set to the specified username, and with a `-u` flag specifying the username. Unless the `-n` flag is specified, entries are made in `/var/adm/utmpx`, `/var/adm/wtmpx`, and `/var/adm/lastlog` for the username. `in.uucpd` must be invoked by a user with appropriate privilege (usually root) in order to be able to verify that the password is correct.

Security `in.uucpd` uses `pam(3PAM)` for authentication, account management, and session management. The PAM configuration policy, listed through `/etc/pam.conf`, specifies the modules to be used for `in.uucpd`. Here is a partial `pam.conf` file with entries for `uucp` using the UNIX authentication, account management, and session management module.

```
uucp    auth requisite          pam_authtok_get.so.1
uucp    auth required           pam_dhkeys.so.1
uucp    auth required           pam_unix_auth.so.1

uucp    account requisite       pam_roles.so.1
uucp    account required        pam_projects.so.1
uucp    account required        pam_unix_account.so.1

uucp      session required      pam_unix_session.so.1
```

If there are no entries for the `uucp` service, then the entries for the "other" service will be used. If multiple authentication modules are listed, then the peer may be prompted for multiple passwords.

Files

`/var/adm/utmpx`	accounting
`/var/adm/wtmpx`	accounting
`/var/adm/lastlog`	time of last login

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	service/network/uucp

See Also svcs(1), inetadm(1M), inetd(1M), svcadm(1M), uucico(1M), pam(3PAM), pam.conf(4), services(4), attributes(5), pam_authtok_check(5), pam_authtok_get(5), pam_authtok_store(5), pam_dhkeys(5), pam_passwd_auth(5), pam_unix_account(5), pam_unix_auth(5), pam_unix_session(5), smf(5)

Diagnostics All diagnostic messages are returned on the connection, after which the connection is closed.

`user read` An error occurred while reading the username.

`passwd read` An error occurred while reading the password.

`Login incorrect.` The username is invalid or refers to an account with a login shell other than `/usr/lib/uucp/uucico`, or the password is not the correct password for the account.

Notes The `in.uucpd` service is managed by the service management facility, smf(5), under the service identifier:

```
svc:/network/uucp
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using svcadm(1M). Responsibility for initiating and restarting this service is delegated to inetd(1M). Use inetadm(1M) to make configuration changes and to view configuration information for this service. The service's status can be queried using the svcs(1) command.

Name iostat – report I/O statistics

Synopsis

```
/usr/bin/iostat  [-cCdDeEiImMnpPrstxXYz] [-l n] [-T u | d]
 [disk]... [interval [count]]
```

Description The `iostat` utility iteratively reports terminal, disk, and tape I/O activity, as well as CPU utilization. The first line of output is for all time since boot; each subsequent line is for the prior interval only.

To compute this information, the kernel maintains a number of counters. For each disk, the kernel counts reads, writes, bytes read, and bytes written. The kernel also takes hi-res time stamps at queue entry and exit points, which allows it to keep track of the residence time and cumulative residence-length product for each queue. Using these values, `iostat` produces highly accurate measures of throughput, utilization, queue lengths, transaction rates and service time. For terminals collectively, the kernel simply counts the number of input and output characters.

During execution of the kernel status command, the `state` of the system can change. If relevant, a state change message is included in the `iostat` output, in one of the following forms:

```
<<device added: sd0>>
<<device removed: sd0>>
<<partition added: sd0,a>>
<<partition removed: sd0,a>>
<<NFS mounted: nfs1>>
<<NFS unmounted: nfs1>>
<<multi-path added: ssd4>>
<<multi-path removed: ssd4>>
<<controller added: c1>>
<<controller removed: c1>>
<<processors added: 1, 3>>
<<processors removed: 1, 3>>
```

Note that the `names` printed in these state change messages are affected by the `-n` and `-m` options as appropriate.

For more general system statistics, use `sar(1)`, `sar(1M)`, or `vmstat(1M)`.

Output The output of the `iostat` utility includes the following information.

`device` name of the disk

`r/s` reads per second

`w/s` writes per second

`kr/s` kilobytes read per second

The average I/O size during the interval can be computed from `kr/s` divided by `r/s`.

`kw/s` kilobytes written per second

The average I/O size during the interval can be computed from `kw/s` divided by `w/s`.

`wait` average number of transactions waiting for service (queue length)

This is the number of I/O operations held in the device driver queue waiting for acceptance by the device.

`actv` average number of transactions actively being serviced (removed from the queue but not yet completed)

This is the number of I/O operations accepted, but not yet serviced, by the device.

`svc_t` average response time of transactions, in milliseconds

The `svc_t` output reports the overall *response* time, rather than the *service* time, of a device. The overall time includes the time that transactions are in queue and the time that transactions are being serviced. The time spent in queue is shown with the `-x` option in the `wsvc_t` output column. The time spent servicing transactions is the true service time. Service time is also shown with the `-x` option and appears in the `asvc_t` output column of the same report.

`%w` percent of time there are transactions waiting for service (queue non-empty)

`%b` percent of time the disk is busy (transactions in progress)

`wsvc_t` average service time in wait queue, in milliseconds

`asvc_t` average service time of active transactions, in milliseconds

`wt` the I/O wait time is no longer calculated as a percentage of CPU time, and this statistic will always return zero.

Options The following options are supported:

`-c` Report the percentage of time the system has spent in user mode, in system mode, waiting for I/O, and idling. See the NOTES section for more information.

`-C` When the `-x` option is also selected, report extended disk statistics aggregated by *controller id*.

`-d` For each disk, report the number of kilobytes transferred per second, the number of transfers per second, and the average service time in milliseconds.

`-D` For each disk, report the reads per second, writes per second, and percentage disk utilization.

`-e` Display device error summary statistics. The total errors, hard errors, soft errors, and transport errors are displayed.

`-E` Display all device error statistics.

`-i` In `-E` output, display the `Device ID` instead of the `Serial No`. The `Device Id` is a unique identifier registered by a driver through `ddi_devid_register(9F)`.

`-I` Report the counts in each interval, rather than rates (where applicable).

`-l` *n* Limit the number of disks included in the report to *n*; the disk limit defaults to 4 for `-d` and `-D`, and unlimited for `-x`. Note: disks explicitly requested (see *disk* below) are not subject to this disk limit.

`-m` Report file system mount points. This option is most useful if the `-P` or `-p` option is also specified or used in conjunction with `-Xn` or `-en`. The `-m` option is useful only if the mount point is actually listed in the output. This option can only be used in conjunction with the `-n` option.

`-M` Display data throughput in MB/sec instead of KB/sec.

`-n` Display names in descriptive format. For example, `cXtYdZ`, `rmt/N`, `server:/export/path`.

By default, disks are identified by instance names such as `ssd23` or `md301`. Combining the `-n` option with the `-x` option causes disk names to display in the `cXtYdZsN` format which is more easily associated with physical hardware characteristics. The `cXtYdZsN` format is particularly useful in FibreChannel (FC) environments where the FC World Wide Name appears in the `t` field.

`-p` For each disk, report per-partition statistics in addition to per-device statistics.

`-P` For each disk, report per-partition statistics only, no per-device statistics.

`-r` Display data in a comma-separated format.

`-s` Suppress messages related to `state changes`.

`-t` Report the number of characters read and written to terminals per second.

`-T u | d` Display a time stamp.

Specify `u` for a printed representation of the internal representation of time. See `time(2)`. Specify `d` for standard date format. See `date(1)`.

`-X` For disks under `scsi_vhci(7D)` control, in addition to disk *lun* statistics, also report statistics for *lun.controller*.

`-x` Report extended disk statistics. By default, disks are identified by instance names such as `ssd23` or `md301`. Combining the `x` option with the `-n` option causes disk names to display in the `cXtYdZsN` format, more easily associated with physical

hardware characteristics. Using the `cXtYdZsN` format is particularly helpful in the FibreChannel environments where the FC World Wide Name appears in the `t` field.

If no output display is requested (no `-x`, `-e`, `-E`), `-x` is implied.

`-Y` For disks under `scsi_vhci(7D)` control, in addition to disk *lun* statistics, also report statistics for *lun.targetport* and *lun.targetport.controller*.

In `-n` (descriptive) mode the *targetport* is shown in using the `target-port` property of the path. Without `-n` the *targetport* is shown using the shorter `port-id`. All target ports with the same `target-port` property value share the same `port-id`. The `target-port`-to-`port-id` association does not persist across reboot.

If no output display is requested (no `-x`, `-e`, `-E`), `-x` is implied.

`-z` Do not print lines whose underlying data values are all zeros.

The option set `-xcnCXTdz` *interval* is particularly useful for determining whether disk I/O problems exist and for identifying problems.

Operands

The following operands are supported:

count Display only *count* reports.

disk Explicitly specify the disks to be reported; in addition to any explicit disks, any active disks up to the disk limit (see `-l` above) will also be reported.

interval Report once each *interval* seconds.

Examples

EXAMPLE 1 Using `iostat` to Generate User and System Operation Statistics

The following command displays two reports of extended device statistics, aggregated by *controller id*, for user (`us`) and system (`sy`) operations. Because the `-n` option is used with the `-x` option, devices are identified by controller names.

```
example% iostat -xcnCXTdz 5


Mon Nov 24 14:58:36 2003
    cpu
 us sy wt id
 14 31  0 20
                    extended device statistics
  r/s    w/s    kr/s      kw wait  actv wsvc_t asvc_t  %w  %b device
  3.8   29.9   145.8    44.0  0.0   0.2    0.1    6.4   0   5      c0
666.3  814.8 12577.6 17591.1 91.3  82.3   61.6   55.6   0   2    c12
180.0  234.6  4401.1  5712.6  0.0 147.7    0.0  356.3   0  98    d10
```

EXAMPLE 1 Using `iostat` to Generate User and System Operation Statistics *(Continued)*

```
Mon Nov 24 14:58:41 2003
    cpu
 us sy wt id
 11 31  0 22
                    extended device statistics
  r/s    w/s    kr/s      kw wait  actv wsvc_t asvc_t  %w  %b device
  0.8   41.0     5.2    20.5 0.0    0.2    0.2    4.4   0   6     c0
565.3  581.7  8573.2 10458.9 0.0   26.6    0.0   23.2   0   3    c12
106.5   81.3  3393.2  1948.6 0.0    5.7    0.0   30.1   0  99    d10
```

EXAMPLE 2 Using `iostat` to Generate TTY Statistics

The following command displays two reports on the activity of five disks in different modes of operation. Because the `-x` option is used, disks are identified by instance names.

```
example% iostat -x tc 5 2
```

```
                 extended device statistics        tty         cpu
device r/s  w/s kr/s  kw/s wait actv svc_t %w  %b  tin tout  us sy wt id
sd0    0.4  0.3 10.4   8.0  0.0  0.0  36.9  0   1    0   10   0  0  0 99
sd1    0.0  0.0  0.3   0.4  0.0  0.0  35.0  0   0
sd6    0.0  0.0  0.0   0.0  0.0  0.0   0.0  0   0
nfs1   0.0  0.0  0.0   0.0  0.0  0.0   0.0  0   0
nfs2   0.0  0.0  0.0   0.1  0.0  0.0  35.6  0   0
            extended device statistics              tty         cpu
device r/s  w/s  kr/s  kw/s wait actv svc_t %w  %b tin tout  us sy wt id
sd0    0.0  0.0  0.0   0.0  0.0  0.0  0.0   0   0   0  155   0  0  0 100
sd1    0.0  0.0  0.0   0.0  0.0  0.0  0.0   0   0
sd6    0.0  0.0  0.0   0.0  0.0  0.0  0.0   0   0
nfs1   0.0  0.0  0.0   0.0  0.0  0.0  0.0   0   0
nfs2   0.0  0.0  0.0   0.0  0.0  0.0  0.0   0   0
```

EXAMPLE 3 Using `iostat` to Generate Partition and Device Statistics

The following command generates partition and device statistics for each disk. Because the `-n` option is used with the `-x` option, disks are identified by controller names.

```
example% iostat -xnp
```

```
                extended device statistics
r/s  w/s  kr/s kw/s wait actv wsvc_t asvc_t %w %b device
0.4  0.3  10.4  7.9  0.0  0.0    0.0   36.9  0  1 c0t0d0
0.3  0.3   9.0  7.3  0.0  0.0    0.0   37.2  0  1 c0t0d0s0
0.0  0.0   0.1  0.5  0.0  0.0    0.0   34.0  0  0 c0t0d0s1
```

EXAMPLE 3 Using `iostat` to Generate Partition and Device Statistics *(Continued)*

```
0.0  0.0   0.0  0.1  0.0  0.0    0.6   35.0  0  0 fuji:/export/home/user3
```

EXAMPLE 4 Show Translation from Instance Name to Descriptive Name

The following example illustrates the use of `iostat` to translate a specific instance name to a descriptive name.

```
example% iostat -xn sd1
                        extended device statistics
r/s    w/s   kr/s   kw/s wait actv wsvc_t asvc_t  %w  %b device
0.0    0.0    0.0    0.0  0.0  0.0    0.0    0.0   0   0 c8t1d0
```

EXAMPLE 5 Show Target Port and Controller Activity for a Specific Disk

In the following example, there are four controllers, all connected to the same target port.

```
# iostat -Y ssd22
                    extended device statistics
device          r/s    w/s   kr/s   kw/s wait actv  svc_t  %w  %b
ssd22           0.2    0.0    1.5    0.0  0.0  0.0    0.7   0   0
ssd22.t2        0.2    0.0    1.5    0.0  0.0  0.0    0.0   0   0
ssd22.t2.fp0    0.0    0.0    0.4    0.0  0.0  0.0    0.0   0   0
ssd22.t2.fp1    0.0    0.0    0.4    0.0  0.0  0.0    0.0   0   0
ssd22.t2.fp2    0.0    0.0    0.4    0.0  0.0  0.0    0.0   0   0
ssd22.t2.fp3    0.0    0.0    0.4    0.0  0.0  0.0    0.0   0   0
```

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os
Interface Stability	See below.

Invocation is evolving. Human readable output is unstable.

See Also

`date(1)`, `sar(1)`, `sar(1M)`, `mpstat(1M)`, `vmstat(1M)`, `time(2)`, `attributes(5)`, `scsi_vhci(7D)`

Notes

The sum of CPU utilization might vary slightly from 100 because of rounding errors in the production of a percentage figure.

The `svc_t` response time is not particularly significant when the `I/0` (`r/s+w/s`) rates are under 0.5 per second. Harmless spikes are fairly normal in such cases.

The `mpstat` utility reports the same `wt`, `usr`, and `sys` statistics. See `mpstat(1M)` for more information.

When executed in a `zone` and if the pools facility is active, `iostat`(1M) will only provide information for those processors in the processor set of the pool to which the `zone` is bound.

Name ipaddrsel – configure IPv6 default address selection

Synopsis

```
/usr/sbin/ipaddrsel

/usr/sbin/ipaddrsel -f file

/usr/sbin/ipaddrsel -d
```

Description Use the `ipaddrsel` utility to configure the IPv6 default address selection policy table. The policy table is a longest-matching-prefix lookup table that is used for IPv6 source address selection and for destination address ordering when resolving names to `AF_INET6` addresses. For a description of how the policy table is used for source address selection, see `inet6`(7P). For a description of how the policy table is used for destination address ordering, see `getaddrinfo`(3SOCKET).

The unmodified policy table is valid for all typical IPv6 deployments. Modify the table only if a circumstance exists for which the default behavior of the IPv6 source address selection or destination address ordering mechanism is unsatisfactory. See the Examples section for examples of such circumstances. You should carefully consider your addressing strategy before you change the table from the provided default.

When the `ipaddrsel` command is issued without any arguments, the address selection policy currently in use is printed. The format of the output is compatible with the format of the configuration file that the `-f` option accepts.

Note – If the `usesrc` subcommand to `ifconfig`(1M) is applied to a particular physical interface, the selection policy specified by `usesrc` overrides the source address selection policies specified by `ipaddrsel`. This is true for packets that are locally generated and for applications that do not choose a non-zero source address using `bind`(3SOCKET).

The Configuration File

The configuration file that the `-f` option accepts can contain either comment lines or policy entries. Comment lines have a '#' character as the first non-blank character. and they are ignored by the `ipaddrsel` utility. Policy entry lines have the following format:

prefix/*prefix_length* *precedence* *label* [`#` *comment*]

The *prefix* must be an IPv6 prefix in a format consistent with `inet`(3SOCKET). The *prefix_length* is an integer ranging from 0 to 128. The IPv6 source address selection and destination address ordering algorithms determine the precedence or label of an address by doing a longest-prefix-match lookup using the prefixes in this table, much like next-hop determination for a destination is done by doing a longest-prefix-match lookup using an IP routing table.

The precedence is a non-negative integer that represents how the destination address ordering mechanism will sort addresses returned from name lookups. In general, addresses with a higher precedence will be in front of addresses with a lower precedence. Other factors, such as destinations with undesirable source addresses can, however, override these precedence values.

The label is a string of at most fifteen characters, not including the NULL terminator. The label allows particular source address prefixes to be used with destination prefixes of the same label. Specifically, for a particular destination address, the IPv6 source address selection algorithm prefers source addresses whose label is equal that of the destination.

The label may be followed by an optional comment.

The file must contain a default policy entry, which is an entry with `::0/0` as its *prefix* and *prefix_length*. This is to ensure that all possible addresses match a policy.

Options The `ippadrsel` utility supports the following options:

`-f` *file* Replace the address selection policy table with the policy specified in the *file*.

`-d` Revert the kernel's address selection policy table back to the default table. Invoking `ipaddrsel` in this way only changes the currently running kernel's policy table, and does not alter the configuration file `/etc/inet/ipaddrsel.conf`. To revert the configuration file back to its default settings, use `ipaddrsel -d`, then dump the contents of the table to the configuration file by redirecting the output of `ipaddrsel` to `/etc/inet/ipaddrsel.conf`.

```
example# ipaddrsel -d
example# ipaddrsel > /etc/inet/ipaddrsel.conf
```

Examples **EXAMPLE 1** The Default Policy in `/etc/inet/ipaddrsel.conf`

The following example is the default policy that is located in `/etc/inet/ipaddrsel.conf`:

```
# Prefix                        Precedence Label
::1/128                         50 Loopback
::/96                           20 IPv4_Compatible
::ffff:0.0.0.0/96               10 IPv4
2002::/16                       30 6to4
::/0                            40 Default
```

EXAMPLE 2 Assigning a Lower Precedence to Link-local and Site-local Addresses

By default, the destination address ordering rules sort addresses of smaller scope before those of larger scope. For example, if a name resolves to a global and a site-local address, the site local address would be ordered before the global address. An administrator can override this ordering rule by assigning a lower precedence to addresses of smaller scope, as the following table demonstrates.

```
# Prefix                        Precedence Label
::1/128                         50 Loopback
::/0                            40 Default
2002::/16                       30 6to4
fec0::/10                       27 Site-Local
fe80::/10                       23 Link-Local
::/96                           20 IPv4_Compatible
```

EXAMPLE 2 Assigning a Lower Precedence to Link-local and Site-local Addresses *(Continued)*

```
::ffff:0.0.0.0/96                   10 IPv4
```

EXAMPLE 3 Assigning Higher Precedence to IPv4 Destinations

By default, IPv6 addresses are ordered in front of IPv4 addresses in name lookups. `::ffff:0.0.0.0/96` has the lowest precedence in the default table. In the following example, IPv4 addresses are assigned higher precedence and are ordered in front of IPv6 destinations:

```
# Prefix                            Precedence Label
::1/128                             50 Loopback
::/0                                40 Default
2002::/16                           30 6to4
::/96                               20 IPv4_Compatible
::ffff:0.0.0.0/96                   60 IPv4
```

EXAMPLE 4 Ensuring that a Particular Source Address is Used

This example ensures that a particular source address is used only when communicating with destinations in a particular network.

The following policy table assigns a label of 5 to a particular source address on the local system, `2001:1111:1111::1`. The table assigns the same label to a network, `2001:2222:2222::/48`. The result of this policy is that the `2001:1111:1111::1` source address will only be used when communicating with destinations contained in the `2001:2222:2222::/48` network. For this example, this network is the `ClientNet`, which could represent a particular client's network.

```
# Prefix                            Precedence Label
::1/128                             50 Loopback
2001:1111:1111::1/128               40 ClientNet
2001:2222:2222::/48                 40 ClientNet
::/0                                40 Default
2002::/16                           30 6to4
::/96                               20 IPv4_Compatible
::ffff:0.0.0.0/96                   10 IPv4
```

This example assumes that the local system has one physical interface, and that all global prefixes are assigned to that physical interface.

Exit Status

`ipaddrsel` returns the following exit values:

`0` `ipaddrsel` successfully completed.

`>0` An error occurred. If a failure is encountered, the kernel's current policy table is unchanged.

Files `/etc/inet/ipaddrsel.conf` The file that contains the IPv6 default address selection policy to be installed at boot time. This file is loaded before any Internet services are started.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os
Interface Stability	Committed

See Also `nscd(1M)`, `inet(3SOCKET)`, `getaddrinfo(3SOCKET)`, `ipaddrsel.conf(4)`, `attributes(5)`, `inet6(7P)`

Notes The ipnodes cache kept by `nscd(1M)` contains addresses that are ordered using the destination address ordering algorithm, which is one of the reasons why `ipaddrsel` is called before `nscd` in the boot sequence. If `ipaddrsel` is used to change the address selection policy after `nscd` has started, you should invalidate the `nscd` ipnodes cache by invoking the following command:

```
example# /usr/sbin/nscd -i ipnodes
```

Name ipadm – configure Internet Protocol network interfaces and TCP/IP tunables

Synopsis

```
ipadm
```

```
ipadm create-ip [-t] IP-interface
ipadm delete-ip IP-interface
```

```
ipadm create-vni [-t] VNI-interface
ipadm delete-vni VNI-interface
```

```
ipadm create-ipmp [-t] [-i interface,[...]...] IPMP-interface
ipadm delete-ipmp [-f] IPMP-interface
ipadm add-ipmp [-t] -i interface,[...] [-i interface,[...]...]
     IPMP-interface
ipadm remove-ipmp [-t] -i interface,[...] [-i interface,[...]...]
     IPMP-interface
```

```
ipadm show-if [[-p] -o field[,...]] [interface]
ipadm disable-if -t interface
ipadm enable-if -t interface
```

```
ipadm set-ifprop [-t] -m protocol -p prop=value[,...] interface
ipadm reset-ifprop [-t] -m protocol -p prop interface
ipadm show-ifprop [[-c] -o field[,...]] [-p prop,...]
     [-m protocol] [interface]
```

```
ipadm create-addr [-t] [-T static] [-d]
     -a {local|remote}=addr[/prefixlen],... addrobj | interface
ipadm create-addr [-t] -T dhcp [-w seconds | forever]
     [-h hostname] addrobj | interface
ipadm create-addr [-t] -T addrconf [-i {local|remote}=interface-id]
     [-p {stateful|stateless}={yes|no},..] addrobj | interface
```

```
ipadm delete-addr [-r] addrobj
ipadm show-addr [[-p] -o field[,...]] [-d]
     [addrobj | interface/ | interface]
ipadm up-addr [-t] addrobj
ipadm down-addr [-t] addrobj
ipadm refresh-addr [-i] addrobj
ipadm disable-addr -t addrobj
ipadm enable-addr -t addrobj
```

```
ipadm set-addrprop [-t] -p prop=value[,...] addrobj
ipadm reset-addrprop [-t] -p prop=value[,...] addrobj
ipadm show-addrprop [[-c] -o field[,...]] [-p prop[,...]]
     [addrobj | interface]
```

```
ipadm set-prop [-t] -p prop=value[,...] protocol
ipadm reset-prop [-t] -p prop protocol
ipadm show-prop [[-c] -o field[,...]] [-p prop[,...] protocol | protocol]
```

```
ipadm help [subcommand-name]
```

Description The `ipadm` command provides a set of subcommands that can be used to:

manage interfaces:

- create and delete interfaces of interface classes `ip`, `ipmp`, and `vni`
- modify interface properties
- display interface configuration

manage addresses:

- create and delete addresses
- modify address properties
- display address configuration

manage TCP/IP protocol properties:

- modify TCP/IP properties
- display TCP/IP properties

Note that `ipadm` is used to manage network configuration manually. The `netadm(1M)` DefaultFixed NCP should be enabled for these manual configurations. Many of the `ipadm` subcommands are not functional if the DefaultFixed NCP is not enabled. The `ipadm` subcommands that do not function unless the DefaultFixed NCP is enabled are:

`create-ip`, `delete-ip`, `create-vni`. `delete-vni`, `create-ipmp`, `delete-ipmp`, `add-ipmp`, `remove-ipmp`, `disable-if`, `enable-if` `set-ifprop`, `reset-ifprop`, `create-addr`, `down-addr`, `up-addr`, `disable-addr`, `enable-addr`, `refresh-addr`, `delete-addr`, `set-addrprop`, `reset-addrprop`

The following subcommands still continue to function when the DefaultFixed NCP is not enabled:

`show-if`, `show-ifprop`, `refresh-addr`, `show-addr`, `show-addrprop`, `set-prop`, `reset-prop`, `show-prop`

Refer to `netadm(1M)` for more information on how to list and enable NCPs.

The various operands to `ipadm` subcommands are described in the "Operands" section, which follows "Subcommands".

The `ipadm` command with no subcommands displays a concise summary of interface and address configuration on the system. The output contains all the interfaces (`ip`, `loopback`, `vni`, and `ipmp`) configured on the system along with the addresses configured on these interfaces. See `EXAMPLES`, below, for more information.

Required Authorization and Privilege The following subcommands require `solaris.network.interface.config` authorization and `PRIV_SYS_IP_CONFIG` privilege.

```
create-ip          create-addr
delete-ip          up-addr
create-vni         down-addr
```

```
delete-vni          refresh-addr
create-ipmp         disable-addr
delete-ipmp         enable-addr
add-ipmp            set-addrprop
remove-ipmp         reset-addrprop
disable-if          set-prop
enable-if           reset-prop
set-ifprop
reset-ifprop
```

In addition to the authorization and privilege specified above, the `ipadm` subcommands `create-ip`, `create-vni`, `create-ipmp`, and `enable-if` need `PRIV_NET_RAWACCESS` privilege.

Sub-commands The following subcommands are supported:

`create-ip` [`-t`] *IP-interface*
Create an IP interface that handles both IPv4 and IPv6 packets. The address of the IPv4 interface will be set to `0.0.0.0` and the address of the IPv6 interface will be set to `::`. This subcommand, by default, causes the information to persist, so that on the next reboot this interface will be instantiated.

An interface is implicitly enabled for IPv4 and IPv6 when it is created. See the `disable-if` and `enable-if` subcommands below, to disable or enable an interface.

Note that `lo0` is a special interface, called the loopback interface. It is a virtual IP interface and is not associated with any physical hardware. It is one of the first IP interfaces to be created on the system, with IPv4 address of `127.0.0.1` and IPv6 address of `::/128`.

`-t`, `--temporary`
Specifies that the operation is temporary and must not persist. The operation affects only the active configuration.

`delete-ip` *IP-interface*
Deletes the IP interface from active configuration. All addresses configured on the interface will be torn down. Further, all the persistent information related to the interface will be removed from the persistent data store and, for this reason, *interface* will not be instantiated upon reboot. To disable an interface from active configuration (rather than delete the interface), use the `disable-if` subcommand.

`create-vni` [`-t`] *VNI-interface*
Create a VNI interface that handles both IPv4 and IPv6 packets. The address of the IPv4 interface will be set to `0.0.0.0` and the address of the IPv6 interface will be set to `::`. This subcommand, by default, causes the information to persist, so that on the next reboot this interface will be instantiated.

The interface is implicitly enabled for IPv4 and IPv6 when it is created. See the `disable-if` and `enable-if` subcommands below, to disable or enable an interface.

Note that `vni` is a special interface, in that it is a virtual interface and does not have any hardware associated with it. See `vni(7d)`.

`-t`, `--temporary`
Specifies that the operation is temporary and must not persist. The operation affects only the active configuration.

`delete-vni` *VNI-interface*
Deletes the VNI interface from active configuration. All addresses configured on the interface will be torn down. Further, all the persistent information related to the IP interface will be removed from the persistent data store and, for this reason, interface will not be instantiated upon reboot. To disable the interface from active configuration (rather than delete the interface), use the `disable-if` subcommand.

`create-ipmp` [`-t`] [`-i` *interface*,[...]...] *IPMP-interface*
Create a IPMP interface that handles both IPv4 and IPv6 packets. The address of the IPv4 interface will be set to `0.0.0.0` and the address of the IPv6 interface will be set to `::`. This subcommand, by default, causes the information to persist, so that on the next reboot this interface will be instantiated.

The interface is implicitly enabled for IPv4 and IPv6 when it is created. See the `disable-if` and `enable-if` subcommands below, to disable or enable an IPMP interface.

`-t`, `--temporary`
Specifies that the operation is temporary and must not persist. The operation affects only the active configuration.

`-i`, `--interface` *interface*,[...]
A comma-separated list of interfaces to be added as underlying interfaces to the IPMP interface. The specified interfaces must exist in the active configuration to be successfully added to the IPMP group and must not be present in any other IPMP group. More than one `-i` option is allowed. The command returns with partial success if the IPMP interface was created but none of the given underlying interfaces were added successfully.

`delete-ipmp` [`-f`] *IPMP-interface*
Deletes the IPMP interface from active configuration. All addresses configured on the interface will be torn down. The command fails if the IPMP interface has any underlying interfaces, unless the `-f` option is specified. Further, all the persistent information related to the IPMP interface will be removed from the persistent data store and, for this reason, interface will not be instantiated upon reboot. To disable the interface from active configuration only (rather than delete the interface), use the `disable-if` subcommand.

`-f`, `--force`
If the IPMP interface has any underlying interfaces, specifying this option removes all the underlying interfaces from the group first, before deleting the IPMP interface.

`add-ipmp` [`-t`] `-i` *interface*,[...] [`-i` *interface*,[...]...] *IPMP-interface*
Adds one or more underlying IP interfaces to the given IPMP interface.

`-t`, `--temporary`
Specifies that the operation is temporary and must not persist. The operation affects only the active configuration.

`-i`, `--interface` *interface*,[...]
A comma-separated list of interfaces to be added as underlying interfaces to the IPMP interface. The specified interfaces must exist in the active configuration to be successfully added to the IPMP group and must not be present in any other IPMP group. The command returns with partial success if at least one interface was added and adding the remaining interfaces failed. More than one `-i` option is allowed.

`remove-ipmp` [`-t`] `-i` *interface*,[...] [`-i` *interface*,[...]...] *IPMP-interface*
Removes one or more underlying IP interfaces from the IPMP interface.

`-t`, `--temporary`
Specifies that the operation is temporary and must not persist. The operation affects only the active configuration.

`-i`, `--interface` *interface*,[...]
A comma-separated list of underlying interfaces to be removed from the IPMP interface. The specified interfaces must already be underlying interfaces for the given IPMP group. More than one `-i` option is allowed. The command returns with partial success if at least one interface was removed and removing the remaining interfaces failed.

`show-if` [[`-p`] `-o` *field*[,...]] [*interface*]
Show network interface configuration information, either for all the network interfaces configured on the system, including the ones that are only in the persistent configuration, or for the specified network interface.

`-o` *field*[,...], `--output` *field*[,...]
A case-insensitive, comma-separated list of output fields to display. The *field* name must be one of the fields listed below, or the special value `all` to display all fields. For each network interface, the following fields can be displayed:

`IFNAME`
The name of the IP interface.

`CLASS`
Indicates one of the following:

`ip`
An interface that is plumbed over an underlying datalink.

`ipmp`
An IPMP interface that is created over one or more underlying IP interfaces.

`loopback`
A loopback interface.

`vni`
A virtual IP interface. See `vni(7d)`.

`STATE`
Indicates one of the following for the displayed interface.

`ok`
Indicates that the required resources for an interface are allocated. For some interfaces this also indicates that the link is up.

`offline`
The interface is offline and thus cannot send or receive IP data traffic. See `if_mpadm(1M)`.

`failed`
Indicates that the datalink is down. If the interface is part of an IPMP group it could also mean that the interface has failed (that is, IFF_FAILED is set). Failed interfaces will not be used to send or receive IP data traffic. If this is set on a physical IP interface in an IPMP group, IP data traffic will continue to flow over other usable IP interfaces in the IPMP group. If this is set on an IPMP IP interface, the entire group has failed and no data traffic can be sent or received over any interfaces in that group. See `in.ndpd(1M)`.

`down`
Indicates that the interface is administratively down, preventing any IP packets from being sent or received through it.

`disabled`
Indicates that the interface has been disabled from the active configuration using the `disable-if` subcommand.

`ACTIVE`
Either `yes` or `no`, depending on whether the IP interface is being used by the system for IP data traffic.

`CURRENT`
For interface objects, in active configuration, it indicates any of the following flags.

`b`
interface supports broadcast

`m`
interface supports multicast

`p`
interface is a point-to-point link

`v`
interface is a virtual interface (for example, `vni(7d)`, loopback), that is, the physical interface has no underlying hardware.

`s`
IPMP interface is marked standby administratively. See `in.ndpd(1M)`.

`l`
interface is an underlying interface for an IPMP interface. See `in.ndpd(1M)`.

`i`
Underlying interface is inactive. See `in.ndpd(1M)`.

`V`
interface is a VRRP interface

`a`
VRRP interface is in accept mode (~`IFF_NOACCEPT`)

`Z`
Layer-3 protection of IP addresses for the interface has been administratively enforced.

`4`
interface can handle IPv4 packets

`6`
interface can handle IPv6 packets

Note that `b` and `p` are mutually exclusive.

`PERSISTENT`
Specifies the configuration that will be applied when the interface object is instantiated on reboot or re-enabled using the `enable-if` subcommand. It can be any or all of `s`, `l`, `4`, and `6` (see above). This field is not shown by default and will be shown only when `all` or `persistent` is specified with `-o`.

`OVER`
The underlying interface(s) over which the IPMP interface is created. This does not apply to other interface classes.

`-p`, `--parsable`
Display using a stable machine-parsable format. The `-o` option is required with this option. See "Parsable Output Format", below.

`disable-if -t` *interface*
Disables the specified *interface* by removing it from the active configuration. All the addresses configured on the interface will be disabled. If the interface object was created persistently to begin with, then the persistent configuration is unchanged. To re-enable this interface, one should use `enable-if`.

`-t`, `--temporary`
Specifies that the disable is temporary and changes apply only to the active configuration.

`enable-if -t` *interface*
Enables the given interface by reading the configuration from the persistent store. All the persistent interface properties, if any, are applied and all the persistent addresses, if any, on the given interface will be enabled.

`-t`, `--temporary`
Specifies that the enable is temporary and changes apply only to the active configuration.

`set-ifprop` [`-t`] `-m` *protocol* `-p` prop=*value*[,...] *interface*
Modifies an interface property to the value specified by the user. If the property takes multiple values then the values should be specified with a comma as the delimiter. Only one property can be specified at a time. The properties supported on an interface and the property's possible values can be retrieved using `show-ifprop` subcommand. Only one property at a time can be modified.

`-t`, `--temporary`
Specifies that the changes are temporary and changes apply only to the active configuration.

`-m` *protocol*, `--module` *protocol*
Identifies whether property should be applied for IPv4 or IPv6 packets.

`-p` prop=*value*[,...], `-prop` prop=*value*[,...]
A property to set to the specified values.

`reset-ifprop` [`-t`] `-m` *protocol* `-p` *prop interface*
Resets a property of the specified interface to its default value. If `-t` is not used, any persisted value of the property will be deleted. Only one property can be modified at a time.

`-t`, `--temporary`
Specifies that the resets are temporary and changes apply only to the active configuration.

`-m` *protocol*, `--module` *protocol*
Identifies whether the property being reset affects either IPv4 or IPv6 packets.

`-p` *prop*, `-prop` *prop*
A property to set to the specified values.

`show-ifprop` [[`-c`] `-o` *field*[,...]] [`-p` *prop*,...] [`-m` *protocol*] [*interface*]
Show the current and persistent values of one or more properties, either for all the created interfaces or for the specified interface. Several properties of interest can be retrieved at one time by providing comma-separated property names to `-p` option. If the `-p` option is not specified, all available interface properties are displayed.

`-o` *field*[,...], `--output` *field*[,...]
A case-insensitive, comma-separated list of output fields to display. The *field* name must be one of the fields listed below, or the special value `all` to display all fields. For each interface, the following fields can be displayed:

`IFNAME`
The name of the interface.

`PROPERTY`
The name of the property.

`PROTO`
The name of the protocol the property belongs to. The protocols currently supported are IPv4 and IPv6.

`PERM`
The read/write permissions of the property. The value shown will be `r` (read-only), `w` (write-only) or `rw` (read-and-write).

`CURRENT`
The current value of the property. For disabled interfaces, because a value is not set, it will be shown as `--`.

`PERSISTENT`
The persistent value of the property. Persistent values are the values that will be reapplied on reboot.

`DEFAULT`
The default value of the property. If the property has no default value, `--` is displayed.

`POSSIBLE`
A comma-separated list of the values the property can have. If the values span a numeric range, *min* - *max* might be displayed as shorthand. If the possible values are unknown, `?` is displayed or if they are unbounded, `--` is displayed.

`-c`, `--parsable`
Display using a stable machine-parsable format. The `-o` option is required with this option. See "Parsable Output Format", below.

`-p` *prop*,..., `--prop=`*prop*
A comma-separated list of properties to display. See the sections on interface properties following subcommand descriptions.

`-m` *protocol*, `--module` *protocol*
Displays properties matching the given protocol. Valid values are `ipv4` and `ipv6`.

For the supported list of interface properties, see "Interface Properties" below.

`create-addr` [`-t`] [`-T` static] [`-d`] `-a` {local | remote}=*addr*[/*prefixlen*],... *addrobj* | *interface*
Creates a static IPv4 or IPv6 address on an interface. The interface is either specified specifically as an argument or is derived from the *addrobj* argument. The interface on which the address is being created must already exist. The created static address will subsequently be identified by *addrobj*. When the command is invoked with an interface argument, then the command will automatically generate an *addrobj* for the address and will print the generated name to stdout.

Note – Automatically generated *addrobj* names have the following forms:

```
interface/v4            interface/v6
interface/v4a           interface/v6a
interface/v4b           interface/v6b
      .                       .
      .                       .
      .                       .
interface/v4z           interface/v6z
interface/v4aa          interface/v6aa
interface/v4ab          interface/v4ab
      .                       .
      .                       .
      .                       .
```

The IP address version is used in the automatic generation of names and names are made unique by increasingly appending one or more of the characters [a-z] to the v[46] prefix.

By default, a configured address will be marked up, so that it can be used as a source or destination of or for outbound and inbound packets.

All address objects are enabled when they are created. See the `disable-addr` and `enable-addr` subcommands for instructions on disabling or enabling an address object.

A persistent operation cannot be performed on a temporary object. That is, if the interface is temporarily created, then one cannot create the address object persistently.

If the interface specified in the *addrobj* name is an IPMP interface, a static data address is created on the IPMP interface. If the interface specifed in the *addrobj* name is an underlying interface for an IPMP group, a static test address is created on the underlying interface.

`-t`, `--temporary`
: Specifies that the configured address is temporary and changes apply only to the active configuration.

`-d`, `--down`
: Specifies that the configured address should be marked `down`, that is, the address will not be used as a source or destination of IP packets.

`-a` {local | remote}=*addr*[/*prefixlen*],...
`--address` {local | remote}=*addr*[/*prefixlen*],...
: *addr* indicates a literal IP address or a hostname corresponding to the local or remote end-point (for point-to-point interfaces).

  If a hostname is specified its numeric value is uniquely obtained using the entry in `/etc/hosts`. If no numeric IP address is defined in the file, then the numeric value is uniquely obtained using the resolver order specifed for `hosts` or `ipnodes` in `nsswitch.conf(4)`. If there are multiple entries for a given hostname, an error will be

generated. Because IP addresses are created before naming services have been brought online during the boot process, it is important that any hostname used be included in `/etc/hosts`.

If the *prefixlen* is not explicitly specified in the command-line, the netmask for the address is obtained by following the search in the order listed below:

1. using the order specified for `netmasks` in `nsswitch.conf(4)`
2. interpreting IPv4 address using Classful subnetting semantics defined in RFC 791, and interpreting IPv6 addresses using the definitions in RFC 4291.

For point-to-point interfaces, along with the address of the local end-point the address of the remote end-point must be specified (for example, `-a` local=*laddr*,remote=*raddr*). If *prefixlen* for the remote end-point is specified, an error will be returned.

Note that if the interface requires only a local address, specify it directly with the `-a` option as follows: `-a` *addr*[/*prefixlen*]. The address will automatically be considered a local address.

`create-addr` [`-t`] `-T` dhcp [`-w` *seconds* | forever] [`-h` *hostname*] *addrobj* | *interface*

Creates a DHCP-controlled IPv4 address on an interface. The interface is either specified specifically as an argument or is derived from the *addrobj* argument. The created IPv4 address will subsequently be identified by *addrobj*. When the *addrobj* contains an underlying interface, this command creates a test address; when it contains an IPMP interface, it creates a data address.

When the command is invoked with an interface argument, then the command will automatically generate an *addrobj* name for the address and will print the generated name to stdout.

All the address objects are enabled when they are created. See the `disable-addr` and `enable-addr` subcommands for instructions on disabling and enabling an address object.

A persistent operation cannot be performed on a temporary object. That is, if the interface is temporarily created, one cannot create the address object persistently.

If the interface specified in the *addrobj* name is an IPMP interface, the address obtained through DHCP is created as a data address on the IPMP interface.

`-h` *hostname*

Specifies the hostname to which the client would like the DHCP server to map the client's leased IPv4 address. There is no guarantee that the DHCP server will be able to fulfill the hostname request.

`-t`, `--temporary`

Specifies that the configured address is temporary and changes apply only to the active configuration.

`-w` *seconds* | forever, `--wait` *seconds* | forever
Specifies the amount of time, in seconds, to wait until the operation completes. If no wait interval is given, and the operation is one that cannot complete immediately, `ipadm` will, by default, wait 120 seconds for the requested operation to complete. Note that the default wait time is subject to change in future releases. The symbolic value `forever` can be used as well, with obvious meaning.

`create-addr` [`-t`] `-T` addrconf [`-i` {local | remote}=*interface-id*] [`-p` {stateful | stateless}={yes | no},..] *addrobj* | *interface*
Creates an auto-configured IPv6 address on an interface. The interface is either specified specifically as an argument or is derived from the *addrobj* argument. The created IPv6 addresses will be identified by *addrobj*. When the command is invoked with an interface argument, then the command will automatically generate an *addrobj* name for the address and will print the generated name to stdout.

The system uses the default interface ID (for the media-type Ethernet, the Interface ID is the MAC address of the interface) to generate auto-configured addresses. This behavior can be overridden using `-i` option.

By default:

- IPv6 addresses will be auto-configured based on prefixes advertised by routers as described in RFC 4862 and...
- IPv6 addresses will be auto-configured on the specified interface using the IPv6 address offered by DHCPv6 server as described in RFC 3315. (That is, `-p stateful=yes,stateless=yes` is the default option.)

All the address objects are enabled when they are created. See the `disable-addr` and `enable-addr` subcommands for instructions on disabling and enabling an address object.

A persistent operation cannot be performed on a temporary object. That is, if the interface is temporarily created, then one cannot create the address object persistently.

If the interface specified in the *addrobj* name is an IPMP interface, the addresses obtained through IPv6 autoconfiguration are created as data addresses on the IPMP interface.

`-t`, `--temporary`
Specifies that the configured address is temporary and changes apply only to the active configuration.

`-i` {local | remote}=*interface-id*, `--interface-id` {local | remote}=*interface-id*
Specifies the interface ID to be used for generating auto-configured addresses.

For point-to-point interfaces, the interface id of the remote end-point can be specified (for example, `-i local=`*lid*`,remote=`*rid*).

Note that if the interface requires only a local interface id, specify it directly with the `-i` option as follows: `-i` *lid*. The interface id will automatically be considered a local interface id.

`-p` {stateful | stateless}={yes | no},..
`--prop` {stateful | stateless}={yes | no},..
Specifies if stateful or stateless or both methods of auto-configuration should be enabled or not.

If `-p stateful=no` is specified, then stateful auto-configuration based on DHCPv6–specified IPv6 addresses will not be performed.

If `-p stateless=no` is specified, then stateless auto-configuration based on the router-advertised prefixes will not be performed.

If `-p stateful=no,stateless=no` is specified, then both the methods of auto-configuration will not be performed.

With the `-T addrconf` option, `-p stateful=yes,stateless=yes` is used by default.

`delete-addr` [`-r`] *addrobj*
Deletes all the addresses identified by *addrobj* on the interface specified in the *addrobj*. It also removes these addresses from the persistent data-store; thus, these addresses will not be instantiated on reboot.

If the address object is a DHCP-controlled address, `delete-addr` removes the address from the system without notifying the DHCP server, and records the current lease for later use.

`-r`, `--release`
If the *addrobj* is a DHCP-controlled address, this option brings about the relinquishing of the DHCP-controlled IP addresses on the interface by notifying the server and the discarding of the current lease.

`show-addr` [[`-p`] `-o` *field*[,...]] [`-d`] [*addrobj* | *interface*/]
Show address information, either for the given *addrobj* or all the address objects configured on the specified interface, including the address objects that are only in the persistent configuration.

`-p`, `--parsable`
Display using a stable machine-parsable format. The `-o` option is required with this option. See "Parsable Output Format", below.

`-o` *field*[,...], `--output` *field*[,...]
A case-insensitive, comma-separated list of output fields to display. The *field* name must be one of the fields listed below, or the special value `all` to display all fields. For each interface, the following fields can be displayed:

`ADDROBJ`
The name of the address object.

`TYPE`
Type of the address object. It will be one of: `from-gz`, `static`, `dhcp`, or `addrconf`. The `static`, `dhcp`, and `addrconf` types correspond to the type of the address object

specified by the `-T` option of `create-addr`. The `from-gz` type will only be displayed in non-global zones, and indicates that the address was configured based on the `allowed-address` property configured for the non-global exclusive-IP zone from the global zone.

`STATE`
State of the address object. This field is shown only when `all` is specified with `-o`. This indicates one of the following values:

`disabled`
Address is not part of the active configuration (see `disable-addr` and `disable-if`).

`down`
Address is administratively down (see `down-addr`).

`duplicate`
Address was found to conflict with another system's IP address by duplicate address detection (DAD) and cannot be used until the conflict is resolved. The system will periodically rerun DAD to determine if the conflict has been resolved. Alternatively, `refresh-addr` can be used to immediately rerun DAD.

`inaccessible`
Address cannot be used because the IP interface it is configured on has failed.

`ok`
Address is enabled, up, and functioning properly. The system will accept IP packets destined to this address, and will originate IP packets with this address in accordance with the configured IP source address selection policy.

`tentative`
Address is currently undergoing duplicate address detection (for example, as part of `up-addr` or `refresh-addr`).

`CURRENT`
For address objects in active configuration, it indicates any of the following flags. This field is not shown by default and will be shown only when `all` or `current` is specified with `-o`.

`d` (deprecated)
Will not be used as source address for outbound packets unless either there are no other addresses available on the interface or the application has explicitly bound to this address.

`p` (private)
Address not advertised by the routing daemon.

`t` (temporary)
Temporary IPv6 address as defined in RFC 3041.

`U` (up)
Address is marked up for use as a source/destination of outbound/inbound packets.

u (unnumbered)
Address matches the local address of some other link in the system.

PERSISTENT
Specifies the configuration that will be applied when the address object is instantiated on reboot or re-enabled using the `enable-addr` subcommand. It can be any or all of `U`, `p`, and `d` (see above).

ADDR
Numeric IPv4 or IPv6 address. In the case of point-to- point interfaces, the addresses of both the endpoints, are displayed (*laddr-->raddr*). For an address object of type `dhcp`, if the state of the address object is `disabled`, or if the address is `0.0.0.0` for IPv4 or `::` for IPv6, then a question mark (`?`) is displayed.

CID-TYPE
The type of the Client ID used by the dhcpagent(1M), if the address is being obtained using DHCP. For IPv4, this shows the type of the DUID used in constructing the RFC 4361 Client ID. The type is one of `DUID-LLT`, `DUID-EN`, `DUID-LL`, `other`, or `default`. This field is not shown in the default output. It can be shown using `-d` or using `cid-type` or `all` with `-o`.

DUID-LLT
Type 1 RFC 3315 DUID is used in constructing `CID-VALUE` (for example, `1,1,63463777,0a:0b:0c:0d:0e:0f`). Refer to the RFC for more details.

DUID-EN
Type 2 RFC 3315 DUID is used in constructing `CID-VALUE` (for example, `1,1,63463777,0a:0b:0c:0d:0e:0f`). Refer to the RFC for more details.

DUID-LL
Type 3 RFC 3315 DUID is used in constructing `CID-VALUE` (for example, `1,1,63463777,0a:0b:0c:0d:0e:0f`). Refer to the RFC for more details.

other
An RFC 3315 DUID of a Type in {0,4-65535} is used to derive the Client ID (for example, `4,0x734633`) or the `CID-VALUE` is a raw Client ID (for example, `Sun, 0xab3146`) that does not conform to RFC 3315.

default
Indicates that the RFC 3315 DUID is not being used to construct the Client ID. Instead, Client ID is derived using the MAC address of the interface as per RFC 2132. `CID-VALUE` will contain the string `0x01` followed by the MAC address hex string. This is applicable only for IPv4.

CID-VALUE
Value of the Client ID used by the dhcpagent(1M), if the address is being obtained using DHCP. Format used follows that of the configuration parameter `CLIENT_ID` in file `/etc/default/dhcpagent`. Refer to the description of `CLIENT_ID` in dhcpagent(1M). When the `CID-TYPE` is default, the `CID-VALUE` contains the legacy

`CLIENT-ID`, constructed as per RFC 2132. This field is not shown in the default output. It can be shown using `-d` or using `cid-type` or all with `-o`.

`BEGIN`
The time at which the lease began, if one is available, for the addresses obtained using DHCP. The time is displayed in the format dictated by the `LC_TIME` locale environment variable. For addresses not configured using DHCP or for DHCP addresses that do not have a lease yet, `--` (two hyphens) will be displayed. This field is not shown in the default output. It can be shown using `-d` or using `cid-type` or all with `-o`.

`EXPIRE`
The time at which the lease expires, if one is available, for the addresses obtained using DHCP. The time is displayed in the format dictated by the `LC_TIME` locale environment variable. For addresses not configured using DHCP or for DHCP addresses that do not have a lease yet, `--` (two hyphens) will be displayed. This field is not shown in the default output. It can be shown using `-d` or using `cid-type` or all with `-o`.

`RENEW`
The time at which the lease was last renewed for the addresses obtained using DHCP. The time is displayed in the format dictated by the `LC_TIME` locale environment variable. For addresses not configured using DHCP or for DHCP addresses that do not have a lease yet, `--` (two hyphens) will be displayed. This field is not shown in the default output. It can be shown using `-d` or using `cid-type` or all with `-o`.

`-d`, `--dhcp`
Display the `dhcp` status fields for addresses acquired using DHCP. The fields displayed are `ADDROBJ`, `STATE`, `ADDR`, `CID-TYPE`, `CID-VALUE`, `BEGIN`, `EXPIRE`, and `RENEW`. This option displays only the human-readable output and cannot be used in conjunction with `-p`.

Note – In some cases you will see addresses that have a question mark (`?`) in the address object name. This means that those addresses were created outside the `ipadm` library and therefore not known to `ipadm`.

`down-addr` [`-t`] *addrobj*
The address identified by *addrobj* is marked down, so that it cannot be used as a source/destination of outbound/inbound packets. This command has no effect if the address object was already marked down prior to the `down-addr` invocation. If the address object is of type `addrconf`, the command returns an error.

`-t`, `--temporary`
Specifies that the configured address is temporary and changes apply only to the active configuration. This option is mandatory if the address object type is `dhcp`.

`up-addr` [`-t`] *addrobj*
The address identified by *addrobj* is marked up, so that it can be used as a source/destination of outbound/inbound packets. This subcommand has no effect if the

address object has been marked down by the system because it is a duplicate address, or if the address was marked up prior to the `up-addr` invocation. If the address object is of type `addrconf`, the command returns an error.

`-t`, `--temporary`
Specifies that the configured address is temporary and changes apply only to the active configuration. This option is mandatory if the address object type is `dhcp`.

`refresh-addr` [`-i`] *addrobj*
If the *addrobj* is of the type `static` then DAD (Duplicate Address Detection) will be restarted (if necessary) on the address identified by the address object.

If the *addrobj* is of the type `dhcp`, then the lease duration obtained on the address will be extended by the DHCP client daemon.

If the *addrobj* is of the type `addrconf` then the command returns an error.

`-i`, `--inform`
For a specified IP address, obtains network configuration parameters from DHCP without obtaining a lease on it. This is useful in situations where an IP address is obtained through mechanisms other than DHCP.

`disable-addr -t` *addrobj*
Disables the address by removing it from the active configuration. If the address object was originally created persistently, then the persistent configuration is unchanged. To re-enable this *addrobj*, one should use `enable-addr`.

`-t`, `--temporary`
Specifies that the disabling is temporary and changes apply only to the active configuration.

`enable-addr -t` *addrobj*
Enables the given *addrobj* by reading the configuration from the persistent store. All the persistent address properties are applied to the address object. This subcommand requires that the interface on which the address object is being enabled be present. If the interface itself is missing in active configuration and is present in persistent store, that is, if the interface is disabled, then the user has to run `enable-if` before invoking `enable-addr`.

`-t`, `--temporary`
Specifies that the enabling is temporary and changes apply only to the active configuration.

`set-addrprop` [`-t`] `-p` prop=*value*[,...] *addrobj*
Sets the value of a property on the *addrobj* specified. If the *addrobj* maps to several addresses, then property changes applies to all the addresses referenced by the *addrobj*. Only one property can be specified at a time. The properties supported on the *addrobj* and the property's possible values can be retrieved using `show-addrprop` subcommand. If the *addrobj* is of type `addrconf`, the command returns an error.

`-t`, `--temporary`
Specifies that the changes are temporary and changes apply only to the active configuration.

`-p` prop=*value*[,...], `--prop` prop=*value*[,...]
A property to set to the specified values.

`reset-addrprop` [`-t`] `-p` *prop addrobj*
Resets the given address property to its default value. If `-t` is not used, any persistent value of the property will be deleted. Only one property can be modified at a time. If the *addrobj* is of type `addrconf`, the command returns an error.

`-t`, `--temporary`
Specifies that the resets are temporary and changes apply only to the active configuration.

`-p` *prop*, `--prop` *prop*
A property to be reset.

`show-addrprop` [[`-c`] `-o` *field*[,...]] [`-p` *prop*,...] [*addrobj*]
Show the current and persistent values of one or more properties, either for all the configured address objects or for the specified *addrobj*. Several properties of interest can be retrieved at one time by providing comma-separated property names to `-p` option. If the `-p` option is not specified, all available properties are displayed. If the *addrobj* is of type `addrconf`, the command returns an error.

`-o` *field*[,...], `--output` *field*[,...]
A case-insensitive, comma-separated list of output fields to display. The *field* name must be one of the fields listed below, or the special value `all` to display all fields. For each *addrobj*, the following fields can be displayed:

`ADDROBJ`
The name of the address object.

`PROPERTY`
The name of the property.

`PERM`
The read/write permissions of the property. The value shown will be `r` (read only), `w` (write only) or `rw` (read/write).

`CURRENT`
The current value of the property. For the disabled addresses, because the value is not set, the value displays as a double hyphen (`--`).

`PERSISTENT`
The persistent value of a property. Persistent values are the values that will be reapplied on reboot.

`DEFAULT`
The default value of the property. If the property has no default value, double hyphen (`--`) is shown.

`POSSIBLE`
A comma-separated list of the values a property can have. If the values span a numeric range, *min* - *max* might be shown as shorthand. If the possible values are unknown, a question mark (`?`) is displayed or if they are unbounded, double hyphen (`--`) will be shown.

`-c`, `--parsable`
Display using a stable machine-parsable format. The `-o` option is required with this option. See "Parsable Output Format", below.

`-p` *prop*,..., `--prop=`*prop*
A comma-separated list of properties to display. See the sections on address object properties following subcommand descriptions.

`set-prop` [`-t`] `-p` prop[+ | –]=*value*[,...] *protocol*
Modifies the value of a protocol property to the *value* specified. If the property takes multiple values, the values should be specified with a comma as the delimiter. Only one property can be specified at a time. By default, the value is persistent and will be reapplied on reboot. The properties supported on a protocol and the property's possible values can be retrieved using the `show-prop` subcommand

The following protocols are supported: `ip`, `ipv4`, `ipv6`, `icmp`, `tcp`, `udp` and `sctp`.

Note that for some properties, it might be possible to set the value of the property both globally, and on a per-interface basis. The per-interface value can be set using the `set-ifprop` subcommand. In such cases, if the administrator chooses to customize the per-interface value of the property to be distinct from the global value, the per-interface value overrides the global setting for that interface.

`-t`, `--temporary`
Specifies that the changes to properties are temporary and changes apply only to the active configuration.

`-p` prop[+|-]=*value*[,...], `--prop` prop[+|-]=*value*[,...]
A property to set to the specified values. It also provides the following "qualifiers" to perform add and delete operations in addition to assignment.

`+`
Adds the given value to the current list of value(s).

`-`
Removes the given value from the current list of value(s).

`=`
Makes a new assignment and removes all the current value(s).

See EXAMPLES for more information on how to use the qualifiers.

`reset-prop` [`-t`] `-p` *prop protocol*
Resets a property of the specified protocol to the default value of the property. If `-t` is not used, any persistent value of the property will be deleted. Only one property can be modified at a time.

`-t`, `--temporary`
Specifies that the resets are temporary and changes apply only to the active configuration.

`-p` *prop*, `--–prop` *prop*
A property to be reset.

`show-prop` [[`-c`] `-o` *field*[,...]] [`-p` *prop*[,...] *protocol* | *protocol*]
Show the current and persistent values of one or more properties, either for all supported protocols or for the specified protocol. Several properties of interest can be retrieved at a time by providing comma-separated property names to `-p` option. If the `-p` option is not specified, all available properties are displayed.

`-o` *field*[,...], `--output` *field*[,...]
A case-insensitive, comma-separated list of output fields to display. The *field* name must be one of the fields listed below, or the special value `all` to display all fields. For each *protocol*, the following fields can be displayed:

`PROTO`
The name of the protocol.

`PROPERTY`
The name of the property.

`PERM`
The read/write permissions of the property. The value shown will be `r` (read only), `w` (write only) or `rw` (read/write).

`CURRENT`
The current value of the property. For the disabled addresses, because the value is not set, the value displays as a double hyphen (`--`). If the value is unknown, it is displayed as a question mark (`?`). If the current value of the property is not in the set of listed `POSSIBLE` values, the keyword `custom` is displayed.

`PERSISTENT`
The persistent value of a property. Persistent values are the values that will be reapplied on reboot.

`DEFAULT`
The default value of the property. If the property has no default value, double hyphen (`--`) is shown.

`POSSIBLE`
A comma-separated list of the values for the property setting to be used with the `set-prop` subcommand. If the values span a numeric range, *min* - *max* might be shown as shorthand. If the possible values are unknown, a question mark (`?`) is displayed or if they are unbounded, double hyphen (`--`) will be shown.

`-c`, `--parsable`
Display using a stable machine-parsable format. The `-o` option is required with this option. See "Parsable Output Format", below.

`-p` *prop*,..., `--prop=`*prop*
A comma-separated list of properties to display. See the sections on protocol properties following subcommand descriptions.

For the supported list of properties for every protocol, see "Protocol Properties" below.

`help` [*subcommand-name*]
Displays all of the supported `ipadm` subcommands or usage for a given subcommand. If you display help for a specific subcommand, the command syntax is displayed, along with an example. Using `ipadm help` without any argument displays all of the subcommands.

Parseable Output Format

The `ipadm` "show" subcommands have an `-o` option that displays output in a machine-parsable format. The output format is one or more lines of colon (:) delimited fields. The fields displayed are specific to the subcommand used and are listed under the entry for the `-o` option for a given subcommand. Output includes only those fields requested by means of the `-o` option, in the order requested. Note that the `-o all` option, which displays all the fields for a given subcommand, cannot be used with parsable output option.

When you request multiple fields, any literal colon characters are escaped by a backslash (`\`) before being output. Similarly, literal backslash characters are also escaped (`\\`). This escape format is parsable by using shell `read(1)` functions with the environment variable set as `IFS=:`. Note that escaping is not done when you request only a single field.

Protocol Properties

The following protocol properties are supported:

Note – There are protocol properties, specific to a protocol, that begin with "_" (underbar). These properties are subject to change or removal and by default, are not displayed in `ipadm show-prop` output. See *Oracle Solaris Tunable Parameters Reference Manual* for details.

`cong_default` (TCP, SCTP)
Specify the default congestion control algorithm used by the protocol when new connections are created. Applications can opt to choose a different algorithm at a later point in the connnection's lifetime. Only enabled algorithms can be set as default (see `cong_enabled`).

`cong_enabled` (TCP, SCTP)
This option can be used to enable or disable congestion control algorithms. By default, all algorithms installed on the systems are enabled. Disabled algorithms cannot be set as default (see `cong_default`) or used by applications.

Algorithms can be added or removed using the `set-prop` subcommand and the modifiers + and -.

`ecn` (TCP)
Explicit Congestion Control (see RFC 3168 for more information). Possible values are the same as above: `never`, `passive`, and `active`.

`extra_priv_ports` (TCP, SCTP, UDP)
This option define additional privileged ports outside of the 1-1023 range. Any program that attempts to bind the ports listed here must run as root. This prevents normal users from starting server processes on specific ports.

These ports can be added, removed, or assigned using the `set-prop` subcommand and the modifiers +, -, and =. See `EXAMPLES` below on usage.

`forwarding` (IPv4), `forwarding` (IPv6)
Enable/disable global IPv4 or IPv6 forwarding. All the configured interfaces will start/stop forwarding packets. Individual interfaces can override the global option using `set-ifprop`.

`hostmodel` (IPv4), `hostmodel` (IPv6)
Control send/receive behavior for IP packets on a multi-homed system. The value of `hostmodel` can be set to `strong` or `weak`, corresponding to the equivalent end-system model definitions of RFC 1122. In addition, a third value of `src-priority` is also supported. In the `src-priority` `hostmodel` scenario, a packet will be accepted on any interface, as long as the packet's destination IP address is configured and marked `UP` on one of the host's interfaces. When transmitting a packet, if multiple routes for the IP destination in the packet are available, the system will prefer routes where the IP source address in the packet is configured on the outgoing interface. If no such route is available, the system will fall back to selecting the “best” route, as with the weak ES case.

`max_buf` (TCP, SCTP, UDP, ICMP)
Maximum size of the send or receive socket buffer. The current value of this property limits the maximum value of `recv_buf` and `send_buf`.

`recv_buf` (TCP, SCTP, UDP, ICMP)
`send_buf` (TCP, SCTP, UDP, ICMP)
Modifies the receive or send buffer sizes for the specified protocol. The maximum value of these properties is bound by the current value of the `max_buf` property.

`sack` (TCP)
Selective acknowledgment (SACK) allows recipients to selectively acknowledge out-of-sequence data and is intended to increase performance for data transfers over lossy links. See RFC 2018 for information on the SACK. Possible values and meanings:

`never`
Will neither accept SACK nor send out SACK information.

`passive`
Will accept SACK but not send out.

`active`
Will both accept SACK and send out SACK information.

`smallest_anon_port` (TCP, SCTP, UDP)
`largest_anon_port` (TCP, SCTP, UDP)
These options define the upper and lower bounds on ephemeral ports. Ephemeral (means short-lived) ports are used when establishing outbound network connections. Note that the current value of the `smallest_anon_port` should be always less than or equal to the current value of `largest_anon_port`.

`smallest_nonpriv_port` (TCP, SCTP, UDP)
This option define the start of non-privileged ports. The non-privileged port range normally starts at 1024. Any program that attempts to bind a non-privileged port does not have to run as root.

`ttl` (IPv4), `hoplimit` (IPv6)
Specifies the value that will be set for `ttl/hoplimit` field of an IPv4 or IPv6 header. Can be used to prevent the system from reaching other systems more than *N* hops away where *N* was the value specified.

Interface Properties

The following interface properties are supported:

`arp`
Enables/disables the use of the Address Resolution Protocol (ARP) on an interface. ARP is used in mapping between network level addresses and link level addresses. This is currently implemented for mapping between IPv4 addresses and MAC addresses. Possible values are `on` or `off`. Default is `on`.

`exchange_routes`
Enables/disables exchanging of routing information on this interface. Possible values are `on` or `off`. Default is `off`.

`group`
Specifies the group name of the IPMP interface for which this interface is an underlying interface. If the interface is of class IPMP, this specifies the name of the IPMP group. It is a read-write property only on IPMP interfaces. For other interface classes, this property is read-only.

`forwarding`
Enables/disables IP forwarding on an interface. When enabled, the IP packets can be forwarded to and from the interface. Possible values are `on` or `off`. Default is `off`.

`metric`
Set the routing metric of the interface to *n*; if no value is specified, the default is `0`. The routing metric is used by the routing protocol. Higher metrics have the effect of making a route less favorable. Metrics are counted as additional hops to the destination network or host.

`mtu`
Set the maximum transmission unit of the interface to *n*. For many types of networks, the MTU has an upper limit, for example, 1500 for Ethernet.

`nud`
Enables/disables the neighbor unreachability detection mechanism on a point-to-point physical interface. Possible values are `on` or `off`. Default is `on`.

`standby`
Specifies whether the interface is configured as a standby interface for an IPMP group. This property is not applicable to IPMP interfaces.

`usesrc`
Specifies a physical or virtual interface to be used for source address selection. If the keyword `none` is used, then any previous selection is cleared. Default is `none`.

Address Properties

The address properties listed below are supported. Note that modifying address properties for `addrconf` address objects is not supported.

`deprecated`
The address should no longer used as a source address in new communications, but packets addressed to this address are processed as expected. Possible values are `on` or `off`. Default is `off`. This property is not supported on an address object of type `dhcp`.

`prefixlen`
Specifies the number of left-most contiguous bits of the address that comprise the IPv6 prefix or IPv4 netmask of the address. The remaining low-order bits define the host part of the address. When `prefixlen` is converted to a text representation of the address, the address contain 1's for the bit positions that are to be used for the network part, and 0's for the host part. The `prefixlen` must be specified as a single decimal number. This property is not supported on an address object of type `dhcp`.

`private`
Specifies that the addresses should not be advertised by the `in.routed` routing daemon. Possible values are `on` or `off`. Default is `off`.

`reqhost`
The hostname to which the client would like the DHCP server to map the client's leased IPv4. A hostname request is not guaranteed to be fulfilled.

`transmit`
Enables packets to be transmitted using the addresses referenced by the address object. This is the default behavior when the address is up. Possible values are `on` or `off`. Default is `on`.

`zone`
Specifies the zone in which all the addresses referenced by the address object should be placed. The named zone must be active in the kernel in the ready or running state. The interface is unplumbed when the zone is halted or rebooted. The zone must be configured

to be an shared-IP zone. zonecfg(1M) is used to assign network interface names to exclusive-IP zones. To modify the zone assignment such that it persists across reboots, please use zonecfg(1M). Possible values are the list of all the zones configured on the system. Default is `global`.

Operands Each `ipadm` subcommand operates on one of the following objects:

addrobj
An address configured on a network interface is identified by an *addrobj*. An *addrobj* consists of two parts. The first part is the name of the network interface on which the address is configured. The second part is a user-specified string that can use any of the alphanumeric characters and can be at-most 32 characters in length and must begin with a letter. The two parts of the *addrobj* are delimited by a slash (`/`). An address object always represents a unique set of addresses in a system.

Note – The exception to this namespace rule is that some addresses, created externally to `ipadm`, might be created with an *addrobj* where the second part of the *addrobj* includes a prefix. In such cases, the second part of the *addrobj* will contain a prefix, a "`.`" (period) delimiter, and a consumer-defined string (for example, `net0/zoneadmd.v4`). `ipadm` users cannot create an *addrobj* with this format because `ipadm` cannot create an *addrobj* that includes a period. It is possible, though not optimal, to use `ipadm` to further manage an *addrobj* of this type.

interface
Name of the network interface on which network address is configured. In general, the name can use any alphanumeric characters, plus the underscore (_) and the period (`.`), but must start with an alphabetic character and end with a number.

protocol
Name of the TCP/IP Internet protocol family for which a property is to be configured. Following protocols are supported: `ip`, `ipv4`, `ipv6`, `icmp`, `tcp`, `sctp` and `udp`.

Examples **EXAMPLE 1** Using `ipadm` with No Arguments

The following command displays a concise view of the interface and address configuration on a system.

```
# ipadm
NAME              CLASS/TYPE STATE     UNDER   ADDR
ipmp0             ipmp       degraded  --      --
  ipmp0/v6        static     ok        --      2001:db8:1:2::4c08/128
lo0               loopback   ok        --      --
  lo0/v4          static     ok        --      127.0.0.1/8
  lo0/v6          static     ok        --      ::1/128
net0              ip         ok        --      --
  net0/dhcp       dhcp       ok        --      10.132.146.234/23
  net0/v4         static     ok        --      10.132.146.233/23
net1              ip         failed    ipmp0   --
```

EXAMPLE 1 Using `ipadm` with No Arguments *(Continued)*

```
  net1/aconf     addrconf   ok        --      fe80::214:4fff:fe58:1831/10
net2             ip         ok        ipmp0   --
  net2/aconf     addrconf   ok        --      fe80::214:4fff:fe58:1832/10
```

EXAMPLE 2 Creating IPv4 Static Addresses

The following command creates the address 10.2.3.4/24 on interface `bge1` (linkname `net1`) and marks the address `up`, for use.

```
# ipadm create-ip net1
# ipadm create-addr -T static -a local=10.2.3.4/24 net1/v4static1
```

Alternatively automatic address object name generation can be used. The automatically generated name will be displayed to the console and can be used in any future `ipadm` commands requiring an address object name.

```
# ipadm create-ip net1
# ipadm create-addr -T static -a local=10.2.3.4/24 net1
net1/v4
```

The following command creates another address 10.2.3.5/24 on interface `net1` but marks the address `down` until explicitly marked up.

```
# ipadm create-addr -T static -d -a 10.2.3.5/24 net1
net1/v4
```

Note that 10.2.3.5/24 is assumed to be the local address, because `local` was not used and there was only one address.

The following command marks the address object `net1/v4a` `up` that was previously marked `down`.

```
# ipadm up-addr net1/v4a
```

If the `DUPLICATE` flag was set on the address object, then `refresh-addr` will verify that the address is still a duplicate on the network. If it is not, the address will be marked `up`.

```
# ipadm refresh-addr net1/v4a
```

The following command lists the addresses that were configured. This shows that the address `net1/v4a` is not a duplicate.

```
# ipadm show-addr
ADDROBJ           TYPE    STATE      ADDR
lo0/v4            static  ok         127.0.0.1/8
lo0/v6            static  ok         ::/128
net1/v4           static  ok         10.2.3.4/24
net1/v4a          static  ok         10.2.3.10/24
```

EXAMPLE 3 Creating DHCPv4-controlled Addresses

The following command obtains a DHCPv4 address on interface `bge1` (linkname `net1`).

```
# ipadm create-ip net1
# ipadm create-addr -T dhcp net1/dhaddr
# ipadm show-addr net1/dhaddr
ADDROBJ           TYPE    STATE      ADDR
net1/dhaddr       dhcp    ok         10.8.48.173/25
```

The following command extends the lease duration for the DHCPv4 address object `net1/dhaddr`.

```
# ipadm refresh-addr net1/dhaddr
```

EXAMPLE 4 Creating IPv6 Addresses

The following sequence of commands auto-configures IPv6 addresses on `bge1` (linkname `net1`) using `in.ndpd` with the default interface ID. A link-local address is configured first, followed by `in.ndpd` adding the stateless and stateful auto-configured addresses.

```
# ipadm create-ip net1
# ipadm create-addr -T addrconf net1/v6addr
```

The following command creates a IPv6 static address. To be able to configure an IPv6 address that is not a link-local address, the interface should already have a link-local address configured on it. It was accomplished by the previous step with `-T addrconf`.

```
# ipadm create-addr -T static -a local=2ff0::f3ad/64 net1/v6static
```

The following command changes the prefix length of an IPv6 address.

```
# ipadm set-addrprop -p prefixlen=80 net1/v6static
```

All the auto-configured addresses and the updated prefix length can be viewed by listing the addresses:

```
# ipadm show-addr
ADDROBJ       TYPE     STATE     ADDR
lo0/v4        static   ok        127.0.0.1/8
lo0/v6        static   ok        ::/128
net1/v6addr   addrconf ok        fe80::203:baff:fe94:2f01/10
net1/v6addr   addrconf ok        2002:a08:39f0:1:203:baff:\
                                            fe94:2f00/64
net1/v6addr   addrconf ok        2001:db8:1:2::402f/128
net1/v6static static   ok        2ff0::f3ad/80
```

EXAMPLE 5 Configuring an IPv4 Tunnel

The first command below (`ipadm`) creates the tunnel source address. Then, a `dladm` command creates the tunnel link. The final `ipadm` commands configure the IPv4 and IPv6 addresses on the tunnel IP interface.

EXAMPLE 5 Configuring an IPv4 Tunnel *(Continued)*

```
# ipadm create-ip net1
# ipadm create-addr -T static -a local=10.2.3.4/24 net1/v4static
# dladm create-iptun -T ipv4 -a local=10.2.3.4,remote=10.2.3.5 tun0
# ipadm create-ip tun0
# ipadm create-addr -T static \
       -a local=173.129.134.1,remote=173.129.134.2 tun0/v4tunaddr
# ipadm create-addr -T static \
       -a local=2ff1::3344,remote=2ff1::3345 tun0/v6tunaddr
# ipadm show-addr
ADDROBJ        TYPE   STATE     ADDR
lo0/v4         static ok        127.0.0.1/8
lo0/v6         static ok        ::/128
net1/v4static  static ok        10.2.3.4/24
tun0/v4tunaddr static ok        173.129.134.1-->173.129.134.2
tun0/v6tunaddr static ok        2ff1::3344-->2ff1::3345
```

EXAMPLE 6 Viewing All of the Interfaces

The following command enables you to view all interfaces.

```
# ipadm show-if -o all
IFNAME  CLASS    STATE    ACTIVE CURRENT       PERSISTENT OVER
lo0     loopback ok       yes    -m-v------46  --46       --
net0    ip       ok       yes    bm--------46  --46       --
e1000g0 ip       ok       yes    bm---l----46  -l46       --
e1000g1 ip       ok       yes    bm---l----46  -l46       --
ipmp0   ipmp     down     yes    bm--------46  --46       e1000g0 e1000g1
tun0    ip       failed   no     -mp-------46  --46       --
vni0    vni      disabled no     bm-v--------  --46       --
```

EXAMPLE 7 Displaying Interface Properties

The following command displays all interface properties for a specified interface.

```
# ipadm show-ifprop net0
IFNAME PROPERTY        PROTO PERM CURRENT PERSISTENT DEFAULT  POSSIBLE
net0   arp             ipv4  rw   on      --         on       on,off
net0   forwarding      ipv4  rw   off     on         off      on,off
net0   metric          ipv4  rw   2       2          0        --
net0   mtu             ipv4  rw   1500    --         1500     68-1500
net0   exchange_routes ipv4  rw   off     --         off      on,off
net0   usesrc          ipv4  rw   none    --         none     --
net0   forwarding      ipv6  rw   off     --         off      on,off
net0   metric          ipv6  rw   2       2          0        --
net0   mtu             ipv6  rw   1500    --         1500     1280-1500
net0   nud             ipv6  rw   on      --         on       on,off
net0   exchange_routes ipv6  rw   off     on         off      on,off
net0   usesrc          ipv6  rw   none    --         none     --
```

EXAMPLE 7 Displaying Interface Properties *(Continued)*

```
net0   group           ip    rw   grp0    --         --       --
net0   standby         ip    r-   off     --         off      on,off
```

EXAMPLE 8 Configuring per-Interface Properties

The following command sets the IPv4 MTU of the interface `net0` to `900`.

```
# ipadm set-ifprop -m ipv4 -p mtu=900 net0
```

The following command sets the IPv6 MTU of the interface `net0` to `1400`.

```
# ipadm set-ifprop -m ipv6 -p mtu 1400 net0
```

View the results:

```
# ipadm show-ifprop -p mtu net0
IFNAME PROPERTY        PROTO PERM CURRENT PERSISTENT DEFAULT  POSSIBLE
net0   mtu             ipv4  rw   900     900        1500     68-1500
net0   mtu             ipv6  rw   1400    1400       1500     1280-1500

# ipadm show-ifprop -m ipv6 -p mtu net0
IFNAME PROPERTY        PROTO PERM CURRENT PERSISTENT DEFAULT  POSSIBLE
net0   mtu             ipv6  rw   1400    1400       1500     1280-1500
```

EXAMPLE 9 Displaying Supported Properties

The following command displays the properties supported on TCP.

```
# ipadm show-prop tcp
PROTO PROPERTY              PERM CURRENT PERSISTENT DEFAULT   POSSIBLE
tcp   ecn                   rw   active  active     passive   never,passive,
                                                              active
tcp   extra_priv_ports      rw   --      1,65535    2049,4045 1-65535
tcp   largest_anon_port     rw   32768   32768      65535     1024-65535
tcp   sack                  rw   active  --         active    never,passive,
                                                              active
tcp   recv_buf              rw   29567   --         49152     2048-1073741824
tcp   send_buf              rw   21354   --         49152     4096-1073741824
tcp   max_buf               ro   65536   --         32768     4096-1073741824
tcp   smallest_anon_port    rw   32768   --         32768     1024-65535
tcp   smallest_nonpriv_port rw   1024    --         1024      1024-32768
```

EXAMPLE 10 Configuring Global IPv4 Forwarding

The following command sequence configures global IPv4 forwarding and overrides that setting for interface `net0`.

```
# ipadm set-prop -p forwarding=on ipv4
# ipadm set-ifprop -p forwarding=off -m ipv4 net0
# ipadm show-prop -p forwarding ipv4
```

EXAMPLE 10 Configuring Global IPv4 Forwarding *(Continued)*

```
PROTO PROPERTY              PERM CURRENT PERSISTENT DEFAULT POSSIBLE
ipv4  forwarding            rw   on      on         off     on,off

# show-ifprop -p forwarding -m ipv4 net0
IFNAME PROPERTY        PROTO PERM CURRENT PERSISTENT DEFAULT  POSSIBLE
net0   forwarding      ipv4  rw   off     off        off      on,off
```

EXAMPLE 11 Using Qualifiers in `set-prop` Subcommand

The following command sequence uses the plus and minus (+, –) qualifiers to add 1047, 1048, and 1049 as extra privileged ports for TCP.

```
# ipadm set-prop -p extra_priv_ports=1047 tcp
# ipadm set-prop -p extra_priv_ports+=1048 tcp
# ipadm set-prop -p extra_priv_ports+=1049 tcp
# ipadm set-prop -p extra_priv_ports+=1050 tcp
```

The following command deletes 1048 as extra privileged port.

```
# ipadm set-prop -p extra_priv_ports-=1048
```

The following command displays all the extra privileged ports for TCP.

```
# ipadm show-prop -p extra_priv_ports tcp
PROTO PROPERTY             PERM CURRENT    PERSISTENT  DEFAULT   POSSIBLE
ipv4  extra_priv_ports     rw   1047,1049, 1047,1049,  2049,4045 1-65535
                                1050       1050
```

EXAMPLE 12 Enabling and Disabling Objects

The following command sequences enables and disables interface and address objects and display the results of those actions.

```
# ipadm create-ip net1
# ipadm create-addr -T static -a local=10.2.3.4/24 net1/v4static
# ipadm set-addrprop -p private=yes net1/v4static
# ipadm show-addr net1/v4static
ADDROBJ        TYPE   STATE     ADDR
net1/v4static  static ok        10.2.3.4/24
```

The following command disables the address object `net1/v4static`.

```
# ipadm disable-addr -t net1/v4static
# ipadm show-addr net1/v4static
ADDROBJ        TYPE   STATE     ADDR
net1/v4static  static ok        10.2.3.4/24
```

The following command disables the interface object `net1`.

EXAMPLE 12 Enabling and Disabling Objects *(Continued)*

```
# ipadm disable-if -t net1
# ipadm show-if net1 -o all
IFNAME     CLASS    STATE    ACTIVE CURRENT       PERSISTENT OVER
net1       ip       disabled no     bm----------  --46       --
```

The following command enables the interface object from the persistent configuration.

```
# ipadm enable-if -t net1
# ipadm show-if net1 -o all
IFNAME     CLASS    STATE    ACTIVE CURRENT       PERSISTENT OVER
net1       ip       ok       yes    bm--------46  --46       --

# ipadm show-addr net1/v4static
ADDROBJ        TYPE   STATE     ADDR
net1/v4static  static ok        10.2.3.4/24
```

Note that when the interface object is enabled all the address objects configured on that interface are enabled also.

The following command creates persistent configuration for the `net0` interface in a non-global exclusive-IP zone so that the `net0` interface will be configured with the set of addresses made available through the `allowed-address` resource from the global zone on the next reboot.

```
# ipadm create-ip net0
```

The `net0` interface can also be configured with the available set of `allowed-address` values in the non-global exclusive-IP zone without a reboot by executing the following commands:

```
# ipadm disable-if -t net0
# ipadm enable-if -t net0
```

EXAMPLE 13 Creating IPMP Interfaces

The following command sequence creates an IPMP interface and adds underlying interfaces to it.

```
# ipadm create-ip e1000g0
# ipadm create-ip e1000g1
# ipadm create-ip e1000g2
# ipadm set-ifprop -p standby=on -m ip e1000g2
# ipadm create-ipmp testgroup0
# ipadm add-ipmp -i e1000g0 -i e1000g1 -i e1000g2 testgroup0
# ipadm create-addr -T static -a local=192.168.80.5/24 testgroup0/data1
# ipadm create-addr -T static -a local=192.168.80.6/24 testgroup0/data2
# ipadm show-if
IFNAME     CLASS    STATE    ACTIVE OVER
lo0        loopback ok       yes    --
```

EXAMPLE 13 Creating IPMP Interfaces *(Continued)*

```
net0       ip       ok       yes    --
e1000g0    ip       ok       yes    --
e1000g1    ip       ok       yes    --
ipmp0      ipmp     ok       yes    e1000g0 e1000g1
```

The following command sequence disables and subsequently enables the IPMP interface.

```
# ipadm disable-if -t testgroup0
ipadm show-if
IFNAME     CLASS    STATE    ACTIVE OVER
lo0        loopback ok       yes    --
net0       ip       ok       yes    --
e1000g0    ip       disabled no     --
e1000g1    ip       disabled no     --
ipmp0      ipmp     disabled no     e1000g0 e1000g1
# ipadm enable-if -t testgroup0
```

The following command sequence removes underlying interface from the IPMP interface and then deletes the IPMP interface.

```
ipadm remove-ipmp -i e1000g0 -i e1000g1 testgroup0
ipadm delete-ipmp testgroup0
```

EXAMPLE 14 Displaying Help

The following command illustrates the use of the `help` subcommand without any arguments.

```
# ipadm help
The following subcommands are supported:
Address subcommands           : create-addr, delete-addr, disable-addr,
                                down-addr, enable-addr, refresh-addr,
                                reset-addrprop, set-addrprop, show-addr,
                                show-addrprop, up-addr
Interface subcommands         : disable-if, enable-if, reset-ifprop,
                                set-ifprop, show-if, show-ifprop
IP interface subcommands      : create-ip, delete-ip
IPMP interface subcommands    : add-ipmp, create-ipmp, delete-ipmp,
                                remove-ipmp
Protocol property subcommands : reset-prop, set-prop, show-prop
VNI interface subcommands     : create-vni, delete-vni
For more info, run: ipadm help subcommand
```

The following command illustrates the use of the `help` subcommand with a subcommand argument.

```
# ipadm help create-ipmp
usage:
    create-ipmp    [-t] [-i under-interface[,...]]
```

EXAMPLE 14 Displaying Help *(Continued)*

```
... IPMP-interface

example:
    # ipadm create-ipmp -i net0,net1 ipmp0
```

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os
Interface Stability	Committed

See Also

`read(1)`, `arp(1M)`, `cfgadm(1M)`, `dhcpagent(1M)`, `dladm(1M)`, `if_mpadm(1M)`, `ifconfig(1M)`, `in.ndpd(1M)`, `in.mpathd(1M)`, `ndd(1M)`, `netadm(1M)`, `netcfg(1M)`, `zonecfg(1M)`, `nsswitch.conf(4)`, `attributes(5)`, `dhcp(5)`, `vni(7d)`

Oracle Solaris Tunable Parameters Reference Manual

Postel, J., RFC 791, *Internet Protocol - DARPA Internet Program Protocol Specification*, Information Sciences Institute, University of Southern California, September 1981.

Hinden, R. and S. Deering, *IP Version 6 Addressing Architecture*, RFC 4291, February 2006.

Thomson, S., Narten, T., and T. Jinmei, *IPv6 Stateless Address AutoConfiguration*, RFC 4862, September 2007.

Droms, R., Bound, J., Volz, B., Lemon, T., Perkins, C., and M. Carney, *Dynamic Host Configuration Protocol for IPv6 (DHCPv6)*, RFC 3315, July 2003.

Narten, T., Draves, R., and S. Krishnan, *Privacy Extensions for Stateless Address AutoConfiguration in IPv6*, RFC 4941, September 2007.

S. Routhier, Ed., *Management Information Base for the Internet Protocol (IP)*, RFC 4293, April 2006

Braden, R., RFC 1122, *Requirements for Internet Hosts - Communication Layers*, Information Sciences Institute, University of Southern California, October 1989.

Name ipf – alter packet filtering lists for IP packet input and output

Synopsis

```
ipf [-6AdDEInoPRrsvVyzZ] [-l block | pass | nomatch]
     [-T optionlist] [-F i | o | a | s | S] -f filename
     [-f filename...]
```

Description The `ipf` utility is part of a suite of commands associated with the Solaris IP Filter feature. See `ipfilter(5)`.

The `ipf` utility opens the filenames listed (treating a hyphen (`-`) as stdin) and parses the file for a set of rules which are to be added or removed from the packet filter rule set.

If there are no parsing problems, each rule processed by `ipf` is added to the kernel's internal lists. Rules are added to the end of the internal lists, matching the order in which they appear when given to `ipf`.

`ipf`'s use is restricted through access to `/dev/ipauth`, `/dev/ipl`, and `/dev/ipstate`. The default permissions of these files require `ipf` to be run as root for all operations.

Enabling Solaris IP Filter Feature

Solaris IP Filter is installed with the Solaris operating system. However, packet filtering is not enabled by default. Use the following procedure to activate the Solaris IP Filter feature.

1. Assume a role that includes the IP Filter Management rights profile (see `rbac(5)`) or become superuser.
2. Configure system and services' firewall policies. See `svc.ipfd(1M)` and `ipf(4)`.
3. (Optional) Create a network address translation (NAT) configuration file. See `ipnat.conf(4)`.
4. (Optional) Create an address pool configuration file. See `ippool(4)`.

   Create an `ipool.conf` file if you want to refer to a group of addresses as a single address pool. If you want the address pool configuration file to be loaded at boot time, create a file called `/etc/ipf/ippool.conf` in which to put the address pool. If you do not want the address pool configuration file to be loaded at boot time, put the `ippool.conf` file in a location other than `/etc/ipf` and manually activate the rules.
5. Enable Solaris IP Filter, as follows:

   ```
   # svcadm enable network/ipfilter
   ```

To re-enable packet filtering after it has been temporarily disabled either reboot the machine or enter the following command:

```
# svcadm enable network/ipfilter
```

...which essentially executes the following `ipf` commands:

1. Enable Solaris IP Filter:

   ```
   # ipf -E
   ```
2. Load `ippools`:

 # ippool -f *<ippool configuration file>*

See ippool(1M).

3. (Optional) Activate packet filtering:

   ipf -f *<ipf configuration file>*

4. (Optional) Activate NAT:

   ipnat -f *<IPNAT configuration file>*

   See ipnat(1M).

Note – If you reboot your system, the IPfilter configuration is automatically activated.

Options The following options are supported:

`-6`
This option is required to parse IPv6 rules and to have them loaded. Loading of IPv6 rules is subject to change in the future.

`-A`
Set the list to make changes to the active list (default).

`-d`
Turn debug mode on. Causes a hex dump of filter rules to be generated as it processes each one.

`-D`
Disable the filter (if enabled). Not effective for loadable kernel versions.

`-E`
Enable the filter (if disabled). Not effective for loadable kernel versions.

`-F i | o | a`
Specifies which filter list to flush. The parameter should either be `i` (input), `o` (output) or `a` (remove all filter rules). Either a single letter or an entire word starting with the appropriate letter can be used. This option can be before or after any other, with the order on the command line determining that used to execute options.

`-F s | S`
To flush entries from the state table, use the `-F` option in conjuction with either `s` (removes state information about any non-fully established connections) or `S` (deletes the entire state table). You can specify only one of these two options. A fully established connection will show up in `ipfstat -s` output as `4/4`, with deviations either way indicating the connection is not fully established.

`-f` *filename*
Specifies which files `ipf` should use to get input from for modifying the packet filter rule lists.

`-I`
Set the list to make changes to the inactive list.

`-l pass | block | nomatch`
Toggles default logging of packets. Valid arguments to this option are `pass`, `block` and `nomatch`. When an option is set, any packet which exits filtering and matches the set category is logged. This is most useful for causing all packets that do not match any of the loaded rules to be logged.

`-n`
Prevents `ipf` from making any ioctl calls or doing anything which would alter the currently running kernel.

`-o`
Force rules by default to be added/deleted to/from the output list, rather than the (default) input list.

`-P`
Add rules as temporary entries in the authentication rule table.

`-R`
Disable both IP address-to-hostname resolution and port number-to-service name resolution.

`-r`
Remove matching filter rules rather than add them to the internal lists.

`-s`
Swap the currently active filter list to be an alternative list.

`-T` *optionlist*
Allows run-time changing of IPFilter kernel variables. To allow for changing, some variables require IPFilter to be in a disabled state (`-D`), others do not. The *optionlist* parameter is a comma-separated list of tuning commands. A tuning command is one of the following:

`list`
Retrieve a list of all variables in the kernel, their maximum, minimum, and current value.

single variable name
Retrieve its current value.

variable name with a following assignment
To set a new value.

Examples follow:

```
# Print out all IPFilter kernel tunable parameters
ipf -T list
```

```
# Display the current TCP idle timeout and then set it to 3600
ipf -D -T fr_tcpidletimeout,fr_tcpidletimeout=3600 -E

# Display current values for fr_pass and fr_chksrc, then set
# fr_chksrc to 1.
ipf -T fr_pass,fr_chksrc,fr_chksrc=1
```

`-v`
Turn verbose mode on. Displays information relating to rule processing.

`-V`
Show version information. This will display the version information compiled into the `ipf` binary and retrieve it from the kernel code (if running or present). If it is present in the kernel, information about its current state will be displayed; for example, whether logging is active, default filtering, and so forth).

`-y`
Manually resync the in-kernel interface list maintained by IP Filter with the current interface status list.

`-z`
For each rule in the input file, reset the statistics for it to zero and display the statistics prior to them being zeroed.

`-Z`
Zero global statistics held in the kernel for filtering only. This does not affect fragment or state statistics.

Files

`/dev/ipauth`
`/dev/ipl`
`/dev/ipstate`
Links to IP Filter pseudo devices.

`/etc/ipf/ipf.conf`
Location of `ipf` startup configuration file. See `ipf(4)`.

`/usr/share/ipfilter/examples/`
Contains numerous IP Filter examples.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	network/ipfilter
Interface Stability	Committed

See Also ipfstat(1M), ipmon(1M), ipnat(1M), ippool(1M), svcadm(1M), svc.ipfd(1M), ipf(4), ipnat.conf(4), ippool(4), attributes(5), ipfilter(5)

System Administration Guide: IP Services

Diagnostics Needs to be run as root for the packet filtering lists to actually be affected inside the kernel.

Name ipfs – saves and restores information for NAT and state tables

Synopsis

```
ipfs [-nv] -l

ipfs [-nv] -u

ipfs [-nv] [-d dirname] -R

ipfs [-nv] [-d dirname] -W

ipfs [-nNSv] [-f filename] -r

ipfs [-nNSv] [-f filename] -w

ipfs [-nNSv] -f filename -i <if1>,<if2>
```

Description The `ipfs` utility enables the saving of state information across reboots. Specifically, the utility allows state information created for NAT entries and rules using "keep state" to be locked (modification prevented) and then saved to disk. Then, after a reboot, that information is restored. The result of this state-saving is that connections are not interrupted.

Options The following options are supported:

`-d` Change the default directory used with `-R` and `-W` options for saving state information.

`-n` Do not take any action that would affect information stored in the kernel or on disk.

`-v` Provides a verbose description of `ipfs` activities.

`-N` Operate on NAT information.

`-S` Operate on filtering state information.

`-u` Unlock state tables in the kernel.

`-l` Lock state tables in the kernel.

`-r` Read information in from the specified file and load it into the kernel. This requires the state tables to have already been locked and does not change the lock once complete.

`-w` Write information out to the specified file and from the kernel. This requires the state tables to have already been locked and does not change the lock once complete.

`-R` Restores all saved state information, if any, from two files, `ipstate.ipf` and `ipnat.ipf`, stored in the `/var/db/ipf` directory. This directory can be changed with the `-d` option. The state tables are locked at the beginning of this operation and unlocked once complete.

`-W` Saves in-kernel state information, if any, out to two files, `ipstate.ipf` and `ipnat.ipf`, stored in the `/var/db/ipf` directory. This directory can be changed with the `-d` option. The state tables are locked at the beginning of this operation and unlocked once complete.

Files

- `/var/db/ipf/ipstate.ipf`
- `/var/db/ipf/ipnat.ipf`
- `/dev/ipl`
- `/dev/ipstate`
- `/dev/ipnat`

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	network/ipfilter
Interface Stability	Committed

See Also `ipf(1M)`, `ipmon(1M)`, `ipnat(1M)`, `attributes(5)`

Diagnostics Arguably, the `-W` and `-R` operations should set the locking and, rather than undo it, restore it to what it was previously.

Fragment table information is currently not saved.

Name ipfstat – reports on packet filter statistics and filter list

Synopsis

```
ipfstat [-6aACdfghIilnoRstv]

ipfstat [-C] [-D addrport] [-P protocol] [-S addrport]
     [-T refreshtime]
```

Description The `ipfstat` command is part of a suite of commands associated with the Solaris IP Filter feature. See `ipfilter(5)`.

The `ipfstat` command examines `/dev/kmem` using the symbols `_fr_flags`, `_frstats`, `_filterin`, and `_filterout`. To run and work, it needs to be able to read both `/dev/kmem` and the kernel itself.

The default behavior of `ipfstat` is to retrieve and display the statistics which have been accumulated over time as the kernel has put packets through the filter.

The role of `ipfstat` is to display current kernel statistics gathered as a result of applying the filters in place (if any) to packets going in and out of the kernel. This is the default operation when no command line parameters are present. When supplied with either `-i` or `-o`, `ipfstat` will retrieve and display the appropriate list of filter rules currently installed and in use by the kernel.

`ipfstat` uses kernel device files to obtain information. The default permissions of these files require `ipfstat` to be run as root for all operations.

The `ipfstat` command supports the `kstat(3KSTAT)` kernel facility. Because of this support, as an alternative to `ipfstat`, you can use `kstat(1M)`. For example:

```
# kstat -m ipf
```

Using the `ipfstat -t` option causes `ipfstat` to enter the state top mode. In this mode the state table is displayed similarly to the way the Unix `top` utility displays the process table. The `-C`, `-D`, `-P`, `-S` and `-T` command line options can be used to restrict the state entries that will be shown and to specify the frequency of display updates.

In state top mode, use the following keys to influence the displayed information:

`d` Select information to display.

`l` Redraw the screen.

`q` Quit the program.

`s` Switch between different sorting criteria.

`r` Reverse the sorting criteria.

States can be sorted by protocol number, by number of IP packets, by number of bytes, and by time-to-live of the state entry. The default is to sort by the number of bytes. States are sorted in descending order, but you can use the `r` key to sort them in ascending order.

It is not possible to interactively change the source, destination, and protocol filters or the refresh frequency. This must be done from the command line.

The screen must have at least 80 columns for correct display. However, `ipfstat` does not check the screen width.

Only the first *X*-5 entries that match the sort and filter criteria are displayed (where *X* is the number of rows on the display). There is no way to see additional entries.

Options The following options are supported:

`-6` Display filter lists and states for IPv6, if available. This option might change in the future.

`-a` Display the accounting filter list and show bytes counted against each rule.

`-A` Display packet authentication statistics.

`-C` Valid only in combination with `-t`. Display "closed" states as well in the top. Normally, a TCP connection is not displayed when it reaches the `CLOSE_WAIT` protocol state. With this option enabled, all state entries are displayed.

`-d` Produce debugging output when displaying data.

`-D` *addrport* Valid only in combination with `-t`. Limit the state top display to show only state entries whose destination IP address and port match the *addrport* argument. The *addrport* specification is of the form *ipaddress*[,*port*]. The *ipaddress* and *port* should be either numerical or the string `any` (specifying any IP address and any port, in that order). If the `-D` option is not specified, it defaults to `-D any,any`.

`-f` Show fragment state information (statistics) and held state information (in the kernel) if any is present.

`-g` Show groups currently configured (both active and inactive).

`-h` Show per-rule the number of times each one scores a "hit". For use in combination with `-i`.

`-i` Display the filter list used for the input side of the kernel IP processing.

`-I` Swap between retrieving `inactive`/`active` filter list details. For use in combination with `-i`.

`-l` When used with `-s`, show a list of active state entries (no statistics).

`-n` Show the rule number for each rule as it is printed.

`-o` Display the filter list used for the output side of the kernel IP processing.

`-P` *protocol* — Valid only in combination with `-t`. Limit the state top display to show only state entries that match a specific protocol. The argument can be a protocol name (as defined in `/etc/protocols`) or a protocol number. If this option is not specified, state entries for any protocol are specified.

`-R` — Disable both IP address-to-hostname resolution and port number-to-service name resolution.

`-S` *addrport* — Valid only in combination with `-t`. Limit the state top display to show only state entries whose source IP address and port match the *addrport* argument. The *addrport* specification is of the form *ipaddress*[,*port*]. The *ipaddress* and *port* should be either numerical or the string `any` (specifying any IP address and any port, in that order). If the `-S` option is not specified, it defaults to `-S any,any`.

`-s` — Show packet/flow state information (statistics only).

`-T` *refreshtime* — Valid only in combination with `-t`. Specifies how often the state `top` display should be updated. The refresh time is the number of seconds between an update. Any positive integer can be used. The default (and minimal update time) is 1.

`-t` — Show the state table in a way similar to the way the Unix utility, `top`, shows the process table. States can be sorted in a number of different ways.

`-v` — Turn verbose mode on. Displays additional debugging information.

Files

- `/dev/kmem`
- `/dev/ksyms`
- `/dev/ipl`
- `/dev/ipstate`

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	network/ipfilter
Interface Stability	Committed

See Also

`ipf(1M)`, `kstat(1M)`, `kstat(3KSTAT)`, `attributes(5)`, `ipfilter(5)`

System Administration Guide: IP Services

Name ipmgmtd – IP network interfaces and TCP/IP tunables management daemon

Synopsis

```
/lib/inet/ipmgmtd

svc:/network/ip-interface-management:default
```

Description `ipmgmtd` is a system daemon that handles administrative events for network IP interfaces and IP/TCP/UDP/SCTP/ICMP tunables. It is controlled through the service management facility (SMF) service instance:

```
svc:/network/ip-interface-management:default
```

The `ipmgmtd` daemon is started automatically by the SMF service and should not be invoked directly. It does not constitute an administrative or a programming interface. The administrative interface for managing IP interfaces and the aforementioned protocol tunables is through `ipadm(1M)`.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os
Interface Stability	Private

See Also `ipadm(1M)`, `attributes(5)`

Name ipmon – monitors /dev/ipl for logged packets

Synopsis

```
ipmon [-abDFhnpstvxX] [-N device] [ [o] [NSI]] [-O [NSI]]
      [-P pidfile] [-S device] [-f device] [filename]
```

Description The `ipmon` command is part of a suite of commands associated with the Solaris IP Filter feature. See `ipfilter(5)`.

The `ipmon` command opens `/dev/ipl` for reading and awaits data to be saved from the packet filter. The binary data read from the device is reprinted in human readable form. However, IP addresses are not mapped back to hostnames, nor are ports mapped back to service names. The output goes to standard output, by default, or a filename, if specified on the command line. Should the `-s` option be used, output is sent instead to `syslogd(1M)`. Messages sent by means of `syslog` have the day, month, and year removed from the message, but the time (including microseconds), as recorded in the log, is still included.

Messages generated by `ipmon` consist of whitespace-separated fields. Fields common to all messages are:

- The date of packet receipt. This is suppressed when the message is sent to `syslog`.
- The time of packet receipt. This is in the form *HH*:*MM*:*SS*.*F*, for hours, minutes, seconds, and fractions of a second (which can be several digits long).
- The name of the interface on which the packet was processed, for example, `ib1`.
- The group and rule number of the rule, for example, `@0:17`. These can be viewed with `ipfstat -in` for input rules or `ipfstat -out` for output rules. See `ipfstat(1M)`.
- The action: `p` for passed, `b` for blocked, `s` for a short packet, `n` did not match any rules, or `L` for a log rule.
- The addresses. This is actually three fields: the source address and port (separated by a comma), the symbol →, and the destination address and port. For example: `209.53.17.22,80 → 198.73.220.17,1722`.
- `PR` followed by the protocol name or number, for example, `PR tcp`.
- `len` followed by the header length and total length of the packet, for example, `len 20 40`.

If the packet is a TCP packet, there will be an additional field starting with a hyphen followed by letters corresponding to any flags that were set. See `ipf.conf(4)` for a list of letters and their flags.

If the packet is an ICMP packet, there will be two fields at the end, the first always being `icmp`, the next being the ICMP message and submessage type, separated by a slash. For example, `icmp 3/3` for a port unreachable message.

Options The following options are supported:

`-a`

Open all of the device logfiles for reading log entries. All entries are displayed to the same output device (stderr or syslog).

`-b`
For rules which log the body of a packet, generate hex output representing the packet contents after the headers.

`-D`
Cause `ipmon` to turn itself into a daemon. Using subshells or backgrounding of `ipmon` is not required to turn it into an orphan so it can run indefinitely.

`-f` *device*
Specify an alternative device/file from which to read the log information for normal IP Filter log records.

`-F`
Flush the current packet log buffer. The number of bytes flushed is displayed, even if the result is zero.

`-h`
Displays usage information.

`-n`
IP addresses and port numbers will be mapped, where possible, back into hostnames and service names.

`-N` *device*
Set the logfile to be opened for reading NAT log records from or to *device*.

`-o` *letter*
Specify which log files from which to actually read data. `N`, NAT logfile; `S`, state logfile; `I`, normal IP Filter logfile. The `-a` option is equivalent to using `-o NSI`.

`-O` *letter*
Specify which log files you do not wish to read from. This is most commonly used in conjunction with the `-a`. Letters available as parameters are the same as for `-o`.

`-p`
Cause the port number in log messages always to be printed as a number and never attempt to look it up.

`-P` *pidfile*
Write the PD of the `ipmon` process to a file. By default this is `/var/run/ipmon.pid`.

`-s`
Packet information read in will be sent through `syslogd` rather than saved to a file. The default facility when compiled and installed is `local0`. The following levels are used:

`LOG_INFO`
Packets logged using the `log` keyword as the action rather than `pass` or `block`.

`LOG_NOTICE`
Packets logged that are also passed.

`LOG_WARNING`
Packets logged that are also blocked.

`LOG_ERR`
Packets that have been logged and that can be considered “short”.

`-S` *device*
Set the logfile to be opened for reading state log records from or to *device*.

`-t`
Read the input file/device in the way performed by `tail(1)`.

`-v`
Show TCP `window`, `ack`, and `sequence` fields

`-x`
Show the packet data in hex.

`-X`
Show the log header record data in hex.

Files
- `/dev/ipl`
- `/dev/ipnat`
- `/dev/ipstate`

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	network/ipfilter
Interface Stability	Committed

See Also `ipf(1M)`, `ipfstat(1M)`, `ipnat(1M)`, `attributes(5)`, `ipfilter(5)`

System Administration Guide: IP Services

Diagnostics `ipmon` expects data that it reads to be consistent with how it should be saved and aborts if it fails an assertion which detects an anomaly in the recorded data.

Name ipmpstat – display IPMP subsystem status

Synopsis

```
ipmpstat [-n] [-o field[,...] [-P]] -a|-g|-i|-p|-t
```

Description The `ipmpstat` command concisely displays information about the IPMP subsystem. It supports five different output modes, each of which provides a different view of the IPMP subsystem (address, group, interface, probe, and target), described below. At most one output mode may be specified per invocation, and the displayed information is guaranteed to be self-consistent. It also provides a parseable output format which may be used by scripts to examine the state of the IPMP subsystem. Only basic privileges are needed to invoke `ipmpstat`, with the exception of probe mode which requires all privileges.

Options The following options are supported:

`-a`
Display IPMP data address information ("address" output mode).

`-g`
Display IPMP group information ("group" output mode).

`-i`
Display IP interface information ("interface" output mode).

`-n`
Display IP addresses numerically, rather than attempting to resolve them to hostnames. This option may be used in any output mode.

`-o` *field*[,...]
Display only the specified output fields, in order. The list of field names is case-insensitive and comma-separated. The field names that are supported depend on the selected output mode, described below. The special field name `all` may be used to display all fields for a given output mode.

`-p`
Display IPMP probe information ("probe" output mode).

`-t`
Display IPMP target information ("target" output mode).

`-P`
Display using a machine-parseable format, described below. If this option is specified, an explicit list of fields must be specified using the `-o` option.

Output Modes The `ipmpstat` utility supports the output modes listed below. Note that these modes map to some of the options described above.

Address Mode
Address mode displays the state of all IPMP data addresses on the system. The following output fields are supported:

ADDRESS
The hostname (or IP address) associated with the information. Note that because duplicate down addresses may exist, the address must be taken together with the GROUP to form a unique identity. For a given IPMP group, if duplicate addresses exist, at most one will be displayed, and an up address will always take precedence.

STATE
The state of the address. Either up if the address is IFF_UP (see ifconfig(1M)), or down if the address is not IFF_UP.

GROUP
The IPMP IP interface hosting the address.

INBOUND
The underlying IP interface that will receive packets for this address. This may change in response to external events such as IP interface failure. If this field is empty, then the system will not accept IP packets sent to this address (for example, because the address is down or because there are no active IP interfaces left in the IPMP group).

OUTBOUND
The underlying IP interfaces that will send packets using this source address. This may change in response to external events such as IP interface failure. If this field is empty, then the system will not send packets with this address as a source (for example, because the address is down or because there are no active IP interfaces left in the IPMP group).

If -o is not specified, all output fields are displayed.

Group Mode
Group mode displays the state of all IPMP groups on the system. The following output fields are supported:

GROUP
The IPMP IP interface name associated with the information. For the anonymous group (see in.mpathd(1M)), this field will be empty.

GROUPNAME
The IPMP group name. For the anonymous group, this field will be empty.

STATE
The state of the group:

ok	All interfaces in the group are usable.
degraded	Some (but not all) interfaces in the group are usable.
failed	No interfaces in the group are usable.

FDT
The probe-based failure detection time. If probe-based failure detection is disabled, this field will be empty.

INTERFACES
The list of underlying IP interfaces in the group. The list is divided into three parts:

1. Active interfaces are listed first and not enclosed in any brackets or parenthesis. Active interfaces are those being used by the system to send or receive data traffic.
2. INACTIVE interfaces are listed next and enclosed in parenthesis. INACTIVE interfaces are those that are functioning, but not being used according to administrative policy.
3. Unusable interfaces are listed last and enclosed in brackets. Unusable interfaces are those that cannot be used at all in their present configuration (for example, FAILED or OFFLINE).

If -o is not specified, all output fields are displayed.

Interface Mode
Interface mode displays the state of all IP interfaces that are tracked by in.mpathd on the system. The following output fields are supported:

INTERFACE
The IP interface name associated with the information.

ACTIVE
Either yes or no, depending on whether the IP interface is being used by the system for IP data traffic.

GROUP
The IPMP IP interface associated with the IP interface. For IP interfaces in the anonymous group (see in.mpathd(1M)), this field will be empty.

FLAGS
Assorted information about the IP interface:

i Unusable due to being INACTIVE.

s Marked STANDBY.

m Nominated to send/receive IPv4 multicast for its IPMP group.

b Nominated to send/receive IPv4 broadcast for its IPMP group.

M Nominated to send/receive IPv6 multicast for its IPMP group.

d Unusable due to being down.

h Unusable due to being brought OFFLINE by in.mpathd because of a duplicate hardware address.

LINK
The state of link-based failure detection:

up
The link is up.

`down`
The link is down.

`unknown`
The network driver does not report link state changes.

`PROBE`
The state of probe-based failure detection:

`ok`
Probes detect no problems.

`failed`
Probes detect failure.

`unknown`
Probes cannot be sent since no suitable probe targets are known.

`disabled`
Probes have been disabled because a unique IP test address has not been configured.

`STATE`
The overall state of the interface:

`ok`
The interface is online and functioning properly based on the configured failure detection methods.

`failed`
The interface is online but has a link state of `down` or a probe state of `failed`.

`offline`
The interface is offline.

`unknown`
The interface is online but may or may not be functioning because the configured failure detection methods are in `unknown` states.

If `-o` is not specified, all output fields are displayed.

Probe Mode
Probe mode displays information about the probes being sent by `in.mpathd`. Unlike other output modes, this mode runs until explicitly terminated using Ctrl-C. The following output fields are supported:

`TIME`
The time the probe was sent, relative to when `ipmpstat` was started. If the probe was sent prior to starting `ipmpstat`, the time will be negative.

PROBE
An identifier representing the probe. The identifier embeds a prefix denoting the probe type, followed by a numerical identifier for the probe. The permissible values for the probe type are:

i ICMP probes

t transitive probes

ICMP probes are sent from active interfaces; the numeric identifier of the probe is incremented for each IP probe sent by `in.mpathd` over a given active interface. The numeric identifier matches the `icmp_seq` field of the ICMP probe packet and can be used for a more detailed analysis by packet monitoring tools.

When the IPMP group does not have any `NOFAILOVER` test addresses configured on any of the interfaces in the group, transitive probes are sent from all interfaces that are not actively being used for receiving data packets. The numeric identifier of transitive probes is incremented for each transitive probe sent from a given interface. The format of a probe packet is internal to the implementation.

INTERFACE
The IP interface the probe was sent on.

TARGET
The hostname (or IP address) of the target to which an ICMP probe is sent from an active interface or the name of the IP interface to which the transitive probe is sent.

NETRTT
The network round-trip-time for the probe. This is the time between when the IP module sends the probe and when the IP module receives the acknowledgment. If `in.mpathd` has concluded that the probe has been lost, this field will be empty.

RTT
The total round-trip-time for the probe. This is the time between when `in.mpathd` starts executing the code to send the probe, and when it completes processing the `ack`. If `in.mpathd` has concluded that the probe has been lost, this field will be empty. Spikes in the total round-trip time that are not present in the network round-trip time indicate that the local system itself is overloaded.

RTTAVG
The average round-trip-time to `TARGET` over `INTERFACE`. This aids identification of slow targets. If there is insufficient data to calculate the average, this field will be empty.

RTTDEV
The standard deviation for the round-trip-time to `TARGET` over `INTERFACE`. This aids identification of jittery targets. If there is insufficient data to calculate the standard deviation, this field will be empty.

If `-o` is not specified, all fields except for `RTTAVG` and `RTTDEV` are displayed.

Target Mode

Target mode displays IPMP probe target information. The following output fields are supported:

`INTERFACE`
The IP interface name associated with the information.

`MODE`
The probe target discovery mode:

`routes`	Probe targets found by means of the routing table.
`multicast`	Probe targets found by means of multicast ICMP probes.
`disabled`	All probe-based failure detection is disabled.
`transitive`	Failure detection is by means of transitive probing, where the health of the IP interface is determined by probing other active interfaces in the group.

`TESTADDR`
The source address used in outgoing probes. Active interfaces that are being used for data traffic, as well as interfaces that have been explicitly configured with `NOFAILOVER` test addresses, will have the hostname (or IP address) that is used for sending and receiving the ICMP probes. All other interfaces in the group will display the name of the interface from which the probes are sent. Note that if an active IP interface is configured with both IPv4 and IPv6 test addresses, probe target information will be displayed separately for each test address.

`TARGETS`
A space-separated list of probe target hostnames (or IP addresses) for ICMP probes, or target interfaces for transitive probes. The IP targets will be listed in firing order, and, if no probe targets could be found, this field will be empty.

If `-o` is not specified, all output fields are displayed.

Output Format By default, `ipmpstat` uses a human-friendly tabular format for its output modes, where each row contains one or more fields of information about a given object, which is in turn uniquely identified by one or more of those fields. In this format, a header identifying the fields is displayed above the table (and after each screenful of information), fields are separated by whitespace, empty fields are represented by `--` (double hyphens), and other visual aids are used. If the value for a field cannot be determined, its value will be displayed as "`?`" and a diagnostic message will be output to standard error.

Machine-parseable format also uses a tabular format, but is designed to be efficient to programmatically parse. Specifically, machine-parseable format differs from human-friendly format in the following ways:

- No headers are displayed.

- Fields with empty values yield no output, rather than showing `--`.
- Fields are separated by a single colon (`:`), rather than variable amounts of whitespace.
- If multiple fields are requested, and a literal `:` or a backslash (`\`) occur in a field's value, they are escaped by prefixing them with `\`.

Examples

EXAMPLE 1 Obtaining Failure Detection Time of a Specific Interface

The following code uses the machine-parseable output format to create a `ksh` function that outputs the failure detection time of a given IPMP IP interface:

```
getfdt() {
    ipmpstat -gP -o group,fdt | while IFS=: read group fdt; do
        [[ "$group" = "$1" ]] && { echo "$fdt"; return; }
    done
}
```

Attributes

See `attributes(5)` for descriptions of the following attributes:

`/usr/sbin/ipmpstat`:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os
Interface Stability	Committed
Machine-Parseable Format	Committed
Human-Friendly Format	Not-an-Interface

`/usr/sbin/ipmpstat` is not a Committed interface.

See Also

`if_mpadm(1M)`, `ifconfig(1M)`, `in.mpathd(1M)`, `attributes(5)`

Name ipnat – user interface to the NAT subsystem

Synopsis

```
ipnat [-CdFhlnRrsv] -f filename
```

Description The `ipnat` utility opens a specified file (treating - as stdin) and parses it for a set of rules that are to be added or removed from the IP NAT.

If there are no parsing problems, each rule processed by `ipnat` is added to the kernel's internal lists. Rules are appended to the internal lists, matching the order in which they appear when given to `ipnat`.

`ipnat`'s use is restricted through access to `/dev/ipauth`, `/dev/ipl`, and `/dev/ipstate`. The default permissions of these files require `ipnat` to be run as root for all operations.

`ipnat`'s use is restricted through access to `/dev/ipnat`. The default permissions of `/dev/ipnat` require `ipnat` to be run as root for all operations.

Options The following options are supported:

`-C`	Delete all entries in the current NAT rule listing (NAT rules).
`-d`	Turn debug mode on. Causes a hex dump of filter rules to be generated as it processes each one.
`-F`	Delete all active entries in the current NAT translation table (currently active NAT mappings).
`-f` *filename*	Parse specified file for rules to be added or removed from the IP NAT. *filename* can be stdin.
`-h`	Print number of hits for each MAP/Redirect filter.
`-l`	Show the list of current NAT table entry mappings.
`-n`	Prevents `ipf` from doing anything, such as making ioctl calls, which might alter the currently running kernel.
`-R`	Disable both IP address-to-hostname resolution and port number-to-service name resolution.
`-r`	Remove matching NAT rules rather than add them to the internal lists.
`-s`	Retrieve and display NAT statistics.
`-v`	Turn verbose mode on. Displays information relating to rule processing and active rules/table entries.

Files

`/dev/ipnat`	Link to IP Filter pseudo device.
`/dev/kmem`	Special file that provides access to virtual address space.
`/etc/ipf/ipnat.conf`	Location of `ipnat` startup configuration file.

`/usr/share/ipfilter/examples/` Contains numerous IP Filter examples.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	network/ipfilter
Interface Stability	Committed

See Also `ipf(1M)`, `ipfstat(1M)`, `ipnat(4)`, `attributes(5)`

Name ippool – user interface to the IP Filter pools

Synopsis

```
ippool -a [-dnv] [-m poolname] [-o role] -i ipaddr
     [/netmask]

ippool -A [-dnv] [-m poolname] [-o role] [-S seed]
     [-t type]

ippool -f file [-dnuv]

ippool -F [-dv] [-o role] [-t type]

ippool -h [-dv] [-m poolname] [-t type]

ippool -l [-dv] [-m poolname] [-t type]

ippool -r [-dnv] [-m poolname] [-o role] -i ipaddr
     [/netmask]

ippool -R [-dnv] [-m poolname] [-o role] [-t type]

ippool -s [-dtv] [-M core] [-N namelist]
```

Description The `ippool` utility is used to manage information stored in the IP pools subsystem of IP Filter software. Configuration file information can be parsed and loaded into the kernel and currently configured pools can be removed, changed, or inspected.

`ippool`'s use is restricted through access to `/dev/ippool`. The default permissions of `/dev/ippool` require `ippool` to be run as root for all operations.

The command line options used are divided into two sections: the global options and the instance-specific options.

`ippool`'s use is restricted through access to `/dev/ipauth`, `/dev/ipl`, and `/dev/ipstate`. The default permissions of these files require `ippool` to be run as root for all operations.

Options `ippool` supports the option categories described below.

Global Options

The following global options are supported:

`-d` Toggle debugging of processing the configuration file.

`-n` Prevents `ippool` from doing anything, such as making ioctl calls, that would alter the currently running kernel.

`-v` Turn verbose mode on.

Instance-Specific Options

The following instance-specific options are supported:

`-a` Add a new data node to an existing pool in the kernel.

`-A` Add a new (empty) pool to the kernel.

`-f` *file* Read in IP pool configuration information from *file* and load it into the kernel.

`-F` Flush loaded pools from the kernel.

`-h` Display a list of pools of the type: hash loaded in the kernel.

`-l` Display a list of pools of the type: tree loaded in the kernel.

`-r` Remove an existing data node from a pool in the kernel.

`-R` Remove an existing pool from within the kernel.

`-s` Display IP pool statistical information.

Other Options The following, additional options are supported:

`-i` *ipaddr*[/*netmask*] Sets the IP address for the operation being undertaken with an all-one's mask or, optionally, a specific netmask, given in either dotted-quad notation or as a single integer.

`-m` *poolname* Sets the pool name for the current operation.

`-M` *core* Specify an alternative path to `/dev/kmem` from which to retrieve statistical information.

`-N` *namelist* Specify an alternative path to lookup symbol name information when retrieving statistical information.

`-o` *role* Sets the role with which this pool is to be used. Currently only `ipf`, `auth`, and `count` are accepted as arguments to this option.

`-S` *seed* Sets the hashing seed to the number specified. For use with hash-type pools only.

`-t` *type* Sets the type of pool being defined. Must be one of `pool`, `hash`, or `group-map`.

`-u` When parsing a configuration file, rather than load new pool data into the kernel, unload it.

Files

`/dev/ippool` Link to IP Filter pseudo device.

`/dev/kmem` Special file that provides access to virtual address space.

`/etc/ipf/ippool.conf` Location of `ippool` startup configuration file.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	network/ipfilter
Interface Stability	Committed

See Also ipf(1M), ipfstat(1M), ippool(4), attributes(5)

Name ipqosconf – configure the IPQoS facility

Synopsis

```
/usr/sbin/ipqosconf

/usr/sbin/ipqosconf -a conf_file [-vs]

/usr/sbin/ipqosconf -c

/usr/sbin/ipqosconf -f

/usr/sbin/ipqosconf -l

/usr/sbin/ipqosconf -L
```

Description The `ipqosconf` utility configures the Quality of Service facility of the Internet Protocol (IP). Only superusers can use this command.

Without arguments, `ipqosconf` displays the actual `IPQoS` configuration.

Configuration is not preserved across reboot. To apply the configuration early in the boot phase, you can populate the `/etc/inet/ipqosinit.conf` file and use the `svcadm(1M)` command to enable the following service:

```
svc:/network/ipqos:default
```

This service is disabled by default.

Options The following options are supported:

`-a` *conf_file*
: Apply the configuration in *conf_file*. If the *conf_file* is –, `ipqosconf` reads from standard input.

`-c`
: Populate the boot file with the current configuration.

`-f`
: Flush the configuration.

`-l`
: List the current applied configuration.

`-L`
: List the current configuration in verbose mode.

  In addition to the information that the `-l` option provides, the `-L` option provides filters and classes configured through other means than the `iqposconf` command. This option also provides the full set of filters that were created by `ipqosconf` by representing a multi-homed host in a configuration file

`-s`
: Log messages to `syslog` during an `-a` operation.

`-v`

Toggle verbose mode during an `-a` operation.

The `-v` option causes all messages to go to the console in addition to their normal destination. Messages intended to go to `syslog`, because the `-s` flag is set or because it is a log message, still go to `syslog` as well as the console.

Configuration File

The configuration file is composed of a format version and a succession of configuration (action) blocks. There are different configuration blocks for each type of action that is being configured.

Format Version

The first line of the configuration file specifies the format version contained in the configuration file.

The following entry specifies the format version:

`fmt_version` *x.x*

where *x.x* is the format version. `1.0` is the only supported version.

Configuration Blocks

Following the format version, are a succession of configuration (action) blocks that are different for each type of action being configured. A configuration block always has the following structure:

```
action {
      name action_name
      module module_name
      params_clause | ""
      cf_clauses
}

action_name      ::= string
module_name      ::= ipgpc | dlcosmk | dscpmk | flowacct | tswtclmt |
                     tokenmt

params_clause    ::= params {
                        parameters
                        params_stats | ""
                     }

parameters       ::= prm_name_value parameters | ""

prm_name_value   ::= param_name  param_value
```

Modules

The *param_name* and the types of *param_value* are specific to a given module.

```
params_stats     ::= global_stats boolean

cf_clauses       ::= class_clause cf_clauses |
                     filter_clause cf_clauses | ""
```

```
class_clause     ::= class {
                         name class_name
                         next_action next_action_name
                         class_stats | ""
                     }

class_name       ::= string
next_action_name ::= string
class_stats      ::= enable_stats boolean
boolean          ::= TRUE | FALSE

filter_clause    ::= filter {
                        name filter_name
                        class class_name
                        parameters
                     }

filter_name      ::= string
```

There must be exactly one configuration block belonging to module `ipgpc`. The action must be named `ipgpc.classify`. All other actions should be reachable from `ipgpc` by way of parameters of type action or the next_action of a class.

The set of types that are used for parameters of the different modules are:

```
action        ::=     string
protocol      ::=     1..255
port          ::=     1..65535
uint8         ::=     0..255
uint32        ::=     0..4294967296
int32         ::=     -2147483648..2147483648
address       ::=    <see the description section>
ifname        ::=    <interface name recognized by SIOGLIFINDEX ioctl>
enum          ::=     string | { string_list }
boolean       ::=     TRUE | FALSE
integer_array ::=     { range_value_list }
map_index     ::=     uint32
address       ::=     ip_address | ip_node_name
user          ::=     uid | username
uid           ::=     0..65535
username      ::=     string
string_list           ::=   string sl_entrys
sl_entrys             ::=   ',' string sl_entrys | ""
range_value_list      ::=   range_value_entry range_value_entrys
range_value_entry     ::=   range ':' integer_array_value
range                 ::=   uint32 '-' uint32
integer_array_value   ::=   string | integer_array_number
integer_array_number ::=   uint8 | uint32
```

```
range_value_entrys   ::=   ';' range_value_entry range_value_entrys | ""
ip_node_name         ::=   string
ip_address           ::=   v4_address | v6_address
v4_address           ::=   v4_ip_address / v4_cidr_mask |
v4_ip_address
v4_cidr_mask         ::=   1-32
v6_address           ::=   v6_ip_address / v6_cidr_mask |
v6_ip_address
v6_cidr_mask         ::=   1-128
```

METER module tokenmt configuration syntax:

```
red_action_name         action
yellow_action_name      action
green_action_name       action
committed_rate          uint32
committed_burst         uint32
peak_rate               uint32
<if present this signifies that this will be a two rate meter, not
   a single rate meter>
peak_burst              uint32
<this is the 'peak' burst size for a two rate meter, but
   the 'excess' burst size for a single rate meter>
color_aware             boolean
color_map               integer_array
global_stats            boolean
```

METER module tswtclmt configuration syntax:

```
red_action_name         action
yellow_action_name      action
green_action_name       action
committed_rate          uint32
peak_rate               uint32
window                  uint32
global_stats            boolean
```

MARKER module dscpmk configuration syntax:

```
next_action         action
dscp_map            int_array
dscp_detailed_stats boolean
global_stats       boolean
```

MARKER module dlcosmk configuration syntax:

```
next_action         action
cos                 map_index
global_stats        boolean
```

CLASSIFIER module ipgpc configuration syntax:

```
user              user
projid            int32
if_name           ifname
direction         enum {
                  LOCAL_IN,
                  LOCAL_OUT,
                  FWD_IN,
                  FWD_OUT}
protocol          protocol
dsfield           uint8
dsfield_mask      uint8
saddr             address
daddr             address
sport             port
dport             port
priority          uint32
precedence        uint32
ip_version        enum {
                  V4,
                  V6 }
global_stats      boolean
```

ACCOUNTING module flowacct configuration syntax:

```
next_action     action
timer           uint32
timeout         uint32
max_limit       uint32
```

Types *action*

A string of characters with a matching action definition. The character string can be up to twenty three characters in length. To allow for spaces the string needs to be enclosed in quotes and cannot span lines. Two special actions are pre-defined and can not have an explicit action definition. The two pre-defined actions are `continue` and `drop`. continue causes the packet that is passed to it to continue normal processing. `drop` causes the packet that is passed to it to be dropped.

address

A machine name or address recognized by `getipnodebyname(3SOCKET)`. If a machine name is specified, and `ip_version` has been defined, the query is done using that address family. If a machine name is not specified and `ip_version` has not been defined, the query is done using the `AI_DEFAULT` flag to `getipnodebyname()(..AF_INET6..)`. `CIDR` address masks following an IP address are allowed. Specify the `CIDR` address masks as `1-32` (for `v4`) or `1-128` (for `v6`). `CIDR` addresses are disallowed for node names.

enum

Either one of the supported values or comma delimited list of support values, enclosed in curly braces.

ifname
A non-`NULL`, existing interface name recognized by the `SIOGLIFINDEX` socket ioctl.

integer_array
A comma delimited set of *range*/*value* pairs, enclosed in curly braces.

Specify *range* in the format *x*-*y*, where *x* and *y* are integers that denote the range of array indexes to which the value applies. The minimum value for both *x* and *y* is `0`. The maximum value for *x* is particular to the parameter. Any array indexes not referred to in the set of ranges are left at their previous value.

map_index
A non-negative integer used as an index into any maps associated with a parameter of this type.

The maximum value of this type is dictated by the number of entries in the associated maps. The index starts at `0`.

port
Either a service name recognized by `getservbyname(3SOCKET)` or an integer `1-65535`.

protocol
Either a protocol name recognized by `getprotobyname(3SOCKET)` or an integer `1-255`.

string
A character string. Enclose *string* in quotes. *string* cannot span multiple lines.

user
Either a valid user ID or username for the system that is being configured.

Parameters The configuration file can contain the following parameters

color_aware
A value of `TRUE` or `FALSE`, indicating whether or not the configured action takes account of the previous packet coloring when classifying.

color_map
An integer array that defines which values of the `dscp` field correspond with which colors for when the `color_aware` parameter is set to `TRUE`.

committed_burst
The committed burst size in bits.

committed_rate
The committed rate in bits per second.

cos
The value used to determine the underlying driver level priority applied to the packet which is defined in `802.1D`.

daddr
The destination address of the datagram.

direction
The value used to build a filter matching only part of the traffic.

This parameter is of type `enum` with valid values of `LOCAL_IN` (local bound traffic), `LOCAL_OUT` (local sourced traffic), `FWD_IN` (forwarded traffic entering the system), and `FWD_OUT` (forwarded traffic exiting the system).

dport
The destination port of the datagram.

dscp_detailed_stats
A value of `TRUE` or `FALSE` that determines whether detailed statistics are switched on for this `dscp` action.

Specify `TRUE` to switch on or `FALSE` to switch off.

dscp_map
The *integer_array* that supplies the values that IP packets with a given `dscp` value have their dscp re-marked with.

The existing value is used to index into the array where the new value is taken from. The array is of size `64`, meaning valid indexes are `0`-`63` and valid values are also `0`-`63`.

dsfield
The `DS` field of the `IP` datagram header. This is an 8–bit value, with each bit position corresponding with the same one in the header; this enables matches to be done on the CU bits. If you specify this parameter, you must also specify the `dsfield_mask` parameter.

`dsfield_mask`
The mask applied to the `dsfield` parameter to determine the bits against which to match. This is an 8-bit value, with each bit position corresponding with the same one in the `dsfield` parameter.

global_stats
A value of `TRUE` or `FALSE` to enable or disable the statistic collection for this action.

green_action_name
The action to be executed for packets that are deemed to be green.

if_name
The name of an interface recognized by the `SIOGLIFINDEX` ioctl. This parameter is of type `ifname`.

ip_version
This parameter is of type `enum` and has valid values of `V4` and `V6`.

If it is set to `V4` only then only `ipv4`addresses are requested for a specified hostname. If it is set to `V6`, only `ipv6` addresses are returned if there are any, otherwise `v4` mapped `v6`

addresses are returned. If both `V4` and `V6` are specified, or if `ip_version` is not specified, then both `ipv4` and `ipv6` addresses are requested for a specified hostname.

max_limit
The maximum number of flow entries present at one time in the `flowacct` actions in the memory resident table.

next_action
The action to be executed when the current action is complete.

This value can be either the name of an action defined in the configuration file, or one of the two special action types: `drop` and `continue`.

peak_burst
The peak burst size, for a two rate meter, or excess burst size, for a single rate meter, in bits.

peak_rate
The peak rate in bits per second.

precedence
An integer that is used to order filters. If there are two matching filters that have the same priority value, the one with the lower precedence value is the one matched. This parameter should be used because the order of the filters in a configuration file has no influence on their relative precedence.

priority
An integer that represents the relative priority of a filter. If there are two matching filters, the one with the higher priority value is the one matched. Multiple filters can have the same priority.

projid
The project ID of the process sending the data. This value is always `-1` for received traffic.

protocol
The Upper Layer Protocol against which this entry is matched.

red_action_name
The action to be executed for packets that are determined to be red.

saddr
The source address of the datagram.

sport
The source port of the datagram.

timeout
The timeout in milliseconds after which flows are written to the accounting file.

timer
The period in milliseconds at which timed-out flows are checked for.

user
The user ID or username of the process sending the data. This value is always `-1` for received traffic.

window
The window size in ms.

yellow_action_name
The action to be executed for packets that are determined to be yellow.

Security None.

Examples **EXAMPLE 1** Sending All Traffic From eng to the AF 1 Class of Service

This example sends all traffic from `eng` to the `AF 1` class of service. It is documented in four separate steps:

The following step creates a `tokenmt` action with three outcomes:

```
#meter for class 1.
action {
        name AF_CL1
        module tokenmt
        params{
                committed_rate 64
                committed_burst 75
                peak_burst 150
                global_stats TRUE
                red_action_name drop
                yellow_action_name markAF12
                green_action_name markAF11
        }
}
```

The following step creates two `dscpmk` actions:

```
#class 1, low drop precedence.
action {
        name markAF11
        module dscpmk
        params{
             dscp_map {0-63:28}
             dscp_detailed_stats TRUE
                global_stats TRUE
                next_action acct1
        }
}
#class 1, medium drop precedence.
action {
        name markAF12
```

EXAMPLE 1 Sending All Traffic From eng to the AF 1 Class of Service *(Continued)*

```
        module dscpmk
        params {
                dscp_map {0-63:30}
             dscp_detailed_stats TRUE
                global_stats TRUE
                next_action acct1
        }
}
```

The following step creates an accounting action:

```
#billing for transmitted class 1 traffic.
action {
        name acct1
        module flowacct
        params {
                timer 10
                timeout 30
                global_stats TRUE
max_limit 1024
next_action continue
        }
}
```

The following step creates an `ipgpc` action:

```
#traffic from eng sent, traffic from ebay dropped.
action {
        name ipgpc.classify
        module ipgpc
        class {
                name from_eng
                enable_stats TRUE
                next_action AF_CL1
        }
        class {
                name from_ebay
                enable_stats TRUE
                next_action drop
        }

        filter {
                name from_eng
                saddr eng-subnet
                class from_eng
        }
        filter {
```

EXAMPLE 1 Sending All Traffic From eng to the AF 1 Class of Service *(Continued)*

```
                name from_ebay
                saddr ebay-subnet
                class from_ebay
        }
}
```

Files

`/etc/inet/ipqosinit.conf`
Contains the IPQoS configuration loaded at boot time. If this file exists, it is read by the `svc:/network/ipqos:default` service.

`/etc/inet/ipqosconf.1.sample`
Sample configuration file for an application server.

`/etc/inet/ipqosconf.2.sample`
Sample configuration file that meters the traffic for a specified application

`/etc/inet/ipqosconf.3.sample`
Sample configuration file that marks the ethernet headers of web traffic with a given user priority

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/network/ipqos
Interface Stability	Obsolete

See Also

`svcs(1)`, `dladm(1M)`, `dlstat(1M)`, `flowadm(1M)`, `flowstat(1M)`, `svcadm(1M)`, `syslog(3C)`, `getipnodebyname(3SOCKET)`, `getprotobyname(3SOCKET)`, `getservbyname(3SOCKET)`, `attributes(5)`, `dlcosmk(7ipp)`, `dscpmk(7ipp)`, `flowacct(7ipp)`, `ipgpc(7ipp)`, `ipqos(7ipp)`, `tokenmt(7ipp)`, `tswtclmt(7ipp)`

Diagnostics

`ipqosconf` sends messages to `syslog` of facility user, severity notice when any changes are made to the `IPQoS` configuration.

Errors that occur during an `ipqosconf` operation send an error message to the console by default. For the application of a new configuration if the `-s` option is set then these messages are sent to `syslog` as facility user, severity error instead. If the `-v` option is present during an application then all error and change notification messages are sent to the console as well as their default destination.

Notes

The `ipqos` service is controlled through the service management facility (SMF) under the service identifier:

`svc:/network/ipqos:default`

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. The service's status can be queried using the `svcs(1)` command.

The IPQoS facility may be removed in a future release. Users are encouraged to migrate to `dladm(1M)`, `dlstat(1M)`, `flowadm(1M)`, and `flowstat(1M)`, which support similar bandwidth resource control features.

Name ipsecalgs – configure the IPsec protocols and algorithms table

Synopsis

```
ipsecalgs

ipsecalgs -l

ipsecalgs -s

ipsecalgs -a [-P protocol-number | -p protocol-name] -k keylen-list
     [-i inc] [-K default-keylen] -b blocklen-list -n alg-names
     -N alg-number -m mech-name [-I initialization-vector_length]
     [-M MAC-length] [-S length-of-salt] [-F flags] [-f] [-s]

ipsecalgs -P protocol-number -p protocol-name
     [-e exec-mode] [-f] [-s]

ipsecalgs -r -p protocol-name [] -n alg-name [-s]

ipsecalgs -r -p protocol-name [] -N alg-number [-s]

ipsecalgs -R -P protocol-number [-s]

ipsecalgs -R -p protocol-name [-s]

ipsecalgs -e exec-mode -P protocol-number [-s]

ipsecalgs -e exec-mode -p protocol-name [-s]
```

Description Use the `ipsecalgs` command to query and modify the IPsec protocol and algorithms stored in `/etc/inet/ipsecalgs`. You can use the `ipsecalgs` command to do the following:

- list the currently defined IPsec protocols and algorithms
- modify IPsec protocols definitions
- modify IPsec algorithms definitions

Never edit the `/etc/inet/ipsecalgs` file manually. The valid IPsec protocols and algorithms are described by the ISAKMP DOI. See *RFC 2407*. In the general sense, a Domain of Interpretation (DOI) defines data formats, network traffic exchange types, and conventions for naming security-relevant information such as security policies or cryptographic algorithms and modes. For `ipsecalgs`, the DOI defines naming and numbering conventions for algorithms and the protocols they belong to. These numbers are defined by the Internet Assigned Numbers Authority (IANA). Each algorithm belongs to a protocol. Algorithm information includes supported key lengths, block or MAC length, and the name of the cryptographic mechanism corresponding to that algorithm. This information is used by the IPsec modules, `ipsecesp(7P)` and `ipsecah(7P)`, to determine the authentication and encryption algorithms that can be applied to IPsec traffic.

The following protocols are predefined:

`IPSEC_PROTO_ESP`	Defines the encryption algorithms (transforms) that can be used by IPsec to provide data confidentiality.
`IPSEC_PROTO_AH`	Defines the authentication algorithms (transforms) that can be used by IPsec to provide authentication.

The mechanism name specified by an algorithm entry must correspond to a valid Solaris Cryptographic Framework mechanism. You can obtain the list of available mechanisms by using the cryptoadm(1M) command.

Applications can retrieve the supported algorithms and their associated protocols by using the functions getipsecalgbyname(3NSL), getipsecalgbynum(3NSL), getipsecprotobyname(3NSL) and getipsecprotobynum(3NSL).

Modifications to the protocols and algorithm by default update only the contents of the /etc/inet/ipsecalgs configuration file. In order for the new definitions to be used for IPsec processing, the changes must be communicated to the kernel using the -s option. See NOTES for a description of how the ipsecalgs configuration is synchronized with the kernel at system restart.

When invoked without arguments, ipsecalgs displays the list of mappings that are currently defined in /etc/inet/ipsecalgs. You can obtain the corresponding kernel table of protocols and algorithms by using the -l option.

Options ipsecalgs supports the following options:

-a
: Adds an algorithm of the protocol specified by the -P option. The algorithm name(s) are specified with the -n option. The supported key lengths and block sizes are specified with the -k, -i, and -b options.

-b
: Specifies the block or MAC lengths of an algorithm, in bytes. Set more than one block length by separating the values with commas.

-e
: Designates the execution mode of cryptographic requests for the specified protocol in the absence of cryptographic hardware provider. See cryptoadm(1M). *exec-mode* can be one of the following values:

  sync — Cryptographic requests are processed synchronously in the absence of a cryptographic hardware provider. This execution mode leads to better latency when no cryptographic hardware providers are available

  async — Cryptographic requests are always processed asynchronously in the absence of cryptographic hardware provider. This execution can improve the resource utilization on a multi-CPU system, but can lead to higher latency when no cryptographic hardware providers are available.

  This option can be specified when defining a new protocol or to modify the execution mode of an existing protocol. By default, the sync execution mode is used in the absence of a cryptographic hardware provider.

`-f`
Used with the `-a` option to force the addition of an algorithm or protocol if an entry with the same name or number already exists.

`-i`
Specifies the valid key length increments in bits. This option must be used when the valid key lengths for an algorithm are specified by a range with the `-k` option.

`-K`
Specifies the default key lengths for an algorithm, in bits. If the `-K` option is not specified, the minimum key length will be determined as follows:

- If the supported key lengths are specified by range, the default key length will be the minimum key length.
- If the supported key lengths are specified by enumeration, the default key length will be the first listed key length.

`-k`
Specifies the supported key lengths for an algorithm, in bits. You can designate the supported key lengths by enumeration or by range.

Without the `-i` option, `-k` specifies the supported key lengths by enumeration. In this case, *keylen-list* consists of a list of one or more key lengths separated by commas, for example:

```
128,192,256
```

The listed key lengths need not be increasing, and the first listed key length will be used as the default key length for that algorithm unless the `-K` option is used.

With the `-i` option, `-k` specifies the range of supported key lengths for the algorithm. The minimum and maximum key lengths must be separated by a dash ('`-`') character, for example:

```
32-448
```

`-l`
Displays the kernel algorithm tables.

`-m`
Specifies the name of the cryptographic framework mechanism corresponding to the algorithm. Cryptographic framework mechanisms are described in the `cryptoadm(1M)` man page.

`-N`
Specifies an algorithm number. The algorithm number for a protocol must be unique. IANA manages the algorithm numbers. See *RFC 2407*.

`-n`
Specifies one or more names for an algorithm. When adding an algorithm with the `-a` option, *alg-names* contains a string or a comma-separated list of strings, for example:

```
des-cbs,des
```

When used with the `-r` option to remove an algorithm, *alg-names* contains one of the valid algorithm names.

`-P`
Adds a protocol of the number specified by *protocol-number* with the name specified by the `-p` option. This option is also used to specify an IPsec protocol when used with the `-a` and the `-R` options. Protocol numbers are managed by the IANA. See *RFC 2407*.

`-p`
Specifies the name of the IPsec protocol.

`-R`
Removes an IPsec protocol from the algorithm table. The protocol can be specified by number by using the `-P` option or by name by using the `-p` option. The algorithms associated with the protocol are removed as well.

`-r`
Removes the mapping for an algorithm. The algorithm can be specified by algorithm number using the `-N` option or by algorithm name using the `-A` option.

`-s`
Synchronizes the kernel with the contents of `/etc/inet/ipsecalgs`. The contents of `/etc/inet/ipsecalgs` are always updated, but new information is not passed on to the kernel unless the `-s` is used. See `NOTES` for a description of how the `ipsecalgs` configuration is synchronized with the kernel at system restart.

The following options allow optional parameters to be configured. These are currently only used for combined mode algorithms, that is, algorithms that provide encryption and authentication in a single operation.

`-I`
The length of the Initialization Vector (IV) in bytes. The default IV length is the same as the block length.

`-M`
The length of the MAC or ICV in bytes for combined mode algorithms.

`-S`
The number of bytes of salt needed by the algorithm. The salt needs to be provided by the key management mechanism.

`-F`
Algorithm flags. These influence the way in which the kernel handles security tasks, especially authentication, in the kernel. They are also used by `ipseckey(1M)` and `ipsecconf(1M)`. Flags can be specified as a comma-separated list of tokens; see the example below. The following tokens are supported:

`COUNTERMODE`
The algorithm uses counter mode.

`COMBINED`
The algorithm provides encryption and authentication in the same operation.

`CCM`
The cryptographic framework mechanism needs a `CK_AES_CCM_PARAMS` structure.

`GMAC`
The cryptographic framework mechanism needs a `CK_AES_GMAC_PARAMS` structure.

`GCM`
The cryptographic framework mechanism needs a `CK_AES_GCM_PARAMS` structure.

`CBC`
This flag indicates the algorithm uses Cipher-block chaining. The cryptographic framework mechanism does not need a params structure. This is also the default, this flag can be omitted.

The algorithm flags can be displayed with the `-l` option.

Examples

EXAMPLE 1 Adding a Protocol for IPsec Encryption

The following example shows how to add a protocol for IPsec encryption:

```
example# ipsecalgs -P 3 -p "IPSEC_PROTO_ESP"
```

EXAMPLE 2 Adding the Blowfish Algorithm

The following example shows how to add the Blowfish algorithm:

```
example# ipsecalgs -a -P 3 -k 32-488 -K 128 -i 8 -n "blowfish" \
  -b 8 -N 7 -m CKM_BF_CBC
```

EXAMPLE 3 Updating the Kernel Algorithm Table

The following example updates the kernel algorithm table with the currently defined protocol and algorithm definitions:

```
example# svcadm refresh ipsecalgs
```

EXAMPLE 4 Adding the AES Galois/Counter Mode (GCM) Algorithm

The following command adds this algorithm.

```
example# ipsecalgs -a -P3 -k 128-256 -K 128 -i 64 -N 20 -b 16 \
-n "aes-gcm16,aes-gcm" -m CKM_AES_GCM -M 16 -I 8 -S 4 \
-F GCM,COMBINED,COUNTER
```

Files `/etc/inet/ipsecalgs`
File that contains the configured IPsec protocols and algorithm definitions. Never edit this file manually.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os
Interface Stability	Committed

See Also `cryptoadm(1M)`, `ipsecconf(1M)`, `ipseckey(1M)`, `svcadm(1M)`, `getipsecalgbyname(3NSL)`, `getipsecprotobyname(3NSL)`, `ike.config(4)`, `attributes(5)`, `smf(5)`, `ipsecah(7P)`, `ipsecesp(7P)`

Piper, Derrell, *RFC 2407, The Internet IP Security Domain of Interpretation for ISAKMP*. Network Working Group. November 1998.

Notes When protocols or algorithm definitions that are removed or altered, services that rely upon these definitions can become unavailable. For example, if the `IPSEC_PROTO_ESP` protocol is removed, then IPsec cannot encrypt and decrypt packets.

Synchronization of the `ipsecalgs` configuration with the kernel at system startup is provided by the following `smf(5)` service:

```
svc:/network/ipsec/ipsecalgs:default
```

The IPsec services are delivered as follows:

```
svc:/network/ipsec/policy:default (enabled)
svc:/network/ipsec/ipsecalgs:default (enabled)
svc:/network/ipsec/manual-key:default (disabled)
svc:/network/ipsec/ike:default (disabled)
```

Services that are delivered disabled are delivered that way because the system administrator must create configuration files for those services before enabling them. See `ipseckey(1M)` and `ike.config(4)`. The default policy for the `policy` service is to allow all traffic to pass without IPsec protection. See `ipsecconf(1M)`.

The correct administrative procedure is to create the configuration file for each service, then enable each service using `svcadm(1M)`, as shown in the following example:

```
example# svcadm enable ipsecalgs
```

The service's status can be queried using the `svcs(1)` command.

If the `ipsecalgs` configuration is modified, the new configuration should be resynchronized as follows:

```
example# svcadm refresh ipsecalgs
```

Administrative actions on this service, such as enabling, disabling, refreshing, and requesting restart can be performed using svcadm(1M). A user who has been assigned the authorization shown below can perform these actions:

```
solaris.smf.manage.ipsec
```

See auths(1), user_attr(4), rbac(5).

The ipsecalgs smf(5) service does not have any user-configurable properties.

The smf(5) framework records any errors in the service-specific log file. Use any of the following commands to examine the logfile property:

```
example# svcs -l ipsecalgs
example# svcprop ipsecalgs
example# svccfg -s ipsecalgs listprop
```

This command requires sys_ip_config privilege to operate and thus can run in the global zone and in exclusive-IP zones. All shared-IP zones share the same available set of algorithms; however, you can use ipsecconf(1M) to set up system policy that uses differing algorithms for various shared-IP zones. All exclusive-IP zones have their own set of algorithms.

Name ipsecconf – configure system wide IPsec policy

Synopsis

```
/usr/sbin/ipsecconf

/usr/sbin/ipsecconf -a file [-q]

/usr/sbin/ipsecconf -c file

/usr/sbin/ipsecconf -d [-i tunnel-name] {index, tunnel-name, index}

/usr/sbin/ipsecconf -f [-i tunnel-name]

/usr/sbin/ipsecconf -F

/usr/sbin/ipsecconf -Fa file [-q]

/usr/sbin/ipsecconf -l [-i tunnel-name] [-n]

/usr/sbin/ipsecconf -L [-n]

/usr/sbin/ipsecconf -r file [-q]
```

Description The `ipsecconf` utility configures the IPsec policy for a host or for one of its tunnels. Once the policy is configured, all outbound and inbound datagrams are subject to policy checks as they exit and enter the host or tunnel. For the host policy, if no entry is found, no policy checks will be completed, and all the traffic will pass through. For a tunnel, if no entry is found and there is at least one entry for the tunnel, the traffic will automatically drop. The difference in behavior is because of the assumptions about IPsec tunnels made in many implementations. Datagrams that are being forwarded will not be subjected to policy checks that are added using this command. See `ifconfig(1M)` and `dladm(1M)` for information on how to protect forwarded packets. Depending upon the match of the policy entry, a specific action will be taken.

This command can be run only by superuser.

Each entry can protect traffic in either one direction (requiring a pair of entries) or by a single policy entry which installs the needed symmetric `sadb` rules.

When the command is issued without any arguments, the list of file policy entries loaded are shown. To display the SPD policy entries use the `-l` option. Both will display the index number for the entry. To specify a single tunnel's SPD, use the `-i` option in combination with `-l`. To specify all SPDs, both host and for all tunnels, use `-L`.

Note, since one file policy entry (FPE) can generate multiple SPD pol entries (SPEs), the list of FPEs may not show all the actual entries. However, it is still useful in determining what what rules have been added to get the spd into its current state.

You can use the `-d` option with the index to delete a given policy in the system. If the `-d` option removes an FPE entry that produces multiple SPEs, only then SPD with the same policy index as the FPE will be removed. This can produce a situation where there may be SPEs when there are no FPEs.

As with `-l`, `-d` can use the `-i` flag to indicate a tunnel. An alternate syntax is to specify a tunnel name, followed by a comma (`,`), followed by an index. For example, `ip.tun0,1`.

With no options, the entries are displayed in the order that they were added, which is not necessarily the order in which the traffic match takes place.

To view the order in which the traffic match will take place, use the `-l` option. The rules are ordered such that all bypass rules are checked first, then ESP rules, then AH rules. After that, they are checked in the order entered.

Policy entries are not preserved across system restarts. Permanent policy entries should be added to `/etc/inet/ipsecinit.conf`. This file is read by the following `smf(5)` service:

```
svc:/network/ipsec/policy
```

See `NOTES` for more information on managing IPsec security policy and `SECURITY` for issues in securing `/etc/inet/ipsecinit.conf`.

Options `ipsecconf` supports the following options:

`-a` *file*
: Add the IPsec policy to the system as specified by each entry in the file. An IPsec configuration file contains one or more entries that specify the configuration. Once the policy is added, all outbound and inbound datagrams are subject to policy checks.

  Entries in the files are described in the Operands section below. Examples can be found in the Examples section below.

  Policy is latched for TCP/UDP sockets on which a `connect(3SOCKET)` or `accept(3SOCKET)` is issued. So, the addition of new policy entries may not affect such endpoints or sockets. However, the policy will be latched for a socket with an existing non-null policy. Thus, make sure that there are no preexisting connections that will be subject to checks by the new policy entries.

  The feature of policy latching explained above may change in the future. It is not advisable to depend upon this feature.

  The default behavior is to append new rules to the existing policy. If a new rule conflicts with an existing rule, an error is reported, the new rule will not be added.

  To add a new rule that conflicts with an existing rule, the existing rule must be removed (see below) or the existing policy flushed. If the policy is flushed, IPsec will not protect any network traffic until a new policy is added.

  The `-F` (flush) flag can be combined with `-a` to perform an atomic policy replacement; the existing policy will be replaced by the new policy described in the config file. By combining these two flags, there is not even a small window when the system is without a policy, which is the case when the flush and add commands are run sequentially.

`-c` *file*
: Check the syntax of the configuration file and report any errors without making any changes to the policy. This option is useful when debugging configurations and when `smf(5)` reports a configuration error. See `SECURITY`.

`-d` *index*
Delete the host policy denoted by the index. The index is obtained by invoking `ipsecconf` without any arguments, or with the `-l` option. See DESCRIPTION for more information. Once the entry is deleted, all outbound and inbound datagrams affected by this policy entry will not be subjected to policy checks. Be advised that with connections for which the policy has been latched, packets will continue to go out with the same policy, even if it has been deleted. It is advisable to use the `-l` option to find the correct policy index.

`-d` *name*,*index*
Delete the policy entry denoted by *index* on a tunnel denoted by *name*. Since tunnels affect traffic that might originate off-node, latching does not apply as it does in the host policy case. Equivalent to: `-d` *index* `-i` *name*.

`-f`
Flush all the policies in the system. Constraints are similar to the `-d` option with respect to latching and host versus per-tunnel behavior.

`-F`
Flush all policies on all tunnels and also flush all host policies. See discussion of combining the `-F` and `-a` options, under `-a`, above.

`-i` *name*
Specify a tunnel interface name for use with the `-d`, `-f`, or `-l` flags.

`-l`
Listing of a single policy table, defaulting to the host policy. When `ipsecconf` is invoked without any arguments, a complete list of policy entries with indexes added by the user since boot is displayed. The current table can differ from the previous one if, for example, a multi-homed entry was added or policy reordering occurred, or if a single rule entry generates two `spd` rules. In the case of a multi-homed entry, all the addresses are listed explicitly. If a mask was not specified earlier but was instead inferred from the address, it will be explicitly listed here. This option is used to view policy entries in the correct order. The outbound and inbound policy entries are listed separately.

`-L`
Lists all policy tables, including host policy and all tunnel instances (including configured but unplumbed).

If `-i` is specified, `-L` lists the policy table for a specific tunnel interface.

`-n`
Show network addresses, ports, protocols in numbers. The `-n` option may only be used with the `-l` option.

`-q`
Quiet mode. Suppresses the warning message generated when adding policies.

`-r` *file*
Remove IPsec policy rules from the system as specified by each entry in *file*. The format of the file contents is the same as is specified with the `-a` option. The file could be created with `ipsecconf -l` and then modified with an editor.

Operands Each policy entry contains three parts specified as follows:

```
{pattern} action {properties}
```

or

```
{pattern} action {properties} ["or" action {properties}]*
```

Every policy entry begins on a new line and can span multiple lines. If an entry exceeds the length of a line, you should split it only within a "braced" section or immediately before the first (left-hand) brace of a braced section. Avoid using the backslash character (\). See EXAMPLES.

The *pattern* section, as shown in the syntax above, specifies the traffic pattern that should be matched against the outbound and inbound datagrams. If there is a match, a specific *action* determined by the second argument will be taken, depending upon the *properties* of the policy entry.

If there is an `or` in the rule (multiple action-properties for a given pattern), a transmitter will use the first action-property pair that works, while a receiver will use any that are acceptable.

pattern and *properties* are name-value pairs where name and value are separated by a <space>, <tab> or <newline>. Multiple name-value pairs should be separated by <space>, <tab> or <newline>. The beginning and end of the pattern and properties are marked by { and } respectively.

Files can contain multiple policy entries. An unspecified name-value pair in the *pattern* will be considered as a wildcard. Wildcard entries match any corresponding entry in the datagram.

One thing to remember is that UDP port 500 is always bypassed regardless of any policy entries. This is a requirement for `in.iked(1M)` to work.

File can be commented by using a # as the first character. Comments may be inserted either at the beginning or the end of a line.

The complete syntax of a policy entry is:

```
policy ::= { <pattern1> } <action1> { <properties1> } |
     { <pattern2> } <action2> { <properties2> }
     [ 'or' <action2> { <properties2>} ]*

     pattern1 ::=  <pattern_name_value_pair1>*

     pattern2 ::=  <pattern_name_value_pair2>*
```

```
action1 ::= apply | permit | bypass | pass
action2 ::=  bypass | pass | drop | ipsec

properties1 ::=   {<prop_name_value_pair1>}
properties2 ::=   {<prop_name_value_pair2>}


pattern_name_value_pair1 ::=
   saddr <address>/<prefix> |
   src  <address>/<prefix> |
   srcaddr <address>/<prefix> |
   smask <mask> |
   sport <port> |
   daddr <address>/<prefix> |
   dst <address>/<prefix> |
   dstaddr <address>/<prefix> |
   dmask <mask> |
   dport <port> |
   ulp <protocol> |
   proto <protocol> |
   type <icmp-type> |
   type <number>-<number> |
   code <icmp-code>
   code <number>-<number>
   tunnel <interface-name> |
   negotiate <tunnel,transport>

pattern_name_value_pair2 ::=
   raddr <address>/<prefix> |
   remote <address>/<prefix> |
   rport <port> |
   laddr <address>/<prefix> |
   local <address>/<prefix> |
   lport <port> |
   ulp <protocol> |
   type <icmp-type> |
   type <number>-<number> |
   code <icmp-code> |
   code <number>-<number>
   proto <protocol>  |
   tunnel <interface-name> |
   negotiate <tunnel,transport> |
   dir <dir_val2>

address ::=  <IPv4 dot notation> | <IPv6 colon notation> |
             <String recognized by gethostbyname>|
             <String recognized by getnetbyname>
```

```
prefix ::=  <number>

mask ::= <0xhexdigit[hexdigit]> | <0Xhexdigit[hexdigit]> |
         <IPv4 dot notation>

port ::= <number>| <String recognized by getservbyname>

protocol ::=  <number>| <String recognized by getprotobyname>

prop_name_value_pair1 ::=
     auth_algs <auth_alg> |
     encr_algs <encr_alg> |
     encr_auth_algs <auth_alg> |
     sa <sa_val> |
     dir <dir_val1>

prop_name_value_pair2 ::=
     auth_algs <auth_alg> |
     encr_algs <encr_alg> |
     encr_auth_algs <auth_alg> |
     sa <sa_val>

auth_alg ::=  <auth_algname> ['(' <keylen> ')']
auth_algname ::= any | md5 | hmac-md5 | sha | sha1 | hmac-sha |
                 hmac-sha1 | hmac-sha256 | hmac-sha384 |
                 hmac-sha512 | aes-gmac128 | aes-gmac192 |
                 aes-gmac256 | <number>

encr_alg ::= <encr_algname> ['(' <keylen> ')']
encr_algname ::= any | aes | aes-cbc | des | des-cbc | 3des |
                 3des-cbc | blowfish | blowfish-cbc | aes-ccm |
                 aes-gcm | aes-none-gmac | <number>

keylen ::= <number> | <number>'..' | '..'<number> | <number>'..' \
<number>

sa_val ::= shared | unique

dir_val1 ::= out | in
dir_val2 ::= out | in | both

number ::= < 0 | 1 | 2 ... 9> <number>
icmp-type ::= <number> | unreach | echo | echorep | squench |
              redir | timex | paramprob | timest | timestrep |
              inforeq | inforep | maskreq | maskrep | unreach6 |
              pkttoobig6 | timex6 | paramprob6 | echo6 | echorep6 |
              router-sol6 | router-ad6 | neigh-sol6 | neigh-ad6 |
```

```
                redir6

    icmp-code ::= <number> | net-unr | host-unr | proto-unr | port-unr |
                needfrag | srcfail | net-unk | host-unk | isolate |
                net-prohib | host-prohib | net-tos | host-tos |
                filter-prohib | host-preced | cutoff-preced |
                no-route6 | adm-prohib6 | addr-unr6 | port-unr6 |
                hop-limex6 | frag-re-timex6 | err-head6 | unrec-head6 |
                unreq-opt6
```

Policy entries may contain the following (name value) pairs in the *pattern* field. Each (name value) pair may appear only once in given policy entry.

laddr/plen
local/plen
: The value that follows is the local address of the datagram with the prefix length. Only plen leading bits of the source address of the packet will be matched. plen is optional. Local means destination on incoming and source on outgoing packets. The source address value can be a hostname as described in getaddrinfo(3SOCKET) or a network name as described in `getnetbyname(3XNET)` or a host address or network address in the Internet standard dot notation. See `inet_addr(3XNET)`. If a hostname is given and getaddrinfo(3SOCKET) returns multiple addresses for the host, then policy will be added for each of the addresses with other entries remaining the same.

raddr/plen
remote/plen
: The value that follows is the remote address of the datagram with the prefix length. Only plen leading bits of the remote address of the packet will be matched. plen is optional. Remote means source on incoming packets and destination on outgoing packets. The remote address value can be a hostname as described in `getaddrinfo(3SOCKET)` or a network name as described in `getnetbyname(3XNET)` or a host address or network address in the Internet standard dot notation. See `inet_addr(3XNET)`. If a hostname is given and `getaddrinfo(3SOCKET)` returns multiple addresses for the host, then policy will be added for each of the addresses with other entries remaining the same.

src/plen
srcaddr/plen
saddr/plen
: The value that follows is the source address of the datagram with the prefix length. Only *plen* leading bits of the source address of the packet will be matched. *plen* is optional.

  The source address value can be a hostname as described in `getaddrinfo(3SOCKET)` or a network name as described in `getnetbyname(3XNET)` or a host address or network address in the Internet standard dot notation. See `inet_addr(3XNET)`.

  If a hostname is given and `getaddrinfo(3SOCKET)` returns multiple addresses for the host, then policy will be added for each of the addresses with other entries remaining the same.

daddr/plen
dest/plen
dstaddr/plen
The value that follows is the destination address of the datagram with the prefix length. Only *plen* leading bits of the destination address of the packet will be matched. *plen* is optional.

See *saddr* for valid values that can be given. If multiple source and destination addresses are found, then a policy entry that covers each source address-destination address pair will be added to the system.

smask
For IPv4 only. The value that follows is the source mask. If prefix length is given with *saddr*, this should not be given. This can be represented either in hexadecimal number with a leading `0x` or `0X`, for example, `0xffff0000`, `0Xffff0000` or in the Internet decimal dot notation, for example, `255.255.0.0` and `255.255.255.0`. The mask should be contiguous and the behavior is not defined for non-contiguous masks.

smask is considered only when *saddr* is given.

For both IPv4 and IPv6 addresses, the same information can be specified as a *slen* value attached to the *saddr* parameter.

dmask
Analogous to *smask*.

lport
The value that follows is the local port of the datagram. This can be either a port number or a string searched with a NULL proto argument, as described in getservbyname(3XNET)

rport
The value that follows is the remote port of the datagram. This can be either a port number or a string searched with a NULL proto argument, as described in getservbyname(3XNET)

sport
The value that follows is the source port of the datagram. This can be either a port number or a string searched with a `NULL` proto argument, as described in `getservbyname(3XNET)`

dport
The value that follows is the destination port of the datagram. This can be either a port number or a string as described in `getservbyname(3XNET)` searched with `NULL` proto argument.

`proto` *ulp*
The value that follows is the Upper Layer Protocol that this entry should be matched against. It could be a number or a string as described in `getprotobyname(3XNET)`. If no smask or plen is specified, a plen of 32 for IPv4 or 128 for IPv6 will be used, meaning a host. If the *ulp* is `icmp` or `ipv6-icmp`, any action applying IPsec must be the same for all `icmp` rules.

`type` *num* or *num-num*
The value that follows is the ICMP type that this entry should be matched against. `type` must be a number from 0 to 255, or one of the appropriate `icmp-type` keywords. Also, *ulp* must be present and must specify either `icmp` or `ipv6-icmp`. A range of types can be specified with a hyphen separating numbers.

`code` *num* or *num-num*
The value that follows is the ICMP code that this entry should be matched against. The value following the keyword `code` must be a number from 0 to 254 or one of the appropriate `icmp-code` keywords. Also, `type` must be present. A range of codes can be specified with a hyphen separating numbers.

`tunnel` *name*
Specifies a tunnel network interface, as configured with `ifconfig(1M)`. If a tunnel of *name* does not yet exist, the policy entries are added anyway, and joined with the tunnel state when it is created. If a tunnel is unplumbed, its policy entries disappear.

`negotiate` *tunnel*
`negotiate` *transport*
For per-tunnel security, specify whether the IPsec SAs protecting the traffic should be tunnel-mode SAs or transport-mode SAs. If transport-mode SAs are specified, no addresses can appear in the policy entry. Transport-mode is backward compatible with Solaris 9, and tunnel IPsec policies configured with `ifconfig(1M)` will show up as transport mode entries here.

Policy entries may contain the following (name-value) pairs in the properties field. Each (name-value) pair may appear only once in a given policy entry.

`auth_algs`
An acceptable value following this implies that IPsec AH header will be present in the outbound datagram. Values following this describe the authentication algorithms that will be used while applying the IPsec AH on outbound datagrams and verified to be present on inbound datagrams. See *RFC 2402*.

This entry can contain either a string or a decimal number.

`string`
This should be either MD5 or HMAC-MD5 denoting the HMAC-MD5 algorithm as described in *RFC 2403*, and SHA1, or HMAC-SHA1 or SHA or HMAC-SHA denoting the HMAC-SHA algorithm described in *RFC 2404*. You can use the `ipsecalgs(1M)` command to obtain the complete list of authentication algorithms.

The string can also be `ANY`, which denotes no-preference for the algorithm. Default algorithms will be chosen based upon the SAs available at this time for manual SAs and the key negotiating daemon for automatic SAs. Strings are not case-sensitive.

`number`
A number in the range 1-255. This is useful when new algorithms can be dynamically loaded.

If *auth_algs* is not present, the AH header will not be present in the outbound datagram, and the same will be verified for the inbound datagram.

`encr_algs`
An acceptable value following this implies that IPsec ESP header will be present in the outbound datagram. The value following this describes the encryption algorithms that will be used to apply the IPsec ESP protocol to outbound datagrams and verify it to be present on inbound datagrams. See *RFC 2406*.

This entry can contain either a string or a decimal number. Strings are not case-sensitive.

`string`
Can be one of the following:

string value:	Algorithm Used:	See RFC:
DES or DES-CBC	DES-CBC	2405
3DES or 3DES-CBC	3DES-CBC	2451
BLOWFISH or BLOWFISH-CBC	BLOWFISH-CBC	2451
AES or AES-CBC	AES-CBC	2451
AES-CCM	AES-CCM	4309
AES-GCM	AES_GCM	4106

You can use the `ipsecalgs(1M)` command to obtain the complete list of authentication algorithms.

The value can be `NULL`, which implies a `NULL` encryption, pursuant to *RFC 2410*. This means that the payload will not be encrypted. The string can also be `ANY`, which indicates no-preference for the algorithm. Default algorithms will be chosen depending upon the SAs available at the time for manual SAs and upon the key negotiating daemon for automatic SAs. Strings are not case-sensitive.

`number`
A decimal number in the range 1-255. This is useful when new algorithms can be dynamically loaded.

`encr_auth_algs`
An acceptable value following `encr_auth_algs` implies that the IPsec ESP header will be present in the outbound datagram. The values following `encr_auth_algs` describe the

authentication algorithms that will be used while applying the IPsec ESP protocol on outbound datagrams and verified to be present on inbound datagrams. See *RFC 2406*. This entry can contain either a string or a number. Strings are case-insensitive.

`string`
Valid values are the same as the ones described for `auth_algs` above.

`number`
This should be a decimal number in the range 1-255. This is useful when new algorithms can be dynamically loaded.

If `encr_algs` is present and `encr_auth_algs` is not present in a policy entry, the system will use an ESP SA regardless of whether the SA has an authentication algorithm or not.

If `encr_algs` is not present and `encr_auth_algs` is present in a policy entry, null encryption will be provided, which is equivalent to `encr_algs` with `NULL`, for outbound and inbound datagrams.

If both `encr_algs` and `encr_auth_algs` are not present in a policy entry, ESP header will not be present for outbound datagrams and the same will be verified for inbound datagrams.

If both `encr_algs` and `encr_auth_algs` are present in a policy entry, ESP header with integrity checksum will be present on outbound datagrams and the same will be verified for inbound datagrams.

For `encr_algs`, `encr_auth_algs`, and `auth_algs` a key length specification may be present. This is either a single value specifying the only valid key length for the algorithm or a range specifying the valid minimum and/or maximum key lengths. Minimum or maximum lengths may be omitted.

`dir`
Values following this decides whether this entry is for outbound or inbound datagram. Valid values are strings that should be one of the following:

`out`
This means that this policy entry should be considered only for outbound datagrams.

`in`
This means that this policy entry should be considered only for inbound datagrams.

`both`
This means that this policy entry should be considered for both inbound and outbound datagrams

This entry is not needed when the action is “apply”, “permit” or “ipsec”. But if it is given while the action is “apply” or “permit”, it should be “out” or “in” respectively. This is mandatory when the action is “bypass”.

`sa`
Values following this decide the attribute of the security association. Value indicates whether a unique security association should be used or any existing SA can be used. If there is a policy requirement, SAs are created dynamically on the first outbound datagram using the key management daemon. Static SAs can be created using `ipseckey(1M)`. The values used here determine whether a new SA will be used/obtained. Valid values are strings that could be one of the following:

`unique`
Unique Association. A new/unused association will be obtained/used for packets matching this policy entry. If an SA that was previously used by the same 5 tuples, that is, {Source address, Destination address, Source port, Destination Port, Protocol (for example, TCP/UDP)} exists, it will be reused. Thus uniqueness is expressed by the 5 tuples given above. The security association used by the above 5 tuples will not be used by any other socket. For inbound datagrams, uniqueness will not be verified.

For tunnel-mode tunnels, `unique` is ignored. SAs are assigned per-rule in tunnel-mode tunnels. For transport-mode tunnels, `unique` is implicit, because the enforcement happens only on the outer-packet addresses and protocol value of either IPv4-in-IP or IPv6-in-IP.

`shared`
Shared association. If an SA exists already for this source-destination pair, it will be used. Otherwise a new SA will be obtained. This is the default.

This is mandatory only for outbound policy entries and should not be given for entries whose action is "bypass". If this entry is not given for inbound entries, for example, when "dir" is in or "action" is permit, it will be assumed to be shared.

Action follows the pattern and should be given before properties. It should be one of the following and this field is mandatory.

`ipsec`
Use IPsec for the datagram as described by the properties, if the pattern matches the datagram. If ipsec is given without a dir spec , the pattern is matched to incoming and outgoing datagrams.

`apply`
Apply IPsec to the datagram as described by the properties, if the pattern matches the datagram. If `apply` is given, the pattern is matched only on the outbound datagram.

`permit`
Permit the datagram if the pattern matches the incoming datagram and satisfies the constraints described by the properties. If it does not satisfy the properties, discard the datagram. If `permit` is given, the pattern is matched only for inbound datagrams.

`bypass`
`pass`

Bypass any policy checks if the pattern matches the datagram. `dir` in the properties decides whether the check is done on outbound or inbound datagrams. All the `bypass` entries are checked before checking with any other policy entry in the system. This has the highest precedence over any other entries. `dir` is the only field that should be present when action is `bypass`.

`drop`
Drop any packets that match the pattern.

If the file contains multiple policy entries, for example, they are assumed to be listed in the order in which they are to be applied. In cases of multiple entries matching the outbound and inbound datagram, the first match will be taken. The system will reorder the policy entry, that is, add the new entry before the old entry, only when:

The level of protection is "stronger" than the old level of protection.

Currently, strength is defined as:

```
AH and ESP > ESP > AH
```

The standard uses of AH and ESP were what drove this ranking of "stronger". There are flaws with this. ESP can be used either without authentication, which will allow cut-and-paste or replay attacks, or without encryption, which makes it equivalent or slightly weaker than AH. An administrator should take care to use ESP properly. See `ipsecesp(7P)` for more details.

If the new entry has `bypass` as action, `bypass` has the highest precedence. It can be added in any order, and the system will still match all the `bypass` entries before matching any other entries. This is useful for key management daemons which can use this feature to bypass IPsec as it protects its own traffic.

Entries with both AH (`auth_algs` present in the policy entry) and ESP (`encr_auth_algs` or `encr_auth_algs` present in the policy entry) protection are ordered after all the entries with AH and ESP and before any AH–only and ESP–only entries. In all other cases the order specified by the user is not modified, that is, newer entries are added at the end of all the old entries. See Examples.

A new entry is considered duplicate of the old entry if an old entry matches the same traffic pattern as the new entry. See Examples for information on duplicates.

Security If, for example, the policy file comes over the wire from an NFS mounted file system, an adversary can modify the data contained in the file, thus changing the policy configured on the machine to suit his needs. Administrators should be cautious about transmitting a copy of the policy file over a network.

To prevent non-privileged users from modifying the security policy, ensure that the configuration file is writable only by trusted users.

The configuration file is defined by a property of the `policy` `smf(5)` service. The default configuration file, is `/etc/inet/ipsecinit.conf`. This can be changed using the `svcprop(1)` command. See `NOTES` for more details.

The policy description language supports the use of tokens that can be resolved by means of a name service, using functions such as `gethostbyname(3NSL)`. While convenient, these functions are only secure as the name service the system is configured to use. Great care should be taken to secure the name service if it is used to resolve elements of the security policy.

If your source address is a host that can be looked up over the network and your naming system itself is compromised, then any names used will no longer be trustworthy.

If the name switch is configured to use a name service that is not local to the system, bypass policy entries might be required to prevent the policy from preventing communication to the name service. See `nsswitch.conf(4)`.

Policy is latched for TCP/UDP sockets on which a `connect(3SOCKET)` or `accept(3SOCKET)` has been issued. Adding new policy entries will not have any effect on them. This feature of latching may change in the future. It is not advisable to depend upon this feature.

The `ipsecconf` command can only be run by a user who has sufficient privilege to open the `pf_key(7P)` socket. The appropriate privilege can be assigned to a user with the Network IPsec Management profile. See `profiles(1)`, `rbac(5)`, `prof_attr(4)`.

Make sure to set up the policies before starting any communications, as existing connections may be affected by the addition of new policy entries. Similarly, do not change policies in the middle of a communication.

Note that certain `ndd` tunables affect how policies configured with this tool are enforced; see `ipsecesp(7P)` for more details.

Examples

EXAMPLE 1 Protecting Outbound TCP Traffic With ESP and the AES Algorithm

The following example specifies that any TCP packet from spiderweb to arachnid should be encrypted with AES, and the SA could be a shared one. It does not verify whether or not the inbound traffic is encrypted.

```
#
# Protect the outbound TCP traffic between hosts spiderweb
# and arachnid with ESP and use AES algorithm.
#
{
     laddr spiderweb
     raddr arachnid
     ulp tcp
     dir out
} ipsec {
```

EXAMPLE 1 Protecting Outbound TCP Traffic With ESP and the AES Algorithm *(Continued)*

```
         encr_algs AES
}
```

EXAMPLE 2 Verifying Whether or Not Inbound Traffic is Encrypted

Example 1 does not verify whether or not the inbound traffic is encrypted. The entry in this example protects inbound traffic:

```
#
# Protect the TCP traffic on inbound with ESP/DES from arachnid
# to spiderweb
#
{
        laddr spiderweb
        raddr arachnid
        ulp tcp
        dir in
} ipsec {
        encr_algs AES
}
```

`sa` can be absent for inbound policy entries as it implies that it can be a shared one. Uniqueness is not verified on inbound. Note that in both the above entries, authentication was never specified. This can lead to cut and paste attacks. As mentioned previously, though the authentication is not specified, the system will still use an ESP SA with `encr_auth_alg` specified, if it was found in the SA tables.

EXAMPLE 3 Protecting All Traffic Between Two Hosts

The following example protects both directions at once:

```
{
        laddr spiderweb
        raddr arachnid
        ulp tcp
} ipsec {
        encr_algs AES
}
```

EXAMPLE 4 Authenticating All Inbound Traffic to the Telnet Port

This entry specifies that any inbound datagram to telnet port should come in authenticated with the SHA1 algorithm. Otherwise the datagram should not be permitted. Without this entry, traffic destined to port number 23 can come in clear. `sa` is not specified, which implies that it is shared. This can be done only for inbound entries. You need to have an equivalent entry to protect outbound traffic so that the outbound traffic is authenticated as well, remove the dir.

EXAMPLE 4 Authenticating All Inbound Traffic to the Telnet Port *(Continued)*

```
#
# All the inbound traffic to the telnet port should be
# authenticated.
#
{
       lport telnet
       dir in
} ipsec {
       auth_algs sha1
}
```

EXAMPLE 5 Verifying Inbound Traffic is Null-Encrypted

The first entry specifies that any packet with address host-B should not be checked against any policies. The second entry specifies that all inbound traffic from network-B should be encrypted with a `NULL` encryption algorithm and the MD5 authentication algorithm. `NULL` encryption implies that ESP header will be used without encrypting the datagram. As the first entry is `bypass` it need not be given first in order, as `bypass` entries have the highest precedence. Thus any inbound traffic will be matched against all `bypass` entries before any other policy entries.

```
#
# Make sure that all inbound traffic from network-B is NULL
# encrypted, but bypass for host-B alone from that network.
# Add the bypass first.
{
raddr host-B
    dir in
} bypass {}

# Now add for network-B.
{
    raddr network-B/16
    dir in
} ipsec {
encr_algs NULL
encr_auth_algs md5
}
```

EXAMPLE 6 Entries to Bypass Traffic from IPsec

The first two entries provide that any datagram leaving the machine with source port 53 or coming into port number 53 should not be subjected to IPsec policy checks, irrespective of any other policy entry in the system. Thus the latter two entries will be considered only for ports other than port number 53.

EXAMPLE 6 Entries to Bypass Traffic from IPsec *(Continued)*

```
#
# Bypass traffic for port no 53
     #
{lport 53} bypass {}
{rport 53} bypass {}
{raddr spiderweb } ipsec {encr_algs any sa unique}
```

EXAMPLE 7 Protecting Outbound Traffic

```
 #
     # Protect the outbound traffic from all interfaces.
     #
{raddr spiderweb dir out} ipsec {auth_algs any sa unique}
```

If the gethostbyname(3XNET) call for spiderweb yields multiple addresses, multiple policy entries will be added for all the source address with the same properties.

```
{
    laddr arachnid
    raddr spiderweb
    dir in
} ipsec {auth_algs any sa unique}
```

If the gethostbyname(3XNET) call for spiderweb and the gethostbyname(3XNET) call for arachnid yield multiple addresses, multiple policy entries will be added for each (saddr daddr) pair with the same properties. Use ipsecconf -l to view all the policy entries added.

EXAMPLE 8 Bypassing Unauthenticated Traffic

```
#
# Protect all the outbound traffic with ESP except any traffic
# to network-b which should be authenticated and bypass anything
# to network-c
#
{raddr network-b/16 dir out} ipsec {auth_algs any}
{dir out} ipsec {encr_algs any}
{raddr network-c/16 dir out} bypass {} # NULL properties
```

Note that bypass can be given anywhere and it will take precedence over all other entries. NULL pattern matches all the traffic.

EXAMPLE 9 Encrypting IPv6 Traffic with 3DES and MD5

The following entry on the host with the link local address fe80::a00:20ff:fe21:4483 specifies that any outbound traffic between the hosts wtih IPv6 link-local addresses fe80::a00:20ff:fe21:4483 and fe80::a00:20ff:felf:e346 must be encrypted with 3DES and MD5.

EXAMPLE 9 Encrypting IPv6 Traffic with 3DES and MD5 *(Continued)*

```
{
    laddr fe80::a00:20ff:fe21:4483
    raddr fe80::a00:20ff:felf:e346
    dir out
} ipsec {
    encr_algs 3DES
    encr_auth_algs MD5
}
```

EXAMPLE 10 Verifying IPv6 Traffic is Authenticated with SHA1

The following two entries require that all IPv6 traffic to and from the IPv6 site-local network `fec0:abcd::0/32` be authenticated with SHA1.

```
{raddr fec0:abcd::0/32} ipsec { auth_algs SHA1 }
```

EXAMPLE 11 Key Lengths

```
# use aes at any key length
{raddr spiderweb} ipsec {encr_algs aes}

# use aes with a 192 bit key
{raddr spiderweb} ipsec {encr_algs aes(192)}

# use aes with any key length up to 192 bits
# i.e. 192 bits or less
{raddr spiderweb} ipsec {encr_algs aes(..192)}

# use aes with any key length of 192 or more
# i.e. 192 bits or more
{raddr spiderweb} ipsec {encr_algs aes(192..)}

#use aes with any key from 192 to 256 bits
{raddr spiderweb} ipsec {encr_algs aes(192..256)}

#use any algorithm with a key of 192 bits or longer
{raddr spiderweb} ipsec {encr_algs any(192..)}
```

EXAMPLE 12 Correct and Incorrect Policy Entries

The following are examples of correctly formed policy entries:

```
{ raddr that_system rport telnet } ipsec { encr_algs 3des encr_auth_algs
sha1 sa shared}

{
        raddr that_system
        rport telnet
```

EXAMPLE 12 Correct and Incorrect Policy Entries *(Continued)*

```
} ipsec {
        encr_algs 3des
        encr_auth_algs sha1
        sa shared
}

{ raddr that_system rport telnet } ipsec
        { encr_algs 3des encr_auth_algs sha1 sa shared}

{ raddr that_system rport telnet } ipsec
        { encr_algs 3des encr_auth_algs sha1 sa shared} or ipsec
        { encr_algs aes encr_auth_algs sha1 sa shared}
```

...and the following is an incorrectly formed entry:

```
{ raddr that_system rport telnet } ipsec
        { encr_algs 3des encr_auth_algs sha1 sa shared}
        or ipsec { encr_algs aes encr_auth_algs sha1 sa shared}
```

In the preceding, incorrect entry, note that the third line begins with "`or ipsec`". Such an entry causes `ipsecconf` to return an error.

EXAMPLE 13 Allowing Neighbor Discovery to Occur in the Clear

The following two entries require that all IPv6 traffic to and from the IPv6 site-local network `fec0:abcd::0/32` be authenticated with SHA1. The second entry allows neighbor discovery to operate correctly.

```
{raddr fec0:abcd::0/32} ipsec { auth_algs SHA1 }
{raddr fec0:abcd::0/32 ulp ipv6-icmp type 133-137  dir both }
    pass { }
```

EXAMPLE 14 Using "or"

The following entry allows traffic using the AES or Blowfish algorithms from the remote machine spiderweb:

```
{raddr spiderweb} ipsec {encr_algs aes} or ipsec {encr_algs blowfish}
```

EXAMPLE 15 Configuring a Tunnel to be Backward-Compatible with Solaris 9

The following example is equivalent to "`encr_algs aes encr_auth_algs md5`" in `ifconfig(1M)`:

```
{tunnel ip.tun0 negotiate transport} ipsec {encr_algs aes
                                                encr_auth_algs md5}
```

EXAMPLE 16 Configuring a Tunnel to a VPN client with an Assigned Address

The following example assumes a distinct "inside" network with its own topology, such that a client's default route goes "inside".

```
# Unlike route(1m), the default route has to be spelled-out.
{tunnel ip.tun0 negotiate tunnel raddr client-inside/32
laddr 0.0.0.0/0} ipsec {encr_algs aes encr_auth_algs sha1}
```

EXAMPLE 17 Transit VPN router between Two Tunnelled Subnets and a Third

The following example specifies a configuration for a VPN router that routes between two tunnelled subnets and a third subnet that is on-link. Consider remote-site A, remote-site B, and local site C, each with a `/24` address allocation.

```
# ip.tun0 between me (C) and remote-site A.
# Cover remote-site A to remote-side B.
{tunnel ip.tun0 negotiate tunnel raddr A-prefix/24 laddr
B-prefix/24} ipsec {encr_algs 3des encr_auth_algs md5}

# Cover remote-site A traffic to my subnet.
{tunnel ip.tun0 negotiate tunnel raddr A-prefix/24 laddr
C-prefix/24} ipsec {encr_algs 3des encr_auth_algs md5}

# ip.tun1 between me (C) and remote-site B.
# Cover remote-site B to remote-site A.
{tunnel ip.tun1 negotiate tunnel raddr B-prefix/24 laddr
A-prefix/24} ipsec {encr_algs aes encr_auth_algs sha1}

# Cover remote-site B traffic to my subnet.
{tunnel ip.tun1 negotiate tunnel raddr B-prefix/24 laddr
C-prefix/24} ipsec {encr_algs aes encr_auth_algs md5}
```

EXAMPLE 18 Using Combined Mode Ciphers

Combined mode ciphers provide data privacy and message authentication in a single operation. They are treated as special versions of `encr_algs`. They provide message authentication without the need to specify `encr_auth_algs`. The two combined mode ciphers supported are:

`aes-ccm`
: AES CCM Mode(Counter with CBC-MAC)

`aes-gcm`
: AES GCM Mode (Galois/Counter)

The parameters used are the same as any other encr_algs value. In both examples, the number in the algorithm token indicates the length of the Integrity Check Vector (ICV). See `ipsecalgs(1M)`.

EXAMPLE 18 Using Combined Mode Ciphers *(Continued)*

```
# simple example using transport mode
{laddr 192.168.99.2 raddr 192.168.99.3} ipsec
    {encr_algs aes-gcm sa shared}
# simple example using CCM mode and 128 bit keys
{laddr 192.168.99.100 raddr 192.168.99.200} ipsec
    {encr_algs aes-ccm(128) sa shared}
```

EXAMPLE 19 Using AES GMAC

The AES GMAC algorithm is a hash algorithm that provides message authentication. An Integrity Check Vector (ICV) is calculated and transmitted as part of the authenticated packet.

The AES GMAC algorithm can be used in ESP mode, in which case the packet data and the ESP header are authenticated. When used with IPsec ESP, AES GMAC can only be specified as an encryption algorithm, even though it does not provide any encryption.

```
# simple example using AES GMAC and 128 bit keys for ESP
{laddr 192.168.99.100 raddr 192.168.99.200} ipsec
    {encr_algs aes-none-gmac(128) sa shared}
```

The above example is analogous to the following invalid example:

```
{laddr 192.168.99.100 raddr 192.168.99.200} ipsec
    {encr_algs null encr_auth_algs aes-gmac(128) sa shared}
```

When used with ESP, the `aes-none-gmac` algorithm takes the key length as an optional argument, the supported key lengths can be displayed using the `ipsecalgs(1M)` command.

The AES GMAC algorithm can be used in AH mode, in which case the whole packet, including the IP header, is authenticated. When used with IPsec AH, AES GMAC can only be specified as an authentication algorithm.

```
# simple example using AES GMAC and 128 bit keys for AH
{laddr 192.168.99.100 raddr 192.168.99.200} ipsec
    {auth_algs aes-gmac128 sa shared}
```

When used with AH, each key length has its own DOI number, the key length does not need to be specified as an argument.

Files `/var/run/ipsecpolicy.conf`
Cache of IPsec policies currently configured for the system, maintained by `ipsecconf` command. Do not edit this file.

`/etc/inet/ipsecinit.conf`
File containing IPsec policies to be installed at system restart by the `policy` `smf(5)` service. See `NOTES` for more information.

`/etc/inet/ipsecinit.sample`
Sample input file for `ipseconf`.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os
Interface Stability	Committed

See Also auths(1), profiles(1), svcprop(1), svcs(1), in.iked(1M), init(1M), ifconfig(1M), ipsecalgs(1M), ipseckey(1M), svcadm(1M), svccfg(1M), gethostbyname(3NSL), accept(3SOCKET), connect(3SOCKET), gethostbyname(3XNET), getnetbyname(3XNET), getprotobyname(3XNET), getservbyname(3XNET), getaddrinfo(3SOCKET), socket(3SOCKET), ike.config(4), nsswitch.conf(4), prof_attr(4), user_attr(4), attributes(5), rbac(5), smf(5), ipsecah(7P), ipsecesp(7P), pf_key(7P)

Glenn, R. and Kent, S. *RFC 2410, The NULL Encryption Algorithm and Its Use With IPsec*. The Internet Society. 1998.

Kent, S. and Atkinson, R. *RFC 2402, IP Authentication Header*.The Internet Society. 1998.

Kent, S. and Atkinson, R. *RFC 2406, IP Encapsulating Security Payload (ESP)*. The Internet Society. 1998.

Madsen, C. and Glenn, R. *RFC 2403, The Use of HMAC-MD5-96 within ESP and AH*. The Internet Society. 1998.

Madsen, C. and Glenn, R. *RFC 2404, The Use of HMAC-SHA-1-96 within ESP and AH*. The Internet Society. 1998.

Madsen, C. and Doraswamy, N. *RFC 2405, The ESP DES-CBC Cipher Algorithm With Explicit IV*. The Internet Society. 1998.

Pereira, R. and Adams, R. *RFC 2451, The ESP CBC-Mode Cipher Algorithms*. The Internet Society. 1998.

Frankel, S. and Kelly, R. Glenn, *The AES Cipher Algorithm and Its Use With IPsec*. 2001.

Diagnostics Bad “string” on line *N*.
Duplicate “string” on line *N*.
string refers to one of the names in pattern or properties. A Bad string indicates that an argument is malformed; a Duplicate string indicates that there are multiple arguments of a similar type, for example, multiple Source Address arguments.

Interface name already selected
Dual use of -i *name* and *name*,*index* for an interface.

Error before or at line *N*.
Indicates parsing error before or at line *N*.

Non-existent index
Reported when the *index* for delete is not a valid one.

spd_msg return: File exists
Reported when there is already a policy entry that matches the traffic of this new entry.

Notes IPsec manual keys are managed by the service management facility, `smf(5)`. The services listed below manage the components of IPsec. These services are delivered as follows:

```
svc:/network/ipsec/policy:default (enabled)
svc:/network/ipsec/ipsecalgs:default (enabled)
svc:/network/ipsec/manual-key:default (disabled)
svc:/network/ipsec/ike:default (disabled)
```

The manual-key service is delivered disabled. The system administrator must create manual IPsec Security Associations (SAs), as described in `ipseckey(1M)`, before enabling that service.

The policy service is delivered enabled, but without a configuration file, so that, as a starting condition, packets are not protected by IPsec. After you create the configuration file `/etc/inet/ipsecinit.conf`, as described in this man page, and refresh the service (`svcadm refresh`, see below), the policy contained in the configuration file is applied. If there is an error in this file, the service enters maintenance mode.

Services that are delivered disabled are delivered that way because the system administrator must create configuration files for those services before enabling them. See `ike.config(4)` for the `ike` service.

See `ipsecalgs(1M)` for the `ipsecalgs` service.

The correct administrative procedure is to create the configuration file for each service, then enable each service using `svcadm(1M)`.

If the configuration needs to be changed, edit the configuration file then refresh the service, as follows:

```
example# svcadm refresh policy
```

The `smf(5)` framework will record any errors in the service-specific log file. Use any of the following commands to examine the `logfile` property:

```
example# svcs -l policy
example# svcprop policy
example# svccfg -s policy listprop
```

The following property is defined for the `policy` service:

```
config/config_file
```

This property can be modified using `svccfg(1M)` by users who have been assigned the following authorization:

```
solaris.smf.value.ipsec
```

See auths(1), user_attr(4), rbac(5).

The service needs to be refreshed using svcadm(1M) before the new property is effective. General non-modifiable properties can be viewed with the svcprop(1) command.

```
# svccfg -s ipsec/policy setprop config/config_file = /new/config_file
# svcadm refresh policy
```

Administrative actions on this service, such as enabling, disabling, refreshing, and requesting restart can be performed using svcadm(1M). A user who has been assigned the authorization shown below can perform these actions:

```
solaris.smf.manage.ipsec
```

The service's status can be queried using the svcs(1) command.

The `ipsecconf` command is designed to be managed by the `policy` smf(5) service. While the `ipsecconf` command can be run from the command line, this is discouraged. If the `ipsecconf` command is to be run from the command line, the `policy` smf(5) service should be disabled first. See svcadm(1M).

Name ipseckey – manually manipulate an IPsec Security Association Database (SADB)

Synopsis

```
ipseckey [-nvp]

ipseckey [-nvp] -f filename

ipseckey -c filename

ipseckey [-nvp] [delete | delete-pair | get] SA_TYPE {EXTENSION value...}

ipseckey [-np] [monitor | passive_monitor | pmonitor]

ipseckey [-nvp] flush {SA_TYPE}

ipseckey [-nvp] dump {SA_TYPE}

ipseckey [-nvp] save SA_TYPE {filename}

ipseckey [-nvp] -s filename
```

Description The `ipseckey` command is used to manually manipulate the security association databases of the network security services, `ipsecah(7P)` and `ipsecesp(7P)`. You can use the `ipseckey` command to set up security associations between communicating parties when automated key management is not available.

While the `ipseckey` utility has only a limited number of general options, it supports a rich command language. The user may specify requests to be delivered by means of a programmatic interface specific for manual keying. See `pf_key(7P)`. When `ipseckey` is invoked with no arguments, it will enter an interactive mode which prints a prompt to the standard output and accepts commands from the standard input until the end-of-file is reached. Some commands require an explicit security association ("SA") type, while others permit the SA type to be unspecified and act on all SA types.

`ipseckey` uses a `PF_KEY` socket and the message types `SADB_ADD`, `SADB_DELETE`, `SADB_GET`, `SADB_UPDATE`, `SADB_FLUSH`, and `SADB_X_PROMISC`. Thus, you must be a superuser to use this command.

`ipseckey` handles sensitive cryptographic keying information. Please read the `Security` section for details on how to use this command securely.

Options

`-c` [*filename*]
Analogous to the `-f` option (see following), except that the input is not executed but only checked for syntactical correctness. Errors are reported to `stderr`. This option is provided to debug configurations without making changes. See `SECURITY` and "Service Management Facility" for more information.

`-f` [*filename*]
Read commands from an input file, *filename*. The lines of the input file are identical to the command line language. The `load` command provides similar functionality. The `-s` option or the `save` command can generate files readable by the `-f` argument.

`-n`
Prevent attempts to print host and network names symbolically when reporting actions. This is useful, for example, when all name servers are down or are otherwise unreachable.

`-p`
Paranoid. Do not print any keying material, even if saving SAs. Instead of an actual hexadecimal digit, print an `X` when this flag is turned on.

`-s` [*filename*]
The opposite of the `-f` option. If '`-`' is given for a *filename*, then the output goes to the standard output. A snapshot of all current `SA` tables will be output in a form readable by the `-f` option. The output will be a series of `add` commands, but with some names not used. This occurs because a single name may often indicate multiple addresses.

`-v`
Verbose. Print the messages being sent into the `PF_KEY` socket, and print raw seconds values for lifetimes.

Commands `add`
Add an SA. Because it involves the transfer of keying material, it cannot be invoked from the shell, lest the keys be visible in `ps(1)` output. It can be used either from the interactive `ipseckey>` prompt or in a command file specified by the `-f` command. The `add` command accepts all extension-value pairs described below.

`update`
Update SA lifetime, and in the cases of larval SAs (leftover from aborted automated key management), keying material and other extensions. Like `add`, this command cannot be invoked from the shell because keying material would be seen by the `ps(1)` command. It can be used either from the interactive `ipseckey>` prompt or in a command file specified by the `-f` command. The `update` command accepts all extension-value pairs, but normally is only used for SA lifetime updates.

`update-pair`
As update, but apply the update to the SA and its paired SA, if there is one.

`delete`
Delete a specific SA from a specific `SADB`. This command requires the `spi` extension, and the `dest` extension for IPsec SAs. Other extension-value pairs are superfluous for a delete message. If the SA to be deleted is paired with another SA, the SA is deleted and the paired SA is updated to indicate that it is now unpaired.

`delete-pair`
Delete a specific SA from a specific SADB. If the SA is paired with another SA, delete that SA too. This command requires the `spi` extension and the `dest` extension for the IPsec SA, or its pair.

`get`
Lookup and display a security association from a specific `SADB`. Like `delete`, this command only requires `spi` and `dest` for IPsec.

`flush`
Remove all SA for a given `SA_TYPE`, or all SA for all types.

`monitor`
Continuously report on any `PF_KEY` messages. This uses the `SADB_X_PROMISC` message to enable messages that a normal `PF_KEY` socket would not receive to be received. See `pf_key(7P)`.

`passive_monitor`
Like monitor, except that it does not use the `SADB_X_PROMISC` message.

`pmonitor`
Synonym for `passive_monitor`.

`dump`
Will display all SAs for a given SA type, or will display all SAs. Because of the large amount of data generated by this command, there is no guarantee that all SA information will be successfully delivered, or that this command will even complete.

`save`
Is the command analog of the `-s` option. It is included as a command to provide a way to snapshot a particular SA type, for example, `esp` or `ah`.

`help`
Prints a brief summary of commands.

SA_TYPE `all`
Specifies all known SA types. This type is only used for the `flush` and `dump` commands. This is equivalent to having no SA type for these commands.

`ah`
Specifies the IPsec Authentication Header ("`AH`") SA.

`esp`
Specifies the IPsec Encapsulating Security Payload ("`ESP`") SA.

Extension Value Types

Commands like `add`, `delete`, `get`, and `update` require that certain extensions and associated values be specified. The extensions will be listed here, followed by the commands that use them, and the commands that require them. Requirements are currently documented based upon the IPsec definitions of an SA. Required extensions may change in the future. `<number>` can be in either hex (`0xnnn`), decimal (`nnn`) or octal (`0nnn`).`<string>` is a text string. `<hexstr>` is a long hexadecimal number with a bit-length. Extensions are usually paired with values; however, some extensions require two values after them.

`spi` *<number>*
Specifies the security parameters index of the `SA`. This extension is required for the `add`, `delete`, `get` and `update` commands.

`pair-spi` <*number*>
When `pair-spi` is used with the `add` or `update` commands, the SA being added or updated will be paired with the SA defined by `pair-spi`. A pair of SAs can be updated or deleted with a single command.

The two SAs that make up the pair need to be in opposite directions from the same pair of IP addresses. The command will fail if either of the SAs specified are already paired with another SA.

If the pair-spi token is used in a command and the SA defined by pair-spi does not exist, the command will fail. If the command was `add` and the pairing failed, the SA to be added will instead be removed.

`inbound | outbound`
These optional flags specify the direction of the SA. When the `inbound` or `outbound` flag is specified with the `add` command, the kernel will insert the new SA into the specified hash table for faster lookups. If the flag is omitted, the kernel will decide into which hash table to insert the new SA based on its knowledge the IP addresses specified with the `src` and `dst` extensions.

When these flags are used with the `update`, `delete`, `update-pair` or `get` commands, the flags provide a hint as to the hash table in which the kernel should find the SA.

`replay` <*number*>
Specifies the replay window size. If not specified, the replay window size is assumed to be zero. It is not recommended that manually added SAs have a replay window. This extension is used by the `add` and `update` commands.

`replay_value` <*number*>
Specifies the replay value of the SA. This extension is used by the `add` and `update` commands.

`state` <*string*>|<*number*>
Specifies the SA state, either by numeric value or by the strings "`larval`", "`mature`", "`dying`" or "`dead`". If not specified, the value defaults to `mature`. This extension is used by the `add` and `update` commands.

`auth_alg` <*string*>|<*number*>
`authalg <string>|<number>`
Specifies the authentication algorithm for an `SA`, either by numeric value, or by strings indicating an algorithm name. Current authentication algorithms include:

`HMAC-MD5`
`md5`, `hmac-md5`

`HMAC-SH-1`
`sha`, `sha-1`, `hmac-sha1`, `hmac-sha`

HMAC-SHA-256
sha256, sha-256, hmac-sha256, hmac-sha-256

HMAC-SHA-384
sha384, sha-384, hmac-sha384, hmac-sha-384

HMAC-SHA-512
sha512, sha-512, hmac-sha512, hmac-sha-512

Often, algorithm names will have several synonyms. This extension is required by the add command for certain SA types. It is also used by the update command.

Use the ipsecalgs(1M) command to obtain the complete list of authentication algorithms.

encr_alg <*string*>|<*number*>
encralg <*string*>|<*number*>
Specifies the encryption algorithm for an SA, either by numeric value, or by strings indicating an algorithm name. Current encryption algorithms include DES ("des"), Triple-DES ("3des"), Blowfish ("blowfish"), and AES ("aes"). This extension is required by the add command for certain SA types. It is also used by the update command.

Use the ipsecalgs(1M) command to obtain the complete list of encryption algorithms.

The next six extensions are lifetime extensions. There are two varieties, "hard" and "soft". If a hard lifetime expires, the SA will be deleted automatically by the system. If a soft lifetime expires, an SADB_EXPIRE message will be transmitted by the system, and its state will be downgraded to dying from mature. See pf_key(7P). The monitor command to key allows you to view SADB_EXPIRE messages.

idle_addtime <*number*>
idle_usetime <*number*>
Specifies the number of seconds that this SA can exist if the SA is not used before the SA is revalidated. If this extension is not present, the default value is half of the hard_addtime (see below). This extension is used by the add and update commands.

soft_bytes <*number*>
hard_bytes <*number*>
Specifies the number of bytes that this SA can protect. If this extension is not present, the default value is zero, which means that the SA will not expire based on the number of bytes protected. This extension is used by the add and update commands.

soft_addtime <*number*>
hard_addtime <*number*>
Specifies the number of seconds that this SA can exist after being added or updated from a larval SA. An update of a mature SA does not reset the initial time that it was added. If this extension is not present, the default value is zero, which means the SA will not expire based on how long it has been since it was added. This extension is used by the add and update commands.

`soft_usetime` <*number*>
`hard_usetime` <*number*>
Specifies the number of seconds this SA can exist after first being used. If this extension is not present, the default value is zero, which means the SA will not expire based on how long it has been since it was added. This extension is used by the `add` and `update` commands.

`saddr` *address* | *name*
`srcaddr` *address* | *name*
`saddr6` *IPv6 address*
`srcaddr6` *IPv6 address*
`src` *address* | *name*
`src6` *IPv6 address*
`srcaddr` *address* and `src` *address* are synonyms that indicate the source address of the SA. If unspecified, the source address will either remain unset, or it will be set to a wildcard address if a destination address was supplied. To not specify the source address is valid for IPsec SAs. Future SA types may alter this assumption. This extension is used by the `add`, `update`, `get` and `delete` commands.

`daddr` <*address*>|<*name*>
`dstaddr` <*address*>|<*name*>
`daddr6` <*IPv6 address*>|<*name*>
`dstaddr6` <*IPv6 address*>|<*name*>
`dst` <*addr*>|<*name*>
`dst6` <*IPv6 address*>|<*name*>
`dstaddr` <*addr*> and `dst` <*addr*> are synonyms that indicate the destination address of the `SA`. If unspecified, the destination address will remain unset. Because IPsec SAs require a specified destination address and `spi` for identification, this extension, with a specific value, is required for the `add`, `update`, `get` and `delete` commands.

If a name is given, `ipseckey` will attempt to invoke the command on multiple SAs with all of the destination addresses that the name can identify. This is similar to how `ipsecconf` handles addresses.

If `dst6` or `dstaddr6` is specified, only the IPv6 addresses identified by a name are used.

`sport` <*portnum*>
`sport` specifies the source port number for an SA. It should be used in combination with an upper-layer protocol (see below), but it does not have to be.

`dport` <*portnum*>
sport specifies the destination port number for an SA. It should be used in combination with an upper-layer protocol (see below), but it does not have to be.

`encap` <*protocol*>
Identifies the protocol used to encapsulate NAT-traversal IPsec packets. Other NAT-traversal parameters (`nat_*`) are below. The only acceptable value for <*protocol*> currently is `udp`.

`proto` <*protocol number*>
`ulp` <*protocol number*>
`proto`, and its synonym `ulp`, specify the IP protocol number of the SA.

`nat_loc` <*address*>|<*name*>
If the local address in the SA (source or destination) is behind a NAT, this extension indicates the NAT node's globally-routable address. This address can match the SA's local address if there is a `nat_lport` (see below) specified.

`nat_rem` <*address*>|<*name*>
If the remote address in the SA (source or destination) is behind a NAT, this extension indicates that node's internal (that is, behind-the-NAT) address. This address can match the SA's local address if there is a `nat_rport` (see below) specified.

`nat_lport` <*portnum*>
Identifies the local UDP port on which encapsulation of ESP occurs.

`nat_rport` <*portnum*>
Identifies the remote UDP port on which encapsulation of ESP occurs.

`isrc` <*address*> | <*name*>[/<*prefix*>]
`innersrc` <*address*> | <*name*>[/<*prefix*>]
`isrc6` <*address*> | <*name*>[/<*prefix*>]
`innersrc6` <*address*> | <*name*>[/<*prefix*>]
`proxyaddr` <*address*> | <*name*>[/<*prefix*>]
`proxy` <*address*> | <*name*>[/<*prefix*>]
`isrc` <*address*>[/<*prefix*>] and `innersrc` <*address*>[/<*prefix*>] are synonyms. They indicate the inner source address for a tunnel-mode SA.

An inner-source can be a prefix instead of an address. As with other address extensions, there are IPv6-specific forms. In such cases, use only IPv6-specific addresses or prefixes.

Previous versions referred to this value as the proxy address. The usage, while deprecated, remains.

`idst` <*address*> | <*name*>[/<*prefix*>]
`innerdst` <*address*> | <*name*>[/<*prefix*>]
`idst6` <*address*> | <*name*>[/<*prefix*>]
`innerdst6` <*address*> | <*name*>[/<*prefix*>]
`idst` <*address*>[/<*prefix*>] and `innerdst` <*address*>[/<*prefix*>] are synonyms. They indicate the inner destination address for a tunnel-mode SA.

An inner-destination can be a prefix instead of an address. As with other address extensions, there are IPv6-specific forms. In such cases, use only IPv6-specific addresses or prefixes.

`innersport` <*portnum*>
`isport` <*portnum*>

`innersport` specifies the source port number of the inner header for a tunnel-mode SA. It should be used in combination with an upper-layer protocol (see below), but it does not have to be.

`innerdport` <*portnum*>
`idport` <*portnum*>
`innerdport` specifies the destination port number of the inner header for a tunnel-mode SA. It should be used in combination with an upper-layer protocol (see below), but it does not have to be.

`iproto` <*protocol number*>`iulp` <*protocol number*>
`iproto`, and its synonym `iulp`, specify the IP protocol number of the inner header of a tunnel-mode SA.

`authkey` <*hexstring*>
Specifies the authentication key for this SA. The key is expressed as a string of hexadecimal digits, with an optional `/` at the end, for example, `123/12`. Bits are counted from the most-significant bits down. For example, to express three '1' bits, the proper syntax is the string "`e/3`". For multi-key algorithms, the string is the concatenation of the multiple keys. This extension is used by the `add` and `update` commands.

`encrkey` <*hexstring*>
Specifies the encryption key for this SA. The syntax of the key is the same as `authkey`. A concrete example of a multi-key encryption algorithm is `3des`, which would express itself as a 192-bit key, which is three 64-bit parity-included `DES` keys. This extension is used by the `add` and `update` commands.

`reserved_bits` <*number*>
The last <*number*> bits of the `encrkey` string are marked as reserved in the `PF_KEY` message. This option is only for testing certain encryption algorithms.

Certificate identities are very useful in the context of automated key management, as they tie the `SA` to the public key certificates used in most automated key management protocols. They are less useful for manually added SAs. Unlike other extensions, `srcidtype` takes two values, a *type*, and an actual *value*. The type can be one of the following:

`prefix`
An address prefix.

`fqdn`
A fully-qualified domain name.

`domain`
Domain name, synonym for `fqdn`.

`user_fqdn`
User identity of the form *user*@*fqdn*.

`mailbox`
Synonym for `user_fqdn`.

The *value* is an arbitrary text string that should identify the certificate.

`srcidtype` <*type, value*>
Specifies a source certificate identity for this SA. This extension is used by the `add` and `update` commands.

`dstidtype` <*type, value*>
Specifies a destination certificate identity for this SA. This extension is used by the `add` and `update` commands

Label extensions are used on Trusted Extensions to associate sensitivity labels with the traffic carried inside a security association. These extensions are not allowed unless Trusted Extensions is enabled.

`label` *label*
Defines the sensitivity label of traffic carried by this SA. Disallowed on systems not using Trusted Extensions.

`outer-label` *label*
Defines the sensitivity of the ciphertext traffic belonging to this SA; this label will appear in the outer packet header. Disallowed on systems not using Trusted Extensions. Incorrect use of this extension might allow label policy to be circumvented.

`implicit-label` *label*
Defines the sensitivity of the ciphertext traffic belonging to this SA and request that this SA not contain an explicit on-the-wire label. Disallowed on systems not using Trusted Extensions. Incorrect use of this extension might allow label policy to be circumvented.

Tunnel Mode versus Transport Mode SAs

An IPsec SA is a Tunnel Mode SA if the "proto" value is either 4 (`ipip`) or 41 (`ipv6`) *and* there is an inner-address or inner-port value specified. Otherwise, the SA is a Transport Mode SA.

Security

Keying material is very sensitive and should be generated as randomly as possible. Some algorithms have known weak keys. IPsec algorithms have built-in weak key checks, so that if a weak key is in a newly added `SA`, the `add` command will fail with an invalid value.

The `ipseckey` command allows a privileged user to enter cryptographic keying information. If an adversary gains access to such information, the security of IPsec traffic is compromised. The following issues should be taken into account when using the `ipseckey` command.

1. Is the TTY going over a network (interactive mode)?
   - If it is, then the security of the keying material is the security of the network path for this TTY's traffic. Using `ipseckey` over a clear-text `telnet` or `rlogin` session is risky.
   - Even local windows might be vulnerable to attacks where a concealed program that reads window events is present.
2. Is the file accessed over the network or readable to the world (`-f` option)?
   - A network-mounted file can be sniffed by an adversary as it is being read.
   - A world-readable file with keying material in it is also risky.

3. The `ipseckey` command is designed to be managed by the `manual-key` `smf(5)` service. Because the `smf(5)` log files are world-readable, the `ipseckey` does not record any syntax errors in the log files, as these errors might include secret information.

   If a syntax error is found when the `manual-key` `smf(5)` service is enabled, the service enters maintenance mode. The log file will indicate that there was a syntax error, but will not specify what the error was.

   The administrator should use `ipseckey -c` *filename* from the command line to discover the cause of the errors. See `OPTIONS`.

If your source address is a host that can be looked up over the network and your naming system itself is compromised, then any names used will not be trustworthy.

Security weaknesses often lie in misapplication of tools, not in the tools themselves. Administrators are urged to be cautious when using `ipseckey`. The safest mode of operation is probably on a console or other hard-connected TTY.

For further thoughts on this subject, see the afterward by Matt Blaze in Bruce Schneier's *Applied Cryptography: Protocols, Algorithms, and Source Code in C*.

Service Management Facility

IPsec manual keys are managed by the service management facility, `smf(5)`. The services listed below manage the components of IPsec. These services are delivered as follows:

```
svc:/network/ipsec/policy:default (enabled)
svc:/network/ipsec/ipsecalgs:default (enabled)
svc:/network/ipsec/manual-key:default (disabled)
svc:/network/ipsec/ike:default (disabled)
```

The manual-key service is delivered disabled. The system administrator must create manual IPsec Security Associations (SAs), as described in this man page, before enabling that service.

The policy service is delivered enabled, but without a configuration file, so that, as a starting condition, packets are not protected by IPsec. After you create the configuration file `/etc/inet/ipsecinit.conf` and refresh the service (`svcadm refresh`, see below), the policy contained in the configuration file is applied. If there is an error in this file, the service enters maintenance mode. See `ipsecconf(1M)`.

Services that are delivered disabled are delivered that way because the system administrator must create configuration files for those services before enabling them. See `ike.config(4)` for the `ike` service.

See `ipsecalgs(1M)` for the `ipsecalgs` service.

The correct administrative procedure is to create the configuration file for each service, then enable each service using `svcadm(1M)`.

If the configuration needs to be changed, edit the configuration file then refresh the service, as follows:

```
example# svcadm refresh manual-key
```

Warning: To prevent `ipseckey` complaining about duplicate Associations, the `ipseckey` command flushes the Security Association Data Base (SADB) when the `ipseckey` command is run from `smf(5)`, before adding any new Security Associations defined in the configuration file. This differs from the command line behavior where the SADB is not flushed before adding new Security Associations.

The `smf(5)` framework will record any errors in the service-specific log file. Use any of the following commands to examine the `logfile` property:

```
example# svcs -l manual-key
example# svcprop manual-key
example# svccfg -s manual-key listprop
```

The following property is defined for the `manual-key` service:

```
config/config_file
```

This property can be modified using `svccfg(1M)` by users who have been assigned the following authorization:

```
solaris.smf.value.ipsec
```

See `auths(1)`, `user_attr(4)`, `rbac(5)`.

The service needs to be refreshed using `svcadm(1M)` before the new property is effective. General non-modifiable properties can be viewed with the `svcprop(1)` command.

```
# svccfg -s ipsec/manual-key setprop config/config_file = \
/new/config_file
# svcadm refresh manual-key
```

Administrative actions on this service, such as enabling, disabling, refreshing, and requesting restart can be performed using `svcadm(1M)`. A user who has been assigned the authorization shown below can perform these actions:

```
solaris.smf.manage.ipsec
```

The service's status can be queried using the `svcs(1)` command.

The `ipseckey` command is designed to be run under `smf(5)` management. While the `ipsecconf` command can be run from the command line, this is discouraged. If the `ipseckey` command is to be run from the command line, the `manual-key` `smf(5)` service should be disabled first. See `svcadm(1M)`.

Note that, unlike IPsec and IKE policy and configuration, IPsec keying material is *NOT* managed by location profiles. This data is assumed to be for the system in general, and will not change when the location changes.

Examples

EXAMPLE 1 Emptying Out All SAs

To empty out all SA:

```
example# ipseckey flush
```

EXAMPLE 2 Flushing Out IPsec AH SAs Only

To flush out only IPsec AH SAs:

```
example# ipseckey flush ah
```

EXAMPLE 3 Saving All SAs To Standard Output

To save all SAs to the standard output:

```
example# ipseckey save all
```

EXAMPLE 4 Saving ESP SAs To The File /tmp/snapshot

To save ESP SAs to the file /tmp/snapshot:

```
example# ipseckey save esp /tmp/snapshot
```

EXAMPLE 5 Deleting an IPsec SA

To delete an IPsec SA, only the SPI and the destination address are needed:

```
example# ipseckey delete esp spi 0x2112 dst 224.0.0.1
```

An alternative would be to delete the SA and the SAs pair if it has one:

```
example# ipseckey delete-pair esp spi 0x2112 dst 224.0.0.1
```

EXAMPLE 6 Getting Information on an IPsec SA

Likewise, getting information on a SA only requires the destination address and SPI:

```
example# ipseckey get ah spi 0x5150 dst mypeer
```

EXAMPLE 7 Adding or Updating IPsec SAs

Adding or updating SAs requires entering interactive mode:

```
example# ipseckey
ipseckey> add ah spi 0x90125 src me.domain.com dst you.domain.com \
          authalg md5 authkey 1234567890abcdef1234567890abcdef
ipseckey> update ah spi 0x90125 dst you.domain.com hard_bytes \
          16000000
ipseckey> exit
```

Adding two SAs that are linked together as a pair:

```
example# ipseckey
ipseckey> add esp spi 0x2345 src me.domain.com dst you.domain.com \
   authalg md5 authkey bde359723576fdea08e56cbe876e24ad \
```

EXAMPLE 7 Adding or Updating IPsec SAs *(Continued)*

```
   encralg des encrkey be02938e7def2839
ipseckey> add esp spi 0x5432 src me.domain.com dst you.domain.com \
   authalg md5 authkey bde359723576fdea08e56cbe876e24ad \
   encralg des encrkey be02938e7def2839 pair-spi 0x2345
ipseckey> exit
```

EXAMPLE 8 Adding an SA in the Opposite Direction

In the case of IPsec, SAs are unidirectional. To communicate securely, a second SA needs to be added in the opposite direction. The peer machine also needs to add both SAs.

```
example# ipseckey
ipseckey> add ah spi 0x2112 src you.domain.com dst me.domain.com \
          authalg md5 authkey bde359723576fdea08e56cbe876e24ad \
          hard_bytes 16000000
ipseckey> exit
```

EXAMPLE 9 Monitoring PF_KEY Messages

Monitoring for `PF_KEY` messages is straightforward:

```
example# ipseckey monitor
```

EXAMPLE 10 Using Commands in a File

Commands can be placed in a file that can be parsed with the `-f` option. This file may contain comment lines that begin with the "#" symbol. For example:

```
# This is a sample file for flushing out the ESP table and
# adding a pair of SAs.

flush esp

### Watch out!  I have keying material in this file.  See the
### SECURITY section in this manual page for why this can be
### dangerous .

add esp spi 0x2112 src me.domain.com dst you.domain.com \
    authalg md5 authkey bde359723576fdea08e56cbe876e24ad \
    encralg des encrkey be02938e7def2839 hard_usetime 28800
add esp spi 0x5150 src you.domain.com dst me.domain.com \
    authalg md5 authkey 930987dbe09743ade09d92b4097d9e93 \
    encralg des encrkey 8bd4a52e10127deb hard_usetime 28800

## End of file  -  This is a gratuitous comment
```

EXAMPLE 11 Adding SAs for IPv6 Addresses

The following commands from the interactive-mode create an SA to protect IPv6 traffic between the site-local addresses

```
example # ipseckey
ipseckey> add esp spi 0x6789 src6 fec0:bbbb::4483 dst6 fec0:bbbb::7843\
           authalg md5 authkey bde359723576fdea08e56cbe876e24ad \
          encralg des encrkey be02938e7def2839 hard_usetime 28800
ipseckey>exit
```

EXAMPLE 12 Linking Two SAs as a Pair

The following command links two SAs together, as a pair:

```
example# ipseckey update esp spi 0x123456 dst 192.168.99.2 \
pair-spi 0x654321
```

Files

`/etc/inet/secret/ipseckeys`
Default configuration file used at boot time. See "Service Management Facility" and `SECURITY` for more information.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os
Interface Stability	Committed

See Also

`ps(1)`, `svcprop(1)`, `svcs(1)`, `ipsecconf(1M)`, `ipsecalgs(1M)`, `route(1M)`, `svcadm(1M)`, `svccfg(1M)`, `ike.config(4)`, `attributes(5)`, `smf(5)`, `ipsec(7P)`, `ipsecah(7P)`, `ipsecesp(7P)`, `pf_key(7P)`

Schneier, B., *Applied Cryptography: Protocols, Algorithms, and Source Code in C*. Second ed. New York, New York: John Wiley & Sons, 1996.

Diagnostics

The `ipseckey` command parses the configuration file and reports any errors. In the case of multiple errors, `ipseckey` reports as many of these as possible.

The `ipseckey` command does not attempt to use a `COMMAND` that has a syntax error. A `COMMAND` might be syntactically correct but can nevertheless generate an error because the kernel rejected the request made to `pf_key(7P)`. This might occur because a key had an invalid length or because an unsupported algorithm was specified.

If there are any errors in the configuration file, ipseckey reports the number of valid COMMANDS and the total number of COMMANDS parsed.

`Parse error on line` *N*.
If an interactive use of `ipseckey` would print usage information, this would print instead. Usually proceeded by another diagnostic. Because `COMMANDS` can cover more than a single line in the configuration file by using the backslash character to delimit lines, its not always possible to pinpoint in the configuration file the exact line that caused the error.

`Unexpected end of command line.`
An additional argument was expected on the command line.

Unknown
A value for a specific extension was unknown.

`Address type` *N* `not supported.`
A name-to-address lookup returned an unsupported address family.

N `is not a bit specifier`
`bit length` *N* `is too big for`
`string is not a hex string`
Keying material was not entered appropriately.

`Can only specify single`
A duplicate extension was entered.

`Don't use extension for` <*string*> `for` <*command*>.
An extension not used by a command was used.

`One of the entered values is incorrect: Diagnostic code` *NN*: <*msg*>
This is a general invalid parameter error. The diagnostic code and message provides more detail about what precise value was incorrect and why.

Notes In spite of its IPsec-specific name, `ipseckey` is analogous to `route(1M)`, in that it is a command-line interface to a socket-based administration engine, in this case, `PF_KEY`. `PF_KEY` was originally developed at the United States Naval Research Laboratory.

To have machines communicate securely with manual keying, SAs need to be added by all communicating parties. If two nodes wish to communicate securely, both nodes need the appropriate SAs added.

In the future `ipseckey` may be invoked under additional names as other security protocols become available to `PF_KEY`.

This command requires `sys_ip_config` privilege to operate and thus can run in the global zone and in exclusive-IP zones. The global zone can set up security associations with `ipseckey` to protect traffic for shared-IP zones on the system.

Name iscsiadm – enable management of iSCSI initiators

Synopsis `iscsiadm` *subcommand direct-object* [*options*] [*operand*]

Description The `iscsiadm` command enables management of the iSCSI (Internet SCSI) initiator on a host. `iscsiadm` is implemented as a set of subcommands, many with their own options, which are described in the section for that subcommand. Options not associated with a particular subcommand are described under OPTIONS.

`iscsiadm` works only when the following service is online:

```
svc:/network/iscsi/initiator:default
```

The `iscsiadm` command supports the following subcommands, which are described in detail in subsections that follow:

`add` Adds element(s) to an object.

`list` Lists element(s) of an object.

`modify` Modifies attributes of an object.

`remove` Removes an element from an object.

The `iscsiadm` subcommands operate on a *direct-object*. These are described in the section for each subcommand.

The `iscsiadm` command supports the Internet Storage Name Service (iSNS) for the discovery of iSCSI targets. The command supports the Challenge Handshake Authentication Protocol (CHAP) for authentication.

`add` Subcommand The syntax for the `add` subcommand is:

```
# iscsiadm add direct_object [operands...]
```

The `add` subcommand adds the following *direct_object*s:

`discovery-address` *discovery-address* [...]
Adds a target to a list of discovery addresses. A discovery address (as in the syntax shown below) is an IP *address*:*port* combination used in a SendTargets discovery session. Using this discovery approach, a target device can inform an initiator of the target address and target name of each target exposed by that device. Connection to a target is not attempted unless the SendTargets method of discovery has been enabled on the host. You enable this method with the `modify` subcommand.

The *discovery-address* parameter is formatted as:

<*IP address*>[:*port*]

If *port* is not specified, the default of 3260 will be used.

`isns-server` *isns-server* [...]
Add an iSNS server to the list of iSNS server addresses. An iSNS server address (specified in the syntax shown below) is an IP address-port combination used in an iSNS discovery session. By using iSNS discovery, an iSNS server can provide an initiator with information about a portal and the name of each target that belongs to the same discovery domain as that of the initiator. Connection to the iSNS server is not attempted unless the iSNS method of discovery has been enabled on the host. You enable this method with the `modify` subcommand, described below.

The *isns-server* parameter is formatted as:

IP_address`[:`*port*`]`

If a port is not specified, the default of 3205 is used.

`static-config` *static_target* [...]
Adds a target to the list of statically configured targets. A connection to the target will not be attempted unless the static configuration method of discovery has been enabled.

The *static_target* parameter is formatted as:

<*target-name*>`,`<*target address*>`[:`*port-number*`][,`*tpgt*`]`

<*target-name*> can be up to 223 characters.

`list` Subcommand The syntax for the `list` subcommand is:

```
# iscsiadm list direct-object [options]
```

The `list` subcommand displays data for the following *direct-objects*:

`discovery`
Lists the discovery methods and their current activation state, enabled or disabled. Discovery methods are:

- iSNS (Internet Storage Name Service)
- Static
- SendTargets

`initiator-node`
Lists information for the initiator node on the host. The iSCSI initiator node represents a logical HBA and is a logical host connection point for iSCSI targets. The parameter values listed in the response are default parameter settings for the initiator. Each connected target for an initiator can have parameter values that differ from the parameter values on the initiator node.

`static-config` [*static_target*[, ...]]
Lists the target name and address for specified targets or, if no static targets are specified, all statically discovered targets.

`target` [`-S`] [`-v`] [*target*[, ...]]
Lists a target's current parameters, connection state, and which method was used for the target's discovery. Reports information for specified targets or, if no targets are specified, all targets that have been discovered or have had parameters modified by the `modify target` subcommand.

When used with the `-S` option for a specified target, this subcommand returns:

- target name
- logical unit number
- vendor ID
- product ID
- OS device name (for example, `/dev/rdsk/c0t2d0s0`)

The `-v` options gives more details, such as the current login parameters, the detailed connection information, and the discovery method used to discover the target.

A return of `NA` as the discovery method parameter indicates that the target was created with a `iscsiadm modify target-param` command and does not exist as a discovered object. To remove such targets, use `iscsiadm remove target-param`.

`target-param` [`-v`] *target* [...]
Lists a target's default and user-defined parameters.

`discovery-address` [`-v`] [*discovery-address*[, ...]]
Lists the `discovery-address` objects that have been added using the `iscsiadm add discovery-address` subcommand.

When used with the `-v` option, lists all known targets at a specified *discovery-address*. The `-v` option returns one or more target names along with zero or more target addresses and associated target portal group tags (TPGT), if applicable.

`isns-server` [`-v`] [*isns-server*[, ...]]
Lists the `isns-server` objects that have been added using the `iscsiadm add isns-server` subcommand.

When used with the `-v` option, this subcommand lists all known targets at a specified `isns-server` address. The `-v` option returns one of more target names along with zero or more target addresses and associated target portal group tags, if applicable.

`modify` Subcommand The syntax for the `modify` subcommand is:

```
# iscsiadm modify direct_object [options]
```

The `modify` subcommand supports the following *direct_objects*:

`discovery` [*options*]
Enabling a discovery method initiates a discovery using that method. Disabling a discovery method that is currently enabled does not affect connections to any targets that have already been discovered by that method.

Options for `modify discovery` are as follows:

`-i`, `-iSNS enable` | `disable`
Enable or disable iSNS discovery.

`-s`, `--static enable` | `disable`
Enable or disable static discovery.

`-t`, `--sendtargets enable` | `disable`
Enable or disable SendTargets discovery.

`initiator-node` [*options*]
Modifies an initiator's properties. If a target is currently connected, this operation can succeed. However, the modified set of parameters will not be in effect for that target until an existing connection session no longer exists and a new connection has been established. The options `-C` and `--CHAP-secret` require a CHAP secret entry in response to a prompt.

For iSCSI booting when the Solaris I/O multipathing feature (formerly known as Sun StorEdge Traffic Manager [STMS] or MPxIO) is disabled, you can modify only the following initiator-node options:

- `-r`, `--radius-server`
- `-R`, `--radius-access`
- `-P`, `--radius-shared-secret`

For iSCSI booting when the Solaris I/O multipathing feature is enabled, you can modify only the following initiator-node options:

- `-h`, `--headerdigest`
- `-d`, `--datadigest`
- `-c`, `--configured-sessions`

Options for `modify initiator-node` are as follows:

`-A`, `--node-alias` <*initiator node alias*>
Modifies the initiator node alias. Maximum length of 223 characters.

`-a`, `--authentication chap` | `none`
Sets the authentication mode.

`-C`, `--CHAP-secret`
Sets the CHAP secret value. There is no default value. Maximum length is 16 characters; minimum required length is 12 characters.

`-c`, `--configured-sessions` <*num_sessions*> | <*IP address*>[,<*IP address*>...]
Sets the number of configured iSCSI sessions that will be created for each iSCSI target. The feature should be used in combination with the Solaris I/O multipathing feature described in `scsi_vhci(7D)`.

`-d`, `--datadigest none` | `CRC32`
Sets whether CRC32 is enabled to check SCSI data transfers.

`-H`, `--CHAP-name` *CHAP name*
Specifies a CHAP username. If you do not use this option, upon initialization, the CHAP name is set to the initiator node name. When the authentication method is set to CHAP (see `-a`/`--authentication` option, above), the CHAP username is displayed with the command `iscsiadm list initiator-node`.

`-h`, `--headerdigest` `none` | `CRC32`
Sets whether CRC32 is enabled to check SCSI packet headers.

`-m`, `--max-connections` *number_connections*
Modifies the maximum connection number for iSCSI sessions. The default value is `1`. The maximum number of connections for each session is `65535`.

`-N`, `--node-name` <*initiator node name*>
Modifies the initiator node name. Maximum of 223 characters.

Note – During Solaris installation, the initiator node name is set to a globally unique value. Changing this value can adversely affect operation within the iSCSI network.

`-P`, `--radius-shared-secret` (exclusive)
Sets the RADIUS shared secret.

`-R`, `--radius-access` `enable` | `disable`
Sets whether a RADIUS server will be used.

`-r`, `--radius-server` <*IP address*>[:<*port*>]
Sets the IP address and port of the radius server to be used.

`-T`, `--tunable-param` <<*tunable-prop*>=<*value*>, ...>
Specify one or more tunable parameters for all targets that initiator node connected.

Note – These values should only be modified by an administrator with a good working knowledge of the parameter's impact within the iSCSI network.

Supported tunable-prop options are:

`recv-login-rsp-timeout`
Session Login Response Time

The `recv-login-rsp-timeout` option specifies how long iSCSI initiator will wait for the response of iSCSI session login request from the iSCSI target. Valid value is from 0 to 60*60, default to 60 seconds.

`conn-login-max`
Maximized Connection Retry Time

The `conn-login-max` option lets the iSCSI initiator reestablish the connection to the target in case of IO timeout or connection failure during the given time window. Valid value is from 0 to 60*60, default to 180 seconds.

`polling-login-delay`
Login Retry Time Interval

The `polling-login-delay` option specifies the time interval between each login retry when iSCSI initiator to target IO timeout or connection failure. Valid value is from 0 to 60*60, default to 60 seconds.

`target-param` [*options*] *target*
Modifies a target's parameters. If a target is currently connected, the modify operation will succeed, although the modified settings might not take effect for a few seconds. To confirm that these settings are active, use `iscsiadm list target -v`. If a specified target is not associated with any discovery method, a target object is created with the specified parameters. After using this command to modify a target's parameters, the new parameters will persist until they are modified or removed with a `iscsiadm remove target-param` command on that target. The options `-C` and `--CHAP-secret` require a CHAP secret entry in response to a prompt.

Options for `modify target-param` are as follows:

`-B, --bi-directional-authentication enable | disable`
Sets the bidirectional option. If set to `enable`, the initiator performs bidirectional authentication for the specified target.

`-C, --CHAP-secret`
Sets the target's CHAP secret value. There is no default value. Maximum acceptable length is 16 characters.

`-c, --configured-sessions` <*num_sessions*> | <*IP address*>[,<*IP address*>...]
Sets the number of configured iSCSI sessions that will be created for each iSCSI target. The feature should be used in combination with the Solaris I/O multipathing feature described in `scsi_vhci(7D)`.

`-d, --datadigest none` | `CRC32`
Sets whether CRC32 is enabled or disabled for the data.

`-H, --CHAP-name` *CHAP name*
Sets a CHAP username. If you do not use this option, upon initialization, the CHAP name is set to the target name. When the authentication method is set to CHAP (see `-a`/`--authentication` option, under the `initiator-node` direct object, above), the CHAP username is displayed with the command `iscsiadm list initiator-node`.

`-h, --headerdigest none` | `CRC32`
Sets whether CRC32 is enabled or disabled for the header.

`-p, --login-param`
Specify one or more login parameter settings.

Note – These values should only be modified by an administrator with a good working knowledge of the parameter's impact within the iSCSI network.

The login parameters are derived from iSCSI proposed standard RFC 3720. Valid values are:

`dataseqinorder`	`yes` or `no`
`defaulttime2retain`	0–3600
`defaulttime2wait`	0–3600
`firstburstlength`	512 to 2^{24-1}
`immediatedata`	`yes` or `no`
`initialr2t`	`yes` or `no`
`maxburstlength`	512 to 2^{24-1}
`datapduinorder`	`yes` or `no`
`maxoutstandingr2t`	1 to 65535
`maxrecvdataseglen`	512 to 2^{24-1}

`-T`, `--tunable-param` <<*tunable-prop*>=<*value*>, ...>
Specify one or more tunable parameters for all targets that initiator node connected.

Note – Tunable values should only be modified by an administrator with a good working knowledge of the parameter's impact within the iSCSI network.

Supported *tunable-prop* options are:

`recv-login-rsp-timeout`
Session Login Response Time

The `recv-login-rsp-timeout` option specifies how long iSCSI initiator will wait for the response of iSCSI session login request from the iSCSI target. Valid value is from 0 to 60*60, default to 60 seconds.

`conn-login-max`
Maximized Connection Retry Time

The `conn-login-max` option lets the iSCSI initiator reestablish the connection to the target in case of IO timeout or connection failure during the given time window. Valid value is from 0 to 60*60, default to 180 seconds.

`polling-login-delay`
Login Retry Time Interval

The `polling-login-delay` option specifies the time interval between each login retry when iSCSI initiator to target IO timeout or connection failure. Valid value is from 0 to 60*60, default to 60 seconds.

`remove` Subcommand

The syntax for the `remove` subcommand is:

```
# iscsiadm remove direct_object
```

The `remove` subcommand supports the following *direct_object*s:

`discovery-address` *discovery-address*, ...
Removes a target device from the list of discovery addresses. A discovery address (as in the syntax shown below) is an IP address-port combination used in a SendTargets discovery session. Using this discovery approach, a target device can inform an initiator of the target address and target name of each target exposed by that device. If any target exposed by the discovery address is currently mounted or there is active I/O on the device, an error of "logical unit in use" is returned and the operation fails. If the associated devices are not in use, they are removed.

discovery-address must be formatted as:

<*IP address*>`[:`<*port*>`]`

There are no options associated with this direct object.

`isns-server` *isns-server*, ...
Removes an iSNS server from the list of iSNS server addresses. An iSNS server address (specified in the syntax shown below) is an IP address-port combination used in an iSNS discovery session. By using iSNS discovery, an iSNS server can provide an initiator with information about a portal and the name of each target that belongs to the same discovery domain as that of the initiator. If any target discovered by means of iSNS is currently mounted or there is active I/O on the device, an error of "logical unit in use" is returned and the operation fails. If the associated devices are not in use, they are removed.

isns-server must be formatted as:

IP_address`[:`*port*`]`

There are no options associated with this direct object.

`static-config` *static_target*, ...
Removes a target from the list of statically discovered targets. If the target being removed is currently mounted or there is active I/O on the device, an error of "logical unit in use" is returned and the operation fails. If a device is not in use, it will be removed.

static_target must be formatted as:

<*target-name*>`,`<*target-address*>`[:`*port-number*`][,`*tpgt*`]`

There are no options associated with this direct object.

`target-param` *target-name*
Removes target specified by *target-name*. The target name is formatted as:

<*target-name*>

There are no options associated with this direct object. For iSCSI booting when the Solaris I/O multipathing feature (formerly known as Sun StorEdge Traffic Manager [STMS] or MPxIO) is enabled, you cannot remove the target.

Proper Use of Discovery Methods

Do not configure a target to be discovered by both static and dynamic discovery methods. The consequence of using redundant discovery methods might be slow performance when communicating with the iSCSI target device.

Options

The following generic options are supported:

`-V`, `--version` Displays version information. Stops interpretation of subsequent arguments.

`-?`, `--help` Displays help information. Can be used following an `iscsiadm` command with no arguments, following a subcommand, or following a subcommand-direct object combination. Responds with help information appropriate for your entry. For example, if you enter:

```
# iscsiadm modify initiator-node --help
```

...`iscsiadm` responds with a display of the options available for that combination of subcommand and direct object.

Examples

EXAMPLE 1 Adding a Discovery Address

The following command uses the `add` subcommand to add a discovery address.

```
# iscsiadm add discovery-address 10.0.0.1:3260 10.0.0.2:3260
```

EXAMPLE 2 Adding a Static Target

The following command uses the `add` subcommand to add a static target.

```
# iscsiadm add static-config \
iqn.1999-08.com.array:sn.01234567,10.0.0.1:3260
```

EXAMPLE 3 Listing Current Discovery Settings

The following command uses the `list` subcommand to list current discovery settings.

```
# iscsiadm list discovery
        Discovery:
                Static: enabled
                Send Targets: disabled
                iSNS: enabled
```

EXAMPLE 4 Obtaining Verbose Discovery Output

The following commands uses the `-v` option (one with, one without) with the `list` subcommand to obtain verbose output.

EXAMPLE 4 Obtaining Verbose Discovery Output *(Continued)*

```
# iscsiadm list discovery-address
        Discovery Address: 10.0.0.1:3260
        Discovery Address: 10.0.0.2:3260

# iscsiadm list discovery-address -v 10.0.0.1:3260
        Discovery Address: 10.0.0.1:3260
                Target name: eui.210000203787d1f7
                        Target address:    10.0.0.1:3260
                Target name: eui.210000203787a693
                        Target address:    10.0.0.1:3260
```

EXAMPLE 5 Displaying Information on the Initiator

The following command uses the `list` subcommand to display information on the initiator.

```
# iscsiadm list initiator-node
Initiator node name: iqn.1986-03.com.company.central.interopv20-1
Initiator node alias: interopv20-1
        Login Parameters (Default/Configured):
                Header Digest: NONE/NONE
                Data Digest: NONE/NONE
        Authentication Type: CHAP
                CHAP Name: iqn.1986-03.com.company.central.interopv20-1
        RADIUS Server: NONE
        RADIUS access: disabled
        Tunable Parameters (Default/Configured):
                Session Login Response Time: 60/-
                Maximum Connection Retry Time: 180/-
                Login Retry Time Interval: 60/-
        Configured Sessions: 1
```

EXAMPLE 6 Displaying Static Configuration Information

The following command uses the `list` subcommand to display information about static configurations.

```
# iscsiadm list static-config
        Static target: eui.210000203787a693,10.0.0.1:3260
```

EXAMPLE 7 Displaying Target Information

The following commands show the use of the `list` subcommand with various options to display information about targets.

```
# iscsiadm list target
Target: iqn.2004-05.com.abcStorage:Tgt-1
        Alias: -
```

EXAMPLE 7 Displaying Target Information *(Continued)*

```
        TPGT: 12288
        ISID: 4000002a0000
        Connections: 1# iscsiadm list target -v iqn.2004-05.com.abcStorage:Tgt-1
Target: iqn.2004-05.com.abcStorage:Tgt-1
        Alias: -
        TPGT: 12288
        ISID: 4000002a0000
        Connections: 1
                CID: 0
                  IP address (Local): 10.4.52.158:32803
                  IP address (Peer): 10.4.49.70:3260
                  Discovery Method: SendTargets
                  Login Parameters (Negotiated):
                        Data Sequence In Order: yes
                        Data PDU In Order: yes
                        Default Time To Retain: 20
                        Default Time To Wait: 2
                        Error Recovery Level: 0
                        First Burst Length: 65536
                        Immediate Data: yes
                        Initial Ready To Transfer (R2T): yes
                        Max Burst Length: 262144
                        Max Outstanding R2T: 1
                        Max Receive Data Segment Length: 65536
                        Max Connections: 1
                        Header Digest: NONE
                        Data Digest: NONE
# iscsiadm list target -S iqn.2004-05.com.abcStorage:Tgt-1
Target: iqn.2004-05.com.abcStorage:Tgt-1
        Alias: -
        TPGT: 12288
        ISID: 4000002a0000
        Connections: 1
        LUN: 6
             Vendor:  ABCStorage
             Product: iSCSI Target
             OS Device Name: /dev/rdsk/c3t1d0s2
        LUN: 5
             Vendor:  ABCStorage
             Product: iSCSI Target
             OS Device Name: /dev/rdsk/c3t0d0s2
```

EXAMPLE 8 Displaying Target Parameter Information

The following command uses the `list` subcommand to display target information for a specific target.

EXAMPLE 8 Displaying Target Parameter Information *(Continued)*

```
# iscsiadm list target-param -v iqn.2004-05.com.abcStorage:Tgt-1
Target: iqn.2004-05.com.abcStorage:Tgt-1
        Alias: -
        Bi-directional Authentication: disabled
        Authentication Type: NONE
        Login Parameters (Default/Configured):
                Data Sequence In Order: yes/-
                Data PDU In Order: yes/-
                Default Time To Retain: 20/-
                Default Time To Wait: 2/-
                Error Recovery Level: 0/-
                First Burst Length: 65536/-
                Immediate Data: yes/-
                Initial Ready To Transfer (R2T): yes/-
                Max Burst Length: 262144/-
                Max Outstanding R2T: 1/-
                Max Receive Data Segment Length: 65536/-
                Max Connections: 1/-
                Header Digest: NONE/-
                Data Digest: NONE/-
        Tunable Parameters (Default/Configured):
                Session Login Response Time: 60/-
                Maximum Connection Retry Time: 180/-
                Login Retry Time Interval: 60/-
        Configured Sessions: 1
```

EXAMPLE 9 Enabling Static Discovery Method

The following command uses the `modify` subcommand to enable the static discovery method.

```
# iscsiadm modify discovery --static enable
```

EXAMPLE 10 Setting the IP Address for the Radius Server

The following command uses the `modify` subcommand to set the IP address for the radius server, which will be used for CHAP authentication.

```
# iscsiadm modify initiator --radius-server 10.0.0.1
```

EXAMPLE 11 Setting the Node Name for Initiator

The following command uses the `modify` subcommand to set the node name for the initiator node.

```
# iscsiadm modify initiator-node -N iqn.2004-10.com.SUN.host-1
```

EXAMPLE 12 Setting Max Connections for the Initiator Node

The following command uses the `modify` subcommand to set the max connection number for the initiator node. This enables multiple connections to exist in one session.

```
# iscsiadm modify initiator-node -m 3
```

EXAMPLE 13 Changing Target Parameters

The following command uses the `modify` subcommand to change the max connection number of the target parameters for a specified target.

```
# iscsiadm modify target-param -m 3 eui.210000203787a693
```

EXAMPLE 14 Removing a Discovery Address

The following command uses the `remove` subcommand to remove a discovery address.

```
# iscsiadm remove discovery-address 10.0.0.1:3260
```

EXAMPLE 15 Removing Target Parameters

The following command uses the `remove` subcommand to remove a set of target parameters.

```
# iscsiadm remove target-param eui.210000203787a693
```

EXAMPLE 16 Modifying Maximum Connection Number

The following command modifies the maximum number of connections for each session in the initiator's property. The modified value will be used in all sessions, to all targets.

```
# iscsiadm modify initiator-node --max-connections 4
```

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/storage/iscsi/iscsi-initiator
Interface Stability	Committed

See Also

`attributes(5)`, `iscsi(7D)`, `scsi_vhci(7D)`

Oracle Solaris 11.1 Administration: Devices and File Systems

Name isns – Internet Storage Name Service

Synopsis `/usr/sbin/isns`

Description The Internet Storage Name Service (iSNS) provides consolidated discovery services for Internet SCSI (iSCSI) and Internet Fibre Channel Protocol (iFCP) devices in an IP network. iSNS uses a client/server mechanism: servers store configuration information for clients, and provide that information upon a client's request. IETF RFC 4171 describes the protocols between the server and client.

This man page provides a summary of the Solaris iSNS server implementation. The current implementation does not support iFCP devices.

Solaris iSNS server is implemented as the daemon `isns`, which binds to the well –known port 3205 to service client requests. The daemon is started by the service management facility (`smf(5)`), using the fault management resource identifier (FMRI):

```
svc:/network/isns_server
```

Use `svcadm(1M)` to enable `isns`. Enabling the service means that it starts and runs automatically whenever the operating system is booted. The state of service can be displayed with the `svcs(1)` command.

The service properties listed below can be managed using `svccfg(1M)`. The default value is assigned per RFC 4171 and implementation choice.

`data_store_location`
Configuration data store location. The default location is `/etc/isns/isnsdata.xml`.

`ESI_retry_threshold_count`
Entity Status Inquiry retry threshold counter. The default count is `3`.

`Management_SCNs_Enabled`
Boolean that determines whether Management State Change Notification is enabled. The default is `yes`.

`Authorized_Control_Nodes`
Control node names.

After changing a property value, you must use `svcadm(1M)` `refresh` to enable `isns` to recognize the new value. If you change the `data_store_location` property, you must enter a `svcadm restart` command for the change to take effect.

RFC 4171 defines the default discovery domain, the default domain set, and the “Default DD/DDS” setting with the intent of managing clients that have not been assigned to any user-defined discovery domain. The server adopts the following behaviors with respect to the default discovery domain and domain set:

- An unassigned client is added to the default discovery domain. A newly registered client or a client that was removed from its last discovery domain membership is considered to be an unassigned client.

- When a client gets assigned to a user-defined discovery domain, the server will remove the client from the default discovery domain.
- The default discovery domain set is allowed to be administratively activated or deactivated in order to let the administrator control discovery among clients in the default discovery domain.
- It is not allowed to administratively add a client to the default discovery domain, nor to administratively add a user-defined discovery domain to the default discovery domain set.
- The default state of the Default discovery domain set is inactive.

The `isns` server supports certain `rbac(5)` authorizations that allow you to administer `isns` activity. These authorizations include the following `auth_attr(4)` privileges:

`solaris.isnsmgr.write`
: Required to create a discovery domain or domain set, to enable/disable a discovery domain set and to change grouping of iSNS clients in a discovery domain or grouping of discovery domains in a discovery domain set.

`solaris.smf.manage.isns`
: Required to manage the `isns` server through the `smf(5)`.

`solaris.smf.value.isns`
: Required to change the SMF service properties associated with `isns`.

The iSNS Server Management profile (see `prof_attr(4)`) includes all of the preceding authorizations. See `rbac(5)` for an overview of roles and authorizations.

Options

There are no options supported by the `isns` daemon.

Examples

EXAMPLE 1 Starting an `isns` Server

The following command starts the `isns` server.

```
# svcadm enable svc:/network/isns_server
```

EXAMPLE 2 Stopping an `isns` Server

The following command stops the `isns` server.

```
# svcadm disable svc:/network/isns_server
```

EXAMPLE 3 Changing an `isns` Property

The following sequence of commands changes the value of the `ESI_retry_threshold_count` property.

```
# svccfg -s svc:/network/isns_server setprop \
config/ESI_retry_threshold_count = 6
# svcadm refresh svc:/network/isns_server
```

Files `/usr/sbin/isns`
iSNS daemon binary.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	service/storage/isns
Interface Stability	Committed
Standard	See `standards(5)`.

See Also `svcs(1)`, `isnsadm(1M)`, `svcadm(1M)`, `svccfg(1M)`, `auth_attr(4)`, `prof_attr(4)`, `attributes(5)`, `rbac(5)`, `smf(5)`

Notes It is strongly recommended that you restart the server (`svcadm restart`) after a service property is changed. This allows the server to apply a uniform setting for existing and new clients.

A control node, as described in RFC 4171, is not required to administer the server. Control node operations can be achieved through the `isnsadm(1M)` command interface on the local host. For example, `isnsadm` enables you to create a discovery domain and a discovery domain set and to add a member to it, in order to create discovery domain and discovery domain set associations.

Name isnsadm – administer the internet Storage Name Server (iSNS) server

Synopsis

```
isnsadm options

isnsadm subcommand [subcommand_options] [operand]
```

Description The `isnsadm` command is the command-line interface to the Internet Storage Name Service (iSNS) server. `isnsadm` comprises a set of subcommands, described in their own section, each of which accomplishes one of the iSNS server management functions.

For any operations that will change the iSNS configurations the `solaris.isnsmgr.write` authorization is required. Refer to `isns(1M)`. For read operations, the command does not require special authorizations.

`isnsadm` has a set of general options and a set of subcommand-specific options. The first category is described under OPTIONS; the second category is described in the context of each subcommand description.

Options The following options are supported:

`-?`, `--help`
Displays context help. Stops interpretation of any subsequent arguments.

`-V`, `--version`
Displays version information. Stops interpretation of any subsequent arguments.

Operands The following operands are used by one or more `isnsadm` subcommands.

iscsi-node-name
iSCSI target or iSCSI initiator symbolic name. A string with a maximum length of 223 characters.

discovery-domain-name
Discovery domain symbolic name. A string with a maximum length of 256 characters.

discovery-domain-set-name
Discovery domain set symbolic name. A string with a maximum length of 256 characters.

Subcommands The `isnsadm` command supports the subcommands described below.

`add-dd` The `add-dd` subcommand adds a discovery domain to a discovery domain set.

The `add-dd` subcommand has the following syntax:

```
# isnsadm add-dd option discovery-domain-name, ...
```

`add-dd` has the following option:

`-s` *discovery-domain-set-name*
Specifies a discovery domain set.

`add-node` The `add-node` subcommand adds a node to a specified discovery-domain.

The `add-node` subcommand has the following syntax:

```
# isnsadm add-node option iscsi-node-name,...
```

`add-node` has the following options:

`-d`,`--dd` *discovery-domain-name*
Specifies a discovery domain.

`create-dd` The `create-dd` subcommand creates a discovery domain with the name you specify.

The `create-dd` subcommand has the following syntax:

```
# isnsadm create-dd discovery-domain-name, ...
```

`create-dd` has no options.

`create-dd-set` The `create-dd-set` subcommand creates a discovery domain set with the name you specify.

The `create-dd-set` subcommand has the following syntax:

```
# isnsadm create-dd-set discovery-domain-set-name, ...
```

`create-dd-set` has no options.

`delete-dd` The `delete-dd` subcommand deletes a discovery domain of the name you specify.

The `delete-dd` subcommand has the following syntax:

```
# isnsadm delete-dd discovery-domain-name, ...
```

`delete-dd` has no options.

`delete-dd-set` The `delete-dd-set` subcommand deletes a discovery domain set of the name you specify.

The `delete-dd-set` subcommand has the following syntax:

```
# isnsadm delete-dd-set discovery-domain-set-name, ...
```

`delete-dd-set` has no options.

`disable-dd-set` The `disable-dd-set` subcommand disables a discovery domain set.

The `disable-dd-set` subcommand has the following syntax:

```
# isnsadm disable-dd-set discovery-domain-set-name, ...
```

`disable-dd-set` has no options.

`enable-dd-set` The `enable-dd-set` subcommand enables a discovery domain set.

The `enable-dd-set` subcommand has the following syntax:

 # isnsadm enable-dd-set *discovery-domain-set-name*, **...**

`enable-dd-set` has no options.

`list-dd` The `list-dd` subcommand displays information about discovery domains. If no operand is specified, it lists all discovery domains that currently exist on the iSNS server.

The `list-dd` subcommand has the following syntax:

 # isnsadm list-dd [*option***] [***discovery-domain-name*, **...]**

`list-dd` supports the following option:

`-v`, `--verbose`
Displays the member contents of the discovery domain(s).

`list-dd-set` The `list-dd-set` subcommand lists the discovery domain sets, both enabled and disabled, that exist on the iSNS server. Note that there is no `dd-set` registration. If no operand is specified, it lists all of the discovery domain sets.

The `list-dd-set` subcommand has the following syntax:

 # isnsadm list-dd-set [*option***] [***discovery-domain-set-name*, **...]**

`list-dd-set` supports the following option:

`-v`, `--verbose`
Shows all discovery domains within the discovery domain set.

`list-node` The `list-node` subcommand displays information about nodes that are currently registered with the iSNS server or that are not registered and belong to non-default discovery-domain(s). For the latter case, the node has its type field shown as `unknown`. If no operand is specified, `list-node` lists all nodes known by the iSNS server.

The `list-node` subcommand has the following syntax:

 # isnsadm list-node [*options***] [***iscsi-node-name*, **...]**

`list-node` supports the following options:

`-t`, `--target`
Filters the list to display only iSCSI target nodes.

`-i`, `--initiator`
Filters the list to display only iSCSI initiator nodes.

`-v`, `--verbose`
Displays details about a node. Without this option, only the name, alias, and type information are displayed.

`modify-dd` The `modify-dd` subcommand modifies an attribute of a specified discovery domain.

The `modify-dd` subcommand has the following syntax:

```
# isnsadm modify-dd option discovery-domain-name
```

`modify-dd` has the following option:

`-n` *discovery-domain-name*
Specifies the new name of a discovery domain to be applied to an existing discovery-domain.

`modify-dd-set` The `modify-dd-set` subcommand modifies a discovery domain set.

The `modify-dd-set` subcommand has the following syntax:

```
# isnsadm modify-dd-set option discovery-domain-set-name
```

`modify-dd-set` has the following option:

`-n` *discovery-domain-set-name*
Specifies the new name of a discovery domain set to be applied to an existing discovery-domain-set.

`remove-dd` The `remove-dd` subcommand removes the association with a specified discovery domain set.

The `remove-dd` subcommand has the following syntax:

```
# isnsadm remove-dd option discovery-domain-name, ...
```

`remove-dd` has the following option:

`-s` *discovery-domain-set-name*
Specifies the discovery domain set from which the discovery domain will be removed.

`remove-node` The `remove-node` subcommand removes a node.

The `remove-node` subcommand has the following syntax:

```
# isnsadm remove-node option iscsi-node-name, ...
```

`remove-node` has the following option:

`-d` *discovery-domain-name*
Specifies the discovery domain from which a node will be removed.

`show-config` The `show-config` subcommand displays the iSNS server administrative settings. Note that the setting can be modified by means of the service management facility (see `smf(5)`). Refer to `isns(1M)`.

The `show-config` subcommand has the following syntax:

```
# isnsadm show-config
```

`show-config` has no options.

Examples

EXAMPLE 1 Displaying Clients

The following use of the `list-node` subcommand displays clients.

```
# isnsadm list-node -v
iSCSI Name: iqn.1986-03.com.sun:01:000e0c9f10da.45173FEA.engr
        Alias: STK5320_NAS
        Type: Target
        Network Entity: SE5310
        Portal: 172.20.57.95:3260
                Portal Group: 1
        Portal: 172.20.56.95:3260
                Portal Group: 1
        DD Name: Default
iSCSI Name: iqn.1986-03.com.sun:01:000e0c9f10da.454F00A2.acct
        Alias:
        Type: Target
        Network Entity: SE5310
        Portal: 172.20.57.95:3260
                Portal Group: 1
        Portal: 172.20.56.95:3260
                Portal Group: 1
        DD Name: Default
iSCSI Name: iqn.1986-03.com.sun:01:e00000000000.46fd8e2b
        Alias: host-x2100
        Type: Initiator
        Network Entity: iqn.1986-03.com.sun:01:e00000000000.46fd8e2b
        Portal: 172.20.236.123:58530
                Portal Group: 1
        DD Name: Default
```

EXAMPLE 2 Displaying a Discovery Domain

The following use of the `list-dd` subcommand displays discovery domains.

```
# isnsadm list-dd -v
DD name: Default
        DD set(s): Default
    iSCSI Name: iqn.1986-03.com.sun:01:000e0c9f10da.45173FEA.engr
    iSCSI Name: iqn.1986-03.com.sun:01:000e0c9f10da.454F00A2.acct
        iSCSI name: iqn.1986-03.com.sun:01:e00000000000.46fd8e2b
DD name: acct-dd
DD name: engineering-dd
```

EXAMPLE 3 Adding a Node

The following use of the `add-node` subcommand adds a node to a discovery domain, creating a discovery domain membership.

EXAMPLE 3 Adding a Node *(Continued)*

```
# isnsadm add-node -d engineering-dd \
iqn.1986-03.com.sun:01:000e0c9f10da.454F00A2.engr
```

EXAMPLE 4 Removing a Node

The following use of the `remove-node` subcommand removes a node from a discovery domain, thereby removing a discovery domain membership.

```
# isnsadm remove-node -d acct-dd \
iqn.1986-03.com.sun:01:000e0c9f10da.454F00A2.acct
```

EXAMPLE 5 Creating a Discovery Domain Set

The following use of the `create-dd-set` subcommand creates a discovery domain set.

```
# isnsadm create-dd-set operation-dd-set
```

EXAMPLE 6 Displaying a Discovery Domain Set

The following use of the `list-dd-set` subcommand displays discovery domain sets.

```
# isnsadm list-dd-set -v
DD Set name: Default
        State: Disabled
        DD Name: Default
DD Set name: operation-dd-set
        State: Disabled
```

EXAMPLE 7 Adding a Discovery Domain

The following use of the `add-dd` subcommand adds a discovery domain to a discovery domain set.

```
# isnsadm add-dd -s operation-dd-set engineering-dd
```

EXAMPLE 8 Displaying a Discovery Domain Set

The following use of the `list-dd-set` displays the attributes of a discovery domain set.

```
# isnsadm list-dd-set
DD Set name: Default
        State: Disabled
        DD Name: Default
DD Set name: operation-dd-set
        State: Disabled
        DD Name: engineering-dd
```

EXAMPLE 9 Enabling a Discovery Domain Set

The following use of the `enable-dd-set` subcommand enables a discovery domain set.

```
# isnsadm enable-dd-set Default
```

EXAMPLE 10 Disabling a Discovery Domain Set

The following use of the `disable-dd-set` subcommand disables a discovery domain set.

```
# isnsadm disable-dd-set Default
```

EXAMPLE 11 Displaying Administrative Settings

The following use of the `show-config` subcommand displays current administrative settings.

```
# isnsadm show-config
        Data Store Location: /etc/isns/isnsdata.xml
        Entity Status Inquiry Non-Response Threshold: 3
        Management SCN Enabled: yes
        Authorized Control Node Names: -
```

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	service/storage/isns
Interface Stability	Volatile

See Also

`iscsiadm(1M)`, `isns(1M)`, `attributes(5)`, `smf(5)`

Notes

When a subcommand is invoked with multiple operands and there are failures on one or more, but not all, operands, `isnsadm` displays a generic message indicating partial failure, with list of failed operands. An error on a specific operand can be found by issuing the same subcommand on the failing operand.

Name itadm – administer iSCSI targets

Synopsis

```
itadm create-target [-a,--auth-method radius | chap | none | default]
     [-s,--chap-secret] [-S,--chap-secret-file path]
     [-u,--chap-user chap-user-name] [-n,--node-name target_node_name]
     [-l,--alias alias] [-t,--tpg tpg-name[,tpg-name]]

itadm modify-target [-a,--auth-method radius | chap | none | default]
     [-s,--chap-secret] [-S,--chap-secret-file path]
     [-u,--chap-user chap-user-name] [-n,--node-name new_target_node_name]
     [-l,--alias alias] [-t,--tpg tpg-name[,tpg-name]] target_node_name

itadm delete-target [-f,--force] target_node_name

itadm list-target [-v,--verbose] [target_node_name]

itadm create-tpg tpg_name IP-address[:port] [IP-address[:port]]...

itadm list-tpg [-v,--verbose] [tpg_name]

itadm delete-tpg  [-f,--force] tpg_name

itadm create-initiator [-s,--chap-secret] [-S,--chap-secret-file path]
     [-u,--chap-user chap-user-name] initiator_node_name

itadm modify-initiator [-s,--chap-secret] [-S,--chap-secret-file path]
     [-u,--chap-user chap-user-name] initiator_node_name

itadm list-initiator [-v,--verbose] initiator_node_name

itadm delete-initiator initiator_node_name

itadm modify-defaults [-a,--auth-method radius | chap | none]
     [-r,--radius-server IP-address[:port]] [-d,--radius-secret]
     [-D,--radius-secret-file path][-i,--isns enable | disable]
     [-I,--isns-server IP-address[:port][,IP-address[:port]]]

itadm list-defaults
```

Description The `itadm` command manages Internet SCSI (iSCSI) target nodes within the SCSI Target Mode Framework described in `stmfadm(1M)` and `libstmf(3LIB)`. This allows the iSCSI initiators to access STMF logical units using the iSCSI protocol. In addition to iSCSI target nodes, `itadm` manages two other classes of managed objects: iSCSI Target Portal Groups, and iSCSI Initiator Node Contexts.

`itadm` is implemented as a set of subcommands with options and operands for each subcommand. These subcommands are described in their own section, below. In addition to its subcommands, `itadm` has a help command, which displays the utility's usage information. The help command is invoked with the `-?` option.

iSCSI Target Portal Groups An iSCSI Target Network Portal is an IP address and TCP port that can be used by an initiator node to connect to an iSCSI target. A collection of these portals is called a Target Portal Group (TPG). You can use a TPG to limit access to an iSCSI target. Use the `itadm modify -t`

command to bind a specific iSCSI target to the TPG. An iSCSI listener is created on each IP address that belongs to the TPG, and listens for connections to the iSCSI target.

A TPG is identified by a unique name provided when the TPG is created. A numerical "Target Portal Group Tag" from the range 2-65535 is automatically generated when the TPG is created. The Target Portal Group Tag 1 is reserved for the "default" target portal group that is used when no explicit Target Portal Groups are set on the target. The portal for the default TPG matches requests from all network interfaces on port 3260.

iSCSI Initiator Node Contexts

Certain operations such as authentication by means of Challenge Handshake Authentication Protocol (CHAP) require parameters associated with a remote iSCSI Initiator Node. These parameters are associated with an iSCSI Initiator Node Context. An iSCSI Initiator Node Context is identified by its Initiator Node Name, formatted in either IQN or EUI format (see RFC 3720). For example:

```
iqn.1986-03.com.sun:01:e00000000000.47d55444
eui.02004567A425678D
```

Specifying IP Addresses

A number of `itadm` subcommands require that you specify one or more IP addresses with optional port numbers. For IPv4, use standard dotted decimal notation. For IPv6, enclose addresses in square brackets. The following are example specifications.

```
IPv4:  10.2.4.1
       10.2.4.1:3260
IPv6:  [1080:0:0:0:8:800:200C:417A]
       [1080:0:0:0:8:800:200C:417A]:3260
```

Sub-commands

The following are the `itadm` subcommands with their options.

itadm create-target

```
itadm create-target [-a,--auth-method radius | chap | none | default]
     [-s,--chap-secret]
     [-S,--chap-secret-file path] [-u,--chap-user chap-user-name]
     [-n,--node-name target_node_name] [-l,--alias alias]
     [-t,--tpg tpg-name[,tpg-name,...]]
```

Create a iSCSI target with the specified options. Options are as follows.

`-a,--auth-method radius | chap | none | default`
: Specifies the authentication method to use for the target. Valid values are `radius`, `chap`, and `none`. `chap` indicates that initiators connecting to this target must be authenticated using the Challenge Handshake Authentication Protocol (CHAP). `radius` indicates initiators should also be authenticated by means of CHAP but the required authentication parameters should be obtained from a central RADIUS server (see the `radius-server` and `radius-secret` options). `none` means that no authentication is required to connect to the target. `default` means the target will use the global setting of this property. (See the `modify-defaults` subcommand.)

`-s,--chap-secret`
: The CHAP secret to send during mutual CHAP authentication. There is no default for this property. Maximum length is 255 characters; minimum required length is 12 characters.

`-S,--chap-secret-file` *path*
Path to a temporary file containing the CHAP secret as described in the `-s` option.

`-u,--chap-user` *chap-user-name*
Specifies the CHAP username for a target for use in mutual CHAP authentication. This value is allowed only for targets, cannot be set globally, and is used only when the initiator node is configured to use mutual CHAP authentication. If no value is specified then the target node name is used as the username. See `iscsiadm(1M)`.

`-n,--node-name` *target_node_name*
An iSCSI Target Node is identified by its Target Node Name, formatted in either IQN or EUI format (see RFC 3720). This option establishes that name.

`-l,--alias` *alias*
An alternate identifier associated with a target node. The identifier does not need to be unique.

`-t,--tpg` *tpg-name*[,*tpg-name*,...]
A list of Target Portal Group (TPG) identifiers that specifies the TPGs that an initiator can use to access a specific target or the keyword `default`. If `default` is specified, the target will use the default portal, `INADDR_ANY:3260`.

`itadm modify-target`

```
itadm modify-target [-a,--auth-method radius | chap | none | default]
    [-s,--chap-secret] [-S,--chap-secret-file path]
    [-u,--chap-user chap-user-name] [-n,--node-name new_tgt_node_name]
    [-l,--alias alias] [-t,--tpg tpg-name[,tpg-name]] target_node_name
```

Modify an iSCSI target according to the specified options. Options are as follows.

`-a,--auth-method radius | chap | none | default`
As described under the `create-target` subcommand, above.

`-s,--chap-secret`
As described under the `create-target` subcommand, above.

`-S,--chap-secret-file` *path*
As described under the `create-target` subcommand, above.

`-u,--chap-user` *chap-user-name*
As described under the `create-target` subcommand, above. To remove an explicitly set CHAP username use `-u none`.

`-n,--node-name` *target_node_name*
Renames the target. See also the description of `-n` under the `create-target` subcommand, above.

`-l,--alias` *alias*
As described under the `create-target` subcommand, above. To remove an explicitly set alias use `-l none`.

`-t,--tpg` *tpg-name*[,*tpg-name*,...]
As described under the `create-target` subcommand, above.

`itadm list-target` `itadm list-target [-v,--verbose] [`*target_node_name*`]`

List information about the configured targets. If *target_node_name* is specified, list only the information for that target. Option is as follows.

`-v,--verbose`
Verbose mode.

`itadm delete-target` `itadm delete-target [-f,--force]` *target_node_name*

Delete the target specified by *target_node_name*. The target must be `offline` before it can be deleted. Option is as follows.

`-f,--force`
If the target persists in an online state, this option attempts to offline the target before deleting it.

`itadm create-tpg` `itadm create-tpg` *tpg_name IP-address*`[:`*port*`]...`

Create an iSCSI target portal group made up of the specified portals and assign it the identifier *tpg_name*. Each portal is an IP address and port pair. IPv4 portals are specified in dotted address notation, for example, `172.31.255.255`. IPv6 portal addresses must be enclosed in square brackets.

This subcommand has no options.

`itadm list-tpg` `itadm list-tpg [-v,--verbose] [`*tpg_name*`]`

List information about the configured target portal group. If *tpg_name* is specified then list only the information about the target portal group associated with that *tpg_name*. Option is as follows.

`-v,--verbose`
Verbose mode.

`itadm delete-tpg` `itadm delete-tpg [-f,--force]` *tpg_name*

Delete the target portal group associated with *tpg_name*. Option is as follows.

`-f,--force`
If the TPG is associated with any targets, the request to delete will be denied unless this option is specified.

`itadm create-initiator` `itadm create-initiator [-s,--chap-secret] [-S,--chap-secret-file` *path*`]`
`[-u,--chap-user` *chap-user-name*`]` *initiator_node_name*

Configure parameters associated with the remote initiator named *initiator_node_name*. Options are as follows.

`-s,--chap-secret`
As described under the `create-target` subcommand, above.

`-S,--chap-secret-file` *path*
As described under the `create-target` subcommand, above.

`-u,--chap-user` *chap-user-name*
Specifies the CHAP username for an initiator, for use in CHAP authentication. If no value is specified then the initiator node name is used as the username.

`itadm modify-initiator`

```
itadm modify-initiator [-s,--chap-secret] [-S,--chap-secret-file path]
    [-u,--chap-user chap-user-name] initiator_node_name
```

Modify parameters associated with the remote initiator named *initiator_node_name*. Options are as follows.

`-s,--chap-secret`
As described under the `create-target` subcommand, above.

`-S,--chap-secret-file` *path*
As described under the `create-target` subcommand, above.

`-u,--chap-user` *chap-user-name*
Specifies the CHAP username for an initiator, for use in CHAP authentication. If no value is specified then the initiator node name is used as the username.

`itadm delete-initiator`

```
itadm delete-initiator initiator_node_name
```

Delete parameters associated with the remote initiator named *initiator_node_name*. This subcommand has no options.

`itadm list-initiator`

```
itadm list-initiator [-v,--verbose] initiator_node_name
```

List parameters associated with the initiator named *initiator_node_name*. Option is as follows.

`-v,--verbose`
Verbose mode.

`itadm modify-defaults`

```
itadm modify-defaults [-a,--auth-method radius | chap | none]
    [-r,--radius-server IP-address[:port]] [-d,--radius-secret]
    [-D,--radius-secret-file path][-i,--isns enable | disable]
    [-I,--isns-server IP-address[:port][,IP-address[:port]]]
```

Modify default parameters. Options are as follows.

`-a,--auth-method radius | chap | none`
Specifies the default authentication method to use for all targets. Valid values are `radius`, `chap`, and `none`. `chap` indicates that initiators connecting to this target must be authenticated using Challenge Handshake Authentication Protocol (CHAP). `radius` indicates initiators should also be authenticated by means of CHAP, but the required authentication parameters should be obtained from a central RADIUS server. (See

`--radius-server` and `--radius-secret` options.) `none` means that no authentication is required to connect to the target. Individual targets can override this global setting using the `-a` option of the `create-target` and `modify-target` subcommands.

`-d,--radius-secret`
RADIUS Shared Secret for centralized CHAP authentication.

`-D,--radius-secret-file` *path*
Path to a temporary file containing the CHAP secret as described in the `-d` option.

`-i,--sns enable | disable`
Specifies whether targets should be registered with the set of defined iSCSI Name Service (iSNS) servers.

`-I,--isns-server` *IP-address*[:*port*][,*IP-address*[:*port*],...]
Defines a list of iSNS servers with which iSCSI target nodes will be registered when the `isns` option associated with the respective target is set. Up to eight iSNS servers can be specified. To remove all iSNS servers, use `-I none`.

`-r,--radius-server` *IP-address*[:*port*]
Specify the IP address of the RADIUS server used for centralized CHAP authentication.

`itadm list-defaults` `itadm list-defaults`

List information about the default properties. This subcommand has no options.

Examples

EXAMPLE 1 Creating a Target

The following command creates a target.

```
# itadm create-target
Target iqn.1986-03.com.sun:02:72e1b181-7bce-c0e6-851e-ec0d8cf14b7a
successfully created
```

EXAMPLE 2 Creating a Target with a Specific Name

The following command creates a target with a specific IQN.

```
# itadm create-target -n eui.20387ab8943ef7548
or:
# itadm create-target \
-n iqn.1986-03.com.sun:02:a9a366f8-cc2b-f291-840948c7f29e
```

EXAMPLE 3 Changing a Name

The following command changes an IQN for an existing target.

```
# itadm modify-target -n eui.20387ab8943ef7548 \
iqn.1986-03.com.sun:02:a9a366f8-909b-cc2b-f291-840948c7f29e
```

EXAMPLE 4 Setting up CHAP Authentication

The following command sets up CHAP authentication for a target using the default CHAP username.

```
# itadm modify-initiator -s iqn.1986-03.com.sun:01:e00000000000.47d55444
Enter CHAP secret: *********
Re-enter secret: *********

# itadm modify-target -a chap eui.20387ab8943ef7548
```

EXAMPLE 5 Creating Target Portal Groups

The following command creates two target portal groups, `A` and `B`, using port 8000 for the addresses in TPG 2.

```
# itadm create-tpg A 192.168.0.1 192.168.0.2
# itadm create-tpg B 192.168.0.2:8000 192.168.0.2:8000
```

EXAMPLE 6 Configuring a Target to Use TPGs

The following command configures a target to use TPGs A and B.

```
# itadm modify-target -t A,B eui.20387ab8943ef7548
```

EXAMPLE 7 Setting up RADIUS Authentication for Specific Target

The following command sets up RADIUS authentication for a specific target.

```
# itadm modify-defaults -r 192.168.10.1 -d
Enter RADIUS secret: *********
Re-enter secret: *********

# itadm modify-target -a radius eui.20387ab8943ef7548
```

EXAMPLE 8 Setting up RADIUS Authentication for All Targets

The following command sets up RADIUS authentication for all targets.

```
# itadm modify-defaults -d -r 192.168.10.1 -a radius
Enter RADIUS secret: *********
Re-enter secret: *********
```

The preceding command assumes all targets were created with `-a default`.

EXAMPLE 9 Listing Default Properties

The following command lists default properties.

```
# itadm list-defaults
iSCSI Target Default Properties:

alias:          none
```

EXAMPLE 9 Listing Default Properties *(Continued)*

```
auth:           none
radiusserver:   none
radiussecret:   unset
isns:           disabled
isnsserver:     2.3.4.5,4.5.6.7
```

EXAMPLE 10 Listing Targets

The following command lists targets.

```
# itadm list-target
TARGET NAME                                              STATE    SESSIONS
iqn.1986-03.com.sun:02:72e1b181-7bce-c0e6-851e-ec0d8cf14b7a  online   0
iqn.1986-03.com.sun:02:2cb0c526-c05a-e279-e396-a367006f4227  online   0
iqn.1986-03.com.sun:02:d14125bb-1c9d-c28d-97b0-f89259b642f3  online   0
iqn.1986-03.com.sun:02:03ff9fc5-794a-e9b4-a081-bb82917c292a  online   0
```

EXAMPLE 11 Listing Targets (Verbose)

The following command lists targets with the verbose option.

```
# itadm list-target -v
TARGET NAME                                              STATE    SESSIONS
iqn.1986-03.com.sun:02:d23e68d8-2d79-c988-98e7-a6361689d33c  online   0
       alias:                  -
       auth:                   none (defaults)
       targetchapuser:         -
       targetchapsecret:       unset
       tpg-tags:               default
iqn.1986-03.com.sun:02:94ec46d4-c8e1-6993-ef03-ffc1dcd66606  online   1
       alias:                  -
       auth:                   chap
       targetchapuser:         -
       targetchapsecret:       unset
       tpg-tags:               nge1_ipv4 = 3
```

EXAMPLE 12 Listing a Specific Target

The following command lists targets with the verbose option.

```
# itadm list-target -v \
iqn.1986-03.com.sun:02:2cb0c526-c05a-e279-e396-a367006f4227
TARGET NAME                                              STATE    SESSIONS
iqn.1986-03.com.sun:02:2cb0c526-c05a-e279-e396-a367006f4227  online   1
        alias:                  -
        auth:                   chap
        targetchapuser:         -
        targetchapsecret:       unset
        tpg-tags:               nge1_ipv4 = 3
```

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/storage/iscsi/iscsi-target
Interface Stability	Committed

See Also iscsiadm(1M), stmfadm(1M), libstmf(3LIB), attributes(5)

Name itu – convert packages to Driver Update format and patch Solaris install media for Install Time Update

Synopsis

```
itu makedu -r solaris_release [-v] [-f] [-d output_dir] [-o iso_file]
     [-l iso_label] package [package...]

itu updatemedia -R media_root [-v] [-f] [-o iso_file]
     [-l iso_label] pkg_or_patch [pkg_or_patch...]

itu makeiso -o iso_file [-v] [-l iso_label] media_root
```

Description The `itu` utility converts driver packages to Driver Update (DU) format and patches a Solaris install media with driver packages and patches for Install Time Update (ITU). `itu` has three subcommands: `makedu`, `updatemedia` and `makeiso`.

Options The following options are supported:

`-d` *output_dir*
Directory where the Driver Update directory is to be created.

`-f`
If *output_dir*/DU or *iso_file* already exists, remove it without asking first.

`-l` *iso_label*
Label/volume name of the ISO image (if `-o` option is specified).

`-o` *iso_file*
Path of the ISO image file to be created. For subcommands `updatemedia` and `makeiso`, it will be a bootable ISO image. This option must be specified for subcommand `makeiso`.

`-R` *media_root*
Top-level directory of on-disk image of Solaris installation media. This option must be specified for subcommand `updatemedia`.

`-r` *solaris_release*
Solaris release number for which the Driver Update is intended. It takes the form of the output of `uname -r`, for example, `5.10`. This option must be specified for subcommand `makedu`.

`-v`
Verbose. Multiple `-v` options increase verbosity.

Sub-commands The `itu` subcommands are described as follows.

`makedu` The `makedu` subcommand takes one or more driver packages as input and converts them to DU format. At the beginning of an interactive Solaris installation session, these driver updates can be applied to the running kernel, which will then also automatically apply them to the newly installed Solaris at the end of the installation process.

The `-r` option is required to specify the Solaris release number for which the driver updates apply. The *solaris_release* option argument takes the form `uname -r` output, for example, `5.10` or `5.11`.

If the `-d` option is specified, the resulting DU directory tree is placed in the directory *output_dir*.

If the `-o` option is specified, a (non-bootable) ISO image of the DU directory tree is written in the file *iso_file*. This ISO image can be burned onto a CD/DVD using `cdrw(1)` or `cdrecord(1)` (not a SunOS man page).

At least one of `-d` and `-o` option must be specified. If both are specified, then both an ISO image and a directory tree are generated.

`updatemedia` The `updatemedia` subcommand takes a list of driver packages and patches as input and applies them to the miniroot of a Solaris install media. It also places them in a subdirectory called `ITUs` under the Solaris install media's top-level directory:

media_root`/ITUs`

When booting a system from the updated media, the patches and packages will be part of the booted Solaris image. They will also be applied to the target system being installed at the end of the installation process.

The `-R` option must be entered on the command line to specify the Solaris install media. Note that the install media must be on a location that is writable by `itu`.

If the `-o` option is specified, a bootable ISO image of the patched install media is also created in the file *iso_file*. The ISO image can then be burned onto a CD or DVD.

`makeiso` The `makeiso` subcommand runs `mkisofs(8)` to create a bootable Solaris ISO image of the Solaris install media *media_root* and writes it to the file *iso_file*. The ISO image file can then be burned onto a CD or DVD with utilities such as `cdrw(1)` or `cdrecord(1)`. (Note that `mkisofs(8)` and `cdrecord(1)` are not SunOS man pages.)

Caution – The Solaris install media *media-root* must contain the file `boot/grub/stage2_eltorito`, which will be written to the media boot sectors. This file will be modified with some boot information, thus it has to be writable. If necessary, first save a copy, prior to running this subcommand.

Operands The following operands are supported:

package [*package*...]
One or more driver packages.

pkg_or_patch [*pkg_or_patch*...]
One or more patches or packages.

media_root
The top-level directory of a Solaris install media.

Examples

EXAMPLE 1 Creating a DU CD/DVD

The following commands create a Driver Update CD/DVD containing the packages `SAMPLEpkg1` and `SAMPLEpkg2`.

```
# itu makedu -r 5.10 -o my.iso SAMPLEpkg1 SAMPLEpkg2
# cdrw -i my.iso
```

EXAMPLE 2 Patching the Solaris Install Media

The following command patches the Solaris install media in `/export/s10u1` with patch `/opt/patches/123456-07` and driver package `/opt/pkgs/MYdriver`. The command also creates a bootable ISO image with ISO label "`MyS10U1`" in the file `/tmp/dvd.iso`.

```
# /usr/bin/itu updatemedia -R /export/s10u1 -o /tmp/dvd.iso -l MyS10U1 \
       /opt/patches/123456-07 /opt/pkgs/MYdriver
```

EXAMPLE 3 Creating a Bootable ISO Image

The following commands create the bootable ISO image `mydvd.iso` of the Solaris install image `/export/solaris-10u1` with ISO label "`Special-S10`".

```
# /usr/bin/itu makeiso -o mydvd.iso -l "Special-S10" \
     /export/solaris-10u1
# cdrw -i mydvd.iso
```

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os
Interface Stability	Committed

See Also

`cdrw(1)`, `pkgadd(1M)`, `attributes(5)`

`mkisofs(8)`, (`/usr/share/man/man8/mkisofs.8`), in the `SUNWfsman` package (not a SunOS man page)

Name js2ai – Translate JumpStart rules and profiles for use with the Automated Installer (AI).

Synopsis

```
js2ai [-h | --version]

js2ai -r | -p profile [-d jsdir]
    [-D destdir] [-lSv]

js2ai -s [-d jsdir]
    [-D destdir] [-Sv]

js2ai -V manifest
```

Description `js2ai` is a utility for converting Oracle Solaris 10 JumpStart `rules`, profile, and `syscfg` configuration files to a format compatible with Automated Installer (AI). This utility makes a "best effort" to translate those JumpStart keywords that can be translated to the AI context. While this conversion does not create a complete one-to-one equivalence with JumpStart, it does provide AI manifest and system configuration profile entries that can then be used as a template for creating a complete AI configuration setup based on information gathered from JumpStart configuration files.

Using `js2ai`, you can do the following:

- Process the `rules` file and the associated profiles in the current working directory.
- Process the `rules` file and the associated profiles in a specified directory.
- Process a specific profile or `sysidcfg` file.
- Direct the resulting output files to a specific directory. For more information on the `js2ai` output files, see the "Examples" and "Files" sections.

Translating Rule Keywords

TABLE 1 JumpStart Rule Keywords Translation

JumpStart Rule Keyword	AI Criteria Keyword
`arch`	`cpu`
`hostaddress`	`ipv4`
`karch`	`arch`
`memsize`	`mem`
`model`	`platform`
`network`	`ipv4`

JumpStart rule keywords not supported by `js2ai`:

```
any             installed
disksize        osname
domainname      probe
hostname        totaldisk
```

Converting Profile Keywords

TABLE 2 JumpStart Profile Keywords

JumpStart Profile Keyword	Notes
`boot_device`	The `rootdisk` is set to the specified device if not previously set by the `root_device` keyword.
`fdisk`	The value of `disk_name` must be a device. A device of `all` is not supported. The `fdisk` type must be `solaris`. A size of 0 or `delete` is not supported. If `partitioning` is `default` and the `rootdisk` has not been set, the first `fdisk solaris` partition encountered is used as the `rootdisk`.
`filesys`	The local and mirrored file systems are supported when the mount point specified is / or `swap`. No validation of the size is performed. The size specified in the resulting AI manifest might need to be adjusted to achieve a successful installation with this manifest.
`install_type`	Only the value `initial_install` is supported.
`locale`	No translation is performed. Make sure the locale specified is supported in Oracle Solaris 11.
`package`	An attempt to convert the specified package to its Oracle Solaris 11 equivalent is performed. Specifying the location of the package is not supported. Package lookups can take a considerable amount of time. If your profiles contain a long list of packages, you might want to use the `--local` flag during the conversion process.
`partitioning`	Supported types are `default` and `explicit`. Unlike JumpStart, when `partitioning default` is specified, only the disks that `js2ai` knows about are used. If no disks are specified in any keywords, the generated profile tells AI to choose which disk to use.
`pool`	If a pool is specified in a profile, the ZFS root pool is created using the specified devices. The `pool` keyword supersedes all other keywords when determining which devices to use for the ZFS root pool. No validation of the pool size, swap size, or dump size is performed. These sizes might need to be adjusted in the resulting AI manifest to achieve a successful installation with this manifest.
`root_device`	The `rootdisk` is set to the specified device.
`system_type`	Only the value `standalone` is supported.
`usedisk`	The specified device might be used to resolve the `any` or `rootdisk` device during the conversion. Any devices specified that are not used for this purpose are added to the ZFS root pool, when that pool is not mirrored.

JumpStart profile keywords not supported by `js2ai`:

```
archive_location        geo
backup_media            layout_constraint
bootenv                 local_customization
client_arch             metabd
client_root             no_master_check
client_swap             no_content_check
cluster                 num_clients
dontuse                 patch
forced_deployment
```

How the System's Root Disk is Determined During Profile Translation

Since `js2ai` does not have access to the actual system a profile references during the profile translation process, `js2ai` attempts to determine what the root disk is during translation using a process that matches JumpStart as much as possible.

The `js2ai` tool performs the following steps to determine what device to use for the root disk.

Stage	Action
1	If the `root_device` keyword is specified in the profile, `js2ai` sets `rootdisk` to the device on which the slice resides.
2	If `rootdisk` is not set and the `boot_device` keyword is specified in the profile, `js2ai` sets `rootdisk` to the boot device.
3	If `rootdisk` is not set, `partitioning default` is specified, and a `solaris fdisk` entry is encountered, `js2ai` sets `rootdisk` to the specified `disk_name`.
4	If `rootdisk` is not set and a `filesys` *cwtxdysz size* `/` entry is specified in the profile, `js2ai` sets `rootdisk` to the *cwtxdysz* disk specified in the entry.
5	If `rootdisk` is not set and a `usedisk` *disk* entry is specified in the profile, `js2ai` sets `rootdisk` to the *disk* disk specified in the entry.
6	If `rootdisk` is not set and the following specification is encountered in the profile where *size* is not 0 or `delete` and *disk* is not `all`, then `rootdisk` is set to this *disk* name. `fdisk` *disk* `solaris` *size*
7	If `rootdisk` is not set, any occurrence where the device is specified as `rootdisk` generates a conversion error.

How the any Device Is Translated During Profile Translation

The `js2ai` tool performs the following steps to determine what device to use when the `any` keyword is specified.

Stage	Action
1	If the any device is specified and the keyword action specified (non-mirrored pool, or filesys with a / mount point), the any device is set to rootdisk if rootdisk is set.
2	If the any device has not been translated and a usedisk statement exists in the profile, the any device is set to the device specified by the usedisk statement.
3	If the any device has not been translated and the action where the any device is specified causes the ZFS root pool to be created, AI chooses the device. This is not applicable when a mirrored pool is specified.

How the ZFS Root Pool is Determined During Profile Translation

The `js2ai` tool performs the following steps to determine what device to use for the ZFS root pool. Once the ZFS root pool is determined, subsequent definitions encountered are flagged as errors if they conflict with the ZFS root pool that has already been determined.

Stage	Action
1	If the profile specifies the `pool` keyword, `js2ai` sets the ZFS root pool to the devices specified by the `pool` keyword.
2	If the ZFS root pool has not been determined and the profile specifies a `filesys` with a mount point of /, the ZFS root pool is created using the devices specified.
3	If the ZFS root pool has not been determined and all keywords in the profile have been processed, and if `rootdisk` is set, the ZFS root pool is created using the `rootdisk` device.
4	If the ZFS root pool has not been determined and the partition type is `default`, AI chooses the device to use for the ZFS root pool.
5	If the ZFS root pool has not been determined and no errors have occurred during processing, AI chooses the device to use for the ZFS root pool.
6	If the ZFS root pool is not a mirrored pool and one or more `usedisk` devices that were specified have not been used for a `rootdisk` or any device translation, those disks are added to the ZFS root pool.

Converting `sysidcfg` Keywords

TABLE 3 JumpStart `sysidcfg` Keywords

`sysidcfg` Keyword	Notes
`keyboard`	No translation is performed. Make sure the keyboard specified in the `sysidcfg` file is supported in Oracle Solaris 11.
`name_service`	Supports values `None`, `DNS`, `NIS`, and `LDAP`. NIS+ name services are translated as NIS. If a name service is specified, the network interface in Oracle Solaris 11 is configured for `DefaultFixed`. The `network_interface` keyword can be used to define the characteristics of the network.

TABLE 3 JumpStart `sysidcfg` Keywords *(Continued)*

sysidcfg Keyword	Notes
`network_interface`	AI supports configuring only a single interface as part of system installation. Because of this limitation, the `js2ai` tool processes only the interface labeled `PRIMARY` or the first interface encountered in the `sysidcfg` file. If a `name_service` is specified, the network is configured as `DefaultFixed`. A properly configured `DefaultFixed` network needs to provide the host name, IP address, netmask, and gateway. Automated network configuration is only supported if no name service is specified.
`root_password`	No translation is necessary.
`security_policy`	Supports value: `None`
`service_profile`	Supports value: `limited_net`
`system_locale`	No translation is performed. Make sure the locale specified in the `sysidcfg` file is supported in Oracle Solaris 11.
`terminal`	No translation is performed. Make sure the terminal type specified in the `sysidcfg` file is supported in Oracle Solaris 11.
`timeserver`	Supports value: `localhost`
`timezone`	No translation is necessary.

JumpStart `sysidcfg` keywords not supported by `js2ai`:

`nfs4_domain`

Options The `js2ai` command has the following options. The use of these options is illustrated in the "Examples" section.

`-h`, `--help`
Show the usage help message.

`--version`
Show the version number of the `js2ai` utility.

`-d` *jsdir*, `--dir` *jsdir*
Specify the location of the `rules` and profile files or the `sysidcfg` file.

`-D` *destdir*, `--dest` *destdir*
Specify the location for the output files.

`-l`, `--local`
When searching for Image Packaging System (IPS) equivalents for the `package` keyword value in a JumpStart profile, search the IPS packages installed on the host system rather than the packages in an IPS package repository.

`-p` *profile*, `--profile` *profile*
Convert the specified JumpStart profile and generate a manifest for the profile processed. In this case, no criteria file is needed or generated.

`-r`, `--rule`
Convert rules and associated profiles and generate a manifest for each profile processed.

`-s`, `--sysidcfg`
Process the `sysidcfg` file and output the results to `sc_profile.xml`.

`-S`, `--skip`
Skip validation.

`-v`, `--verbose`
Provide details on the actions that occurred during processing.

`-V` *filename*
Validate the specified AI manifest file or SMF system configuration profile file. AI criteria validation is not supported.

Error Report The `js2ai` tool generates an error report when one or more errors occurs during the conversion.

```
# js2ai -r
                    Process  Unsupported  Conversion  Validation
Name      Warnings  Errors   Items        Errors      Errors
--------  --------  -------  -----------  ----------  ----------
rules            0        0            2           0           -
profile1         0        0            0           2           1

Conversion completed. One or more failures occurred.
For errors see ./js2ai.log
```

The report contains one entry for each file in which `js2ai` encountered an error. To generate an error report even when no errors occur, specify `-v` or `--verbose`.

The report tells you what type of errors occurred in what files. Five error types are defined: Warnings, Process Errors, Unsupported Items, Conversion Errors, and Validation Errors.

Warnings
Items in these messages are not required to be corrected. For example, you might receive a warning message that information such as host name or root password was not provided, and default values will be used.

Process Errors
These errors refer to problems that prevent `js2ai` from processing a file or a line within the file. Process errors typically occur when the file has a syntax error.

Unsupported Items
These items refer to a line that `js2ai` does not support. Changing the value associated with a keyword might eliminate this error.

Conversion Errors
: These errors refer to a condition that prevents `js2ai` from processing a line. These errors should be manually corrected, or the offending lines should be removed from the file.

Validation Errors
: These errors refer to the errors that occurred when the generated manifest was validated against the schema definition used by AI. These errors must to be corrected before the manifest can be used by AI.

The `js2ai.log` file indicates what error occurred on what line.

```
# cat js2ai.log
rules: line 4: unsupported keyword: disksize
rules: line 4: unsupported keyword: installed
net924_sun4c: line 4: unsupported keyword: cluster
net924_sun4c: line 5: unsupported keyword: num_clients
net924_sun4c: line 6: unsupported keyword: client_swap
net924_sun4c: line 7: unsupported keyword: client_arch
upgrade: line 1: unsupported value for 'install_type' specified: upgrade
```

If a validation error of the manifest occurs, the `js2ai.log` file contains a pointer to the log file that contains the validation errors, as shown in the following example:

```
Validation Errors:
    profile1: manifest validation of
    ./AI_profile1/profile1.xml failed.
    For details see ./AI_profile1/profile_validation.log
```

Conversion Strategy

Recommended Strategy for Rule and Profile Conversion

A one-to-one conversion between JumpStart and AI does not exist. The following steps provide a general procedure for performing the conversion.

1. The `js2ai` utility attempts to flag any errors it encounters, but `js2ai` assumes the rules, profiles, and `sysidcfg` files that are being converted are valid.
2. Copy the JumpStart configuration directory of `rules`, profile, and `syscfg` configuration files to an Oracle Solaris 11 system that has the `install/installadm` package installed.
3. In the JumpStart configuration directory that you copied to the Oracle Solaris 11 system in step 2, run the `js2ai` conversion tool.

   ```
   # js2ai -rS
   ```

   This command performs a conversion operation on the `rules` file and the profiles referenced by the `rules` file. Each profile referenced in the `rules` file is processed against the AI client provisioning manifest, `/usr/share/auto_install/manifest/default.xml`. This step creates a directory named `AI_`*profile* for each profile specified in the JumpStart `rules` file. The `AI_`*profile* directory contains one or more AI manifests for the translated profile in the form *profile*`${arch}.xml`. See the "Files" section for more information.

   The `-S` option skips the validation sequence. Validation is done in step 5.

4. If the message "Successfully completed conversion" is output, skip to step 5. Otherwise, examine the `js2ai.log` file and follow these steps:
   a. Correct any process errors.
   b. Remove any lines from the `rules` and profile files that are listed as Unsupported Items.
   c. Examine the conversion errors and correct the errors if possible. Otherwise, remove the lines that are causing the errors.
   d. Examine any warning messages and make sure no corrections are necessary.
   e. Repeat step 3 until no processing errors, unsupported items, and conversion errors are reported.
5. Rerun `js2ai` without the `-S` option.

   ```
   # js2ai -r
   ```

   If any validation errors occur for any of the processed profiles, the resulting AI manifest must be manually corrected. Examine the `js2ai.log` file for details of the failure. See the AI documentation for information about AI manifests.
6. Convert any `sysidcfg` files that are associated with this JumpStart configuration.

   For each `sysidcfg` file, execute the following command:

   ```
   # js2ai -sS -d sysidcfgdir
   ```

   For each `sysidcfg` file processed, this step creates an AI system configuration profile file named `sc_profile.xml` in the directory where the `js2ai` command was invoked. Use the `-D` option to specify a different directory for the `sc_profile.xml` file.
7. If the message "Successfully completed conversion" is output, skip to step 8. Otherwise, examine the `js2ai.log` file and follow these steps:
   a. Correct any process errors.
   b. Remove any lines from the `sysidcfg` file that are listed as unsupported items.
   c. Examine the conversion errors and correct the errors if possible. Otherwise, remove the lines that are causing the errors.
   d. Examine any warning messages and make sure no corrections are necessary.
   e. Repeat step 6 until no processing errors, unsupported items, and conversion errors are reported.
8. Rerun `js2ai` without the `-S` option.

   ```
   # js2ai -s -d sysidcfgdir
   ```

   If any validation errors occur for any of the processed `sysidcfg` files, the resulting AI system configuration profile must be manually corrected. Examine the `js2ai.log` file for details of the failure. See the AI documentation for information about system configuration profiles.

9. The `js2ai` conversion process is complete. Perform a manual verification of the resulting criteria, AI manifest, and system configuration profile files. The disk space requirements for an Oracle Solaris 11 installation are different from the disk space required for an Oracle Solaris 10 installation. Make sure the disk space allocated in your AI manifests meets the requirements of Oracle Solaris 11.
10. Configure AI to use the newly generated files. Add the newly generated criteria, AI manifest, and system configuration profile files to an existing AI install service.

    Use the `installadm` command with the `create-manifest` subcommand to add each AI manifest with criteria for selecting that manifest. Each client can use only one AI manifest.

    ```
    # installadm create-manifest -n svcname \
    -f filename -m manifest \
    -C criteriafile
    ```

    Use the `create-profile` subcommand to add each profile with criteria for selecting that configuration profile. Each client can use one or more system configuration profiles.

    ```
    # installadm create-profile -n svcname \
    -f filename -p profile \
    -C criteriafile
    ```

    See the AI documentation and the `installadm`(1M) man page for information about configuring AI install services.

Examples

EXAMPLE 1 Processing a JumpStart Configuration

The following command processes the JumpStart rules and profiles in the current directory. The output is also placed in this directory.

```
# js2ai -r
```

EXAMPLE 2 Processing a Specific JumpStart Directory

The following command processes the JumpStart rules and profiles from the specified directory and places the output files in the same directory.

```
# js2ai -r -d /export/jumpstart
```

For more information about the output files, see Example 4 and the "Files" section.

EXAMPLE 3 Processing a Profile in a Specific JumpStart Directory and Separate Destination Directory

The following command processes the JumpStart `rules` and profile files from the `/export/jumpstart` directory and places the output files in `/export/output`.

```
# js2ai -p profile1 -d /export/jumpstart -D /export/output
```

EXAMPLE 4 Example Input and the Resulting Output for a Specified Rule and Its Profile

Rule:

EXAMPLE 4 Example Input and the Resulting Output for a Specified Rule and Its Profile *(Continued)*

```
arch sparc && karch sun4u && \
    model 'SUNW,Serverblade1'  -   profile    -
```

Profile:

```
install_type    initial_install
pool mypool auto auto auto c1t0d0s0
```

Conversion command:

```
# js2ai -r -d /jumpstart -D /tmp/output
```

Output files:

```
/tmp/output/AI_profile/profile.x86.xml
/tmp/output/AI_profile/profile.sparc.xml
/tmp/output/AI_profile/criteria-1.xml
```

Two manifest files are created, one for SPARC and one for x86, even though the `rules` file specifies the CPU type as SPARC. During the conversion process, rules and profiles are processed independently of one another.

EXAMPLE 5 Adding Generated Files to an AI Install Service

This example adds the manifest and criteria to an existing service, using the files generated in Example 4.

Files:

```
/tmp/output/AI_profile/profile.sparc.xml
/tmp/output/AI_profile/criteria-1.xml
```

`installadm` command:

```
# installadm create-manifest -n svc-name \
-f /tmp/output/AI_profile/profile.sparc.xml \
-m sparc_profile \
-C /tmp/output/AI_profile/criteria-1.xml
```

EXAMPLE 6 Processing a `sysidcfg` File

The following command processes the `sysidcfg` file in the current directory and outputs the resulting SMF system configuration profile as `sc_profile.xml` in the same directory.

```
# js2ai -s
```

Exit Status

The following exit values are returned:

`0` All the files were processed successfully.

`>0` An error occurred.

Files *outputdir*/AI_${*profile*}
Directory that contains all the corresponding files that have been translated to the new AI syntax associated with the profile.

outputdir/AI_${*profile*}.${*arch*}.xml
The manifest file created as a result of translating the profile. ${*arch*} can be one of these three values: sparc, x86, or generic. A manifest file that is in the form ${*profile*}.generic.xml can be used to install both x86 and SPARC systems.

outputdir/AI_${*profile*}/criteria-*rule*.xml
The criteria-*rule*.xml file produced corresponds to the rule in the rules file. The *rule* is the rule number based on its position in the rules file. This criteria file can then be used with the -C option to the installadm command.

Since more than one rule can specify the same profile, more than one criteria file can exist in each directory, but only one instance of the ${*profile*}.${*arch*}.xml file should exist in each output directory.

Note – If the -p option is used, no criteria file is produced for the profile that is processed. Criteria files are only generated when used with the -r option.

outputdir/js2ai.err
This file contains a stack trace of an unexpected condition that occurred during processing. This file is not typically created.

outputdir/js2ai.log
This file contains a log of the files processed and any errors found during processing.

outputdir/sc_profile.xml
This file is the SMF system configuration profile that is generated when the -s option is used to convert a sysidcfg file.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	install/js2ai
Interface Stability	Uncommitted

See Also installadm(1M), pkg(1)

Transitioning From Oracle Solaris 10 JumpStart to Oracle Solaris 11.1 Automated Installer

Part III, "Installing Using an Install Server," in *Installing Oracle Solaris 11.1 Systems*

Name k5srvutil – host key table (keytab) manipulation utility

Synopsis `/usr/sbin/k5srvutil` *operation* `[-ik] [-f` *filename*`]`

Description The `k5srvutil` command allows a system manager to list or change keys currently in his keytab or to add new keys to the keytab.

The operand *operation* must be one of the following:

`list` Lists the keys in a keytab, showing version number and principal name.

`change` Changes all the keys in the keytab to new randomly-generated keys, updating the keys in the Kerberos server's database to match those by using the `kadmin` protocol. If a key's version number does not match the version number stored in the Kerberos server's database, the operation fails. The old keys are retained so that existing tickets continue to work. If the `-i` flag is specified, `k5srvutil` prompts for `yes` or `no` before changing each key. If the `-k` option is used, the old and new keys are displayed.

`delold` Deletes keys that are not the most recent version from the keytab. This operation should be used at some point after a change operation to remove old keys. If the `-i` flag is specified, `k5srvutil` asks the user whether the old keys associated with each principal should be removed.

`delete` Deletes particular keys in the keytab, interactively prompting for each key.

In all cases, the default keytab file is `/etc/krb5.keytab` file unless this is overridden by the `-f` option.

`k5srvutil` uses the `kadmin(1M)` program to edit the keytab in place. However, old keys are retained, so they are available in case of failure.

Options The following options are supported:

`-f` *filename* Specify a keytab file other than the default file, `/etc/krb5.keytab`.

`-i` Prompts user before changing keys when using the `change` or `delold` operands.

`-k` Displays old and new keys when using the `change` operand.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/security/kerberos-5
Interface Stability	Committed

See Also ktutil(1), kadmin(1M), attributes(5)

Name kadb – a kernel debugger

Synopsis

SPARC `ok boot` *device_specifier* `kadb [-d] [`*boot-flags*`]`

x86 `select (b)oot or (i)nterpreter: b kadb [-d] [`*boot-flags*`]`

Description `kadb`, an interactive kernel debugger, has been replaced by `kmdb(1)`. For backwards compatibility, the methods used to load `kadb` will be interpreted as requests to load `kmdb(1)`. Unlike with the compatibility link from `adb(1)` to `mdb(1)`, `kmdb(1)` will always load in its native user interface mode, regardless of the name used to load it.

`kmdb(1)` is based on `mdb(1)`, and thus shares `mdb`'s user interface style and feature set. The `mdb(1)` man page describes the features and operation of `mdb`. The `kmdb(1)` man page describes the differences between `mdb` and `kmdb`. This man page describes the major changes and incompatibilities between `kadb` and `kmdb`.

Consult the *Solaris Modular Debugger Guide* for a detailed description of both `mdb` and `kmdb`.

Major changes This section briefly lists the major differences between `kadb` and `kmdb`. It is not intended to be exhaustive.

Debugger Loading and Unloading	`kmdb(1)` may be loaded at boot, as with `kadb`. It may also be loaded after boot, thus allowing for kernel debugging and execution control without requiring a system reboot. If `kmdb(1)` is loaded after boot, it may be unloaded.
mdb Feature Set	The features introduced by `mdb(1)`, including access to kernel type data, debugger commands (dcmds), debugger modules (dmods), and enhanced execution control facilities, are available under `kmdb(1)`. Support for changing the representative CPU (:x) is available for both SPARC and x86. Furthermore, full execution-control facilities are available after the representative CPU has been changed.

Significant Incompatibilities This section lists the significant features that have changed incompatibly between `kadb` and `kmdb(1)`. It is not intended to be exhaustive. All `kmdb(1)` commands referenced here are fully described in the `kmdb(1)` man page. A description as well as examples can be found in the *Solaris Modular Debugger Guide*.

Deferred Breakpoints	The kadb-style “module#symbol:b” syntax is not supported under `kmdb(1)`. Instead, use “::bp module‘symbol”.
Watchpoints	The ::wp dcmd is the preferred way to set watchpoint with `kmdb`. Various options are available to control the type of watchpoint set, including `-p` for physical watchpoints

	(SPARC only), and -i for I/O port watchpoints (x86 only). $l is not supported, therefore, the watchpoint size must be specified for each watchpoint created.
Access to I/O Ports (x86 only)	The commands used to access I/O ports under kadb have been replaced with the ::in and ::out dcmds. These two dcmds allow both read and write of all I/O port sizes supported by kadb.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/library/processor

See Also adb(1), mdb(1), kmdb(1), attributes(5)

Solaris Modular Debugger Guide

Name kadmin, kadmin.local – Kerberos database administration program

Synopsis

```
/usr/sbin/kadmin [-r realm] [-p principal] [-q query]
 [-s admin_server [:port]] [ [-c credential_cache]
 | [-k [-t keytab] | [-n]] | [-w password]] [-x db_args]...

/usr/sbin/kadmin.local [-r realm] [-p principal]
 [-q query] [-d dbname] [-e "enc:salt..."] [-m] [-D]
```

Description `kadmin` and `kadmin.local` are interactive command-line interfaces to the Kerberos V5 administration system. They provide for the maintenance of Kerberos principals, policies, and service key tables (keytabs). `kadmin` and `kadmin.local` provide identical functionality; the difference is that `kadmin.local` can run only on the master KDC and does not use Kerberos authentication.

Except as explicitly noted otherwise, this man page uses `kadmin` to refer to both versions.

By default, both versions of `kadmin` attempt to determine your user name and perform operations on behalf of your "*username*/*admin*" instance. Operations performed are subject to privileges granted or denied to this user instance by the Kerberos ACL file (see `kadm5.acl(4)`). You may perform administration as another user instance by using the `-p` option.

The remote version, `kadmin`, uses Kerberos authentication and an encrypted RPC to operate securely from anywhere on the network. It normally prompts for a password and authenticates the user to the Kerberos administration server, `kadmind`, whose service principal is `kadmin/`*fqdn*. Some options specific to the remote version permit the password prompt to be bypassed. The `-c` option searches the named credentials cache for a valid ticket for the `kadmin/`*fqdn* service and uses it to authenticate the user to the Kerberos admin server without a password. The `-k` option searches a keytab for a credential to authenticate to the `kadmin/`*fqdn* service, and again no password is collected. If `kadmin` has collected a password, it requests a `kadmin/`*fqdn* Kerberos service ticket from the KDC, and uses that service ticket to interact with `kadmind`.

The local version, `kadmin.local`, must be run with an effective UID of root, and normally uses a key from the `/var/krb5/.k5.`*realm* stash file (see `kdb5_util(1M)`) to decrypt information from the database rather than prompting for a password. The `-m` option will bypass the `.k5.`*realm* stash file and prompt for the master password.

Options The following options are supported:

`-c` *credentials_cache*
: Search *credentials_cache* for a service ticket for the `kadmin/`*fqdn* service; it can be acquired with the `kinit(1)` program. If this option is not specified, `kadmin` requests a new service ticket from the KDC, and stores it in its own temporary credentials cache.

`-d` *dbname*
: Specify a non-standard database name. [Local only]

`-D`
: Turn on debug mode. [Local only]

`-e` *"enc:salt ..."*
Specify a different encryption type and/or key salt. [Local only]

`-k` [`-t` *keytab*]
Use the default keytab (`-k`) or a specific keytab (`-t` *keytab*) to decrypt the KDC response instead of prompting for a password. In this case, the default principal will be `host/hostname`. This is primarily used for keytab maintenance.

`-m`
Accept the database master password from the keyboard rather than using the `/var/krb5/.k5.`*realm* stash file. [Local only]

`-n`
Requests anonymous processing. Two types of anonymous principals are supported. For fully anonymous Kerberos, configure `pkinit` on the KDC and configure `pkinit_anchors` in the client's `krb5.conf`. Then use the `-n` option with a principal of the form `@`*REALM* (an empty principal name followed by the at-sign and a realm name). If permitted by the KDC, an anonymous ticket will be returned. A second form of anonymous tickets is supported; these realm-exposed tickets hide the identity of the client but not the client's realm. For this mode, use `kinit -n` with a normal principal name. If supported by the KDC, the principal (but not realm) will be replaced by the anonymous principal. As of release 1.8, the MIT Kerberos KDC supports only fully anonymous operation.

`-p` *principal*
Authenticate *principal* to the `kadmin/`*fqdn* service. Otherwise, `kadmin` will append `/admin` to the primary principal name of the default credentials cache, the value of the `USER` environment variable, or the username as obtained with `getpwuid`, in that order of preference.

`-q` *query*
Pass *query* directly to `kadmin`, which will perform *query* and then exit. This can be useful for writing scripts.

`-r` *realm*
Use *realm* as the default database realm.

`-s` *admin_server*[*:port*]
Administer the specified *admin* server at the specified port number (*port*). This can be useful in administering a realm not known to your client.

`-w` *password*
Use *password* instead of prompting for one. Note that placing the password for a Kerberos principal with administration access into a shell script can be dangerous if unauthorized users gain read access to the script or can read arguments of this command through `ps(1)`.

`-x` *db_args*
Pass database-specific arguments to `kadmin`. Supported arguments are for LDAP and the `Berkeley-db2` plug-in. These arguments are:

`binddn=`*binddn*
LDAP simple bind DN for authorization on the directory server. Overrides the `ldap_kadmind_dn` parameter setting in `krb5.conf(4)`.

`bindpwd=`*bindpwd*
Bind password.

`dbname=`*name*
For the `Berkeley-db2` plug-in, specifies a name for the Kerberos database.

`nconns=`*num*
Maximum number of server connections.

`port=`*num*
Directory server connection port.

Commands `list_requests`
Lists all the commands available for `kadmin`. Aliased by `lr` and `?`.

`get_privs`
Lists the current Kerberos administration privileges (ACLs) for the principal that is currently running `kadmin`. The privileges are based on the `/etc/krb5/kadm5.acl` file on the master KDC. Aliased by `getprivs`.

`add_principal` [*options*] *newprinc*
Creates a new principal, *newprinc*, prompting twice for a password. If the `-policy` option is not specified and a policy named `default` exists, then the `default` policy is assigned to the principal; note that the assignment of the `default` policy occurs automatically only when a principal is first created, so the `default` policy must already exist for the assignment to occur. The automatic assignment of the `default` policy can be suppressed with the `-clearpolicy` option. This command requires the `add` privilege. Aliased by `addprinc` and `ank`. The options are:

`-expire` *expdate*
Expiration date of the principal. See the `Time Formats` section for the valid absolute time formats that you can specify for *expdate*.

`-pwexpire` *pwexpdate*
Password expiration date. See the `Time Formats` section for the valid absolute time formats that you can specify for *pwexpdate*.

`-maxlife` *maxlife*
Maximum ticket life for the principal. See the `Time Formats` section for the valid time duration formats that you can specify for *maxlife*.

`-maxrenewlife` *maxrenewlife*
Maximum renewable life of tickets for the principal. See the `Time Formats` section for the valid time duration formats that you can specify for *maxrenewlife*.

`-kvno` *kvno*
Explicitly set the key version number.

`-policy` *policy*
Policy used by the principal. If both the `-policy` and `-clearpolicy` options are not specified, the `default` policy is used if it exists; otherwise, the principal will have no policy. Also note that the password and principal name must be different when you add a new principal with a specific policy or the `default` policy.

`-clearpolicy`
`-clearpolicy` prevents the `default` policy from being assigned when `-policy` is not specified. This option has no effect if the `default` policy does not exist.

{–|+}`allow_postdated`
`-allow_postdated` prohibits the principal from obtaining postdated tickets. (Sets the `KRB5_KDB_DISALLOW_POSTDATED` flag.) `+allow_postdated` clears this flag.

{–|+}`allow_forwardable`
`-allow_forwardable` prohibits the principal from obtaining forwardable tickets. (Sets the `KRB5_KDB_DISALLOW_FORWARDABLE` flag.) `+allow_forwardable` clears this flag.

{–|+}`allow_renewable`
`-allow_renewable` prohibits the principal from obtaining renewable tickets. (Sets the `KRB5_KDB_DISALLOW_RENEWABLE` flag.) `+allow_renewable` clears this flag.

{–|+}`allow_proxiable`
`-allow_proxiable` prohibits the principal from obtaining proxiable tickets. (Sets the `KRB5_KDB_DISALLOW_PROXIABLE` flag.) `+allow_proxiable` clears this flag.

{–|+}`allow_dup_skey`
`-allow_dup_skey` disables user-to-user authentication for the principal by prohibiting this principal from obtaining a session key for another user. (Sets the `KRB5_KDB_DISALLOW_DUP_SKEY` flag.) `+allow_dup_skey` clears this flag.

{–|+}`requires_preauth`
`+requires_preauth` requires the principal to preauthenticate before being allowed to `kinit`. (Sets the `KRB5_KDB_REQUIRES_PRE_AUTH` flag.) `-requires_preauth` clears this flag.

{–|+}`requires_hwauth`
`+requires_hwauth` requires the principal to preauthenticate using a hardware device before being allowed to kinit. (Sets the `KRB5_KDB_REQUIRES_HW_AUTH` flag.) `-requires_hwauth` clears this flag.

{–|+}`allow_svr`
`-allow_svr` prohibits the issuance of service tickets for the principal. (Sets the `KRB5_KDB_DISALLOW_SVR` flag.) `+allow_svr` clears this flag.

{–|+}`allow_tgs_req`
–`allow_tgs_req` specifies that a Ticket-Granting Service (TGS) request for a service ticket for the principal is not permitted. This option is useless for most things.

+allow_tgs_req clears this flag. The default is +allow_tgs_req. In effect, –allow_tgs_req sets the KRB5_KDB_DISALLOW_TGT_BASED flag on the principal in the database.

{–|+}allow_tix
–allow_tix forbids the issuance of any tickets for the principal. +allow_tix clears this flag. The default is +allow_tix. In effect, –allow_tix sets the KRB5_KDB_DISALLOW_ALL_TIX flag on the principal in the database.

{–|+}needchange
+needchange sets a flag in attributes field to force a password change; –needchange clears it. The default is –needchange. In effect, +needchange sets the KRB5_KDB_REQUIRES_PWCHANGE flag on the principal in the database.

{–|+}password_changing_service
+password_changing_service sets a flag in the attributes field marking this as a password change service principal (useless for most things). –password_changing_service clears the flag. This flag intentionally has a long name. The default is –password_changing_service. In effect, +password_changing_service sets the KRB5_KDB_PWCHANGE_SERVICE flag on the principal in the database.

{-|+}ok_as_delegate
+ok_as_delegate sets the OK-AS-DELEGATE flag on tickets issued for use with this principal as the service, which clients may use as a hint that credentials can and should be delegated when authenticating to the service. (Sets the KRB5_KDB_OK_AS_DELEGATE flag.) -ok_as_delegate clears this flag.

{-|+}ok_to_auth_as_delegate
+ok_to_auth_as_delegate sets the service to allow the use of S4U2Self. -ok_to_auth_as_delegate clears this flag.

-randkey
Sets the key of the principal to a random value.

-pw *password*
Sets the key of the principal to the specified string and does not prompt for a password. Note that using this option in a shell script can be dangerous if unauthorized users gain read access to the script.

-e “enc:salt ...”
Override the list of enctype:salttype pairs given in kdc.conf(4) for setting the key of the principal. The quotes are necessary if there are multiple enctype:salttype pairs. One key for each similar enctype and same salttype will be created and the first one listed will be used. For example, in a list of two similar enctypes with the same salt, “des-cbc-crc:normal des-cbc-md5:normal”, one key will be created and it will be of type des-cbc-crc:normal.

Example:

```
kadmin: addprinc tlyu/admin
WARNING: no policy specified for "tlyu/admin@ACME.COM";
defaulting to no policy.
Enter password for principal tlyu/admin@ACME.COM:
Re-enter password for principal tlyu/admin@ACME.COM:
Principal "tlyu/admin@ACME.COM" created.
kadmin:
```

Errors:

KADM5_AUTH_ADD (requires add privilege)

KADM5_BAD_MASK (should not happen)

KADM5_DUP (principal exists already)

KADM5_UNK_POLICY (policy does not exist)

KADM5_PASS_Q_* (password quality violations)

`delete_principal` [`-force`] *principal*

Deletes the specified principal from the database. This command prompts for deletion, unless the `-force` option is given. This command requires the `delete` privilege. Aliased by `delprinc`.

Example:

```
kadmin: delprinc mwm_user
Are you sure you want to delete the principal
"mwm_user@ACME.COM"? (yes/no): yes
Principal "mwm_user@ACME.COM" deleted.
Make sure that you have removed this principal from
all kadmind ACLs before reusing.
kadmin:
```

Errors:

KADM5_AUTH_DELETE (requires `delete` privilege)

KADM5_UNK_PRINC (principal does not exist)

`modify_principal` [*options*] *principal*

Modifies the specified principal, changing the fields as specified. The options are as above for `add_principal`, except that password changing is forbidden by this command. In addition, the option `-clearpolicy` will clear the current policy of a principal. This command requires the `modify` privilege. Aliased by `modprinc`.

`-unlock`

Unlocks the principal so that it can successfully authenticate. If the principal had previously been locked due to reaching `maxfailure` in `failurecountinterval` time then the principal will be locked for `lockoutduration` time.

Errors:

`KADM5_AUTH_MODIFY` (requires `modify` privilege)

`KADM5_UNK_PRINC` (principal does not exist)

`KADM5_UNK_POLICY` (policy does not exist)

`KADM5_BAD_MASK` (should not happen)

`change_password` [*options*] *principal*

Changes the password of *principal*. Prompts for a new password if neither `-randkey` or `-pw` is specified. Requires the `changepw` privilege, or that the principal that is running the program to be the same as the one changed. Aliased by `cpw`. The following options are available:

`-randkey`

Sets the key of the principal to a random value.

`-pw` *password*

Sets the password to the specified string. Not recommended.

`-e` "enc:salt ..."

Override the list of enctype:salttype pairs given in `kdc.conf(4)` for setting the key of the principal. The quotes are necessary if there are multiple enctype:salttype pairs. For each key, the first matching similar enctype and same salttype in the list will be used to set the new key(s).

`-keepold`

Keeps the previous kvno's keys around. There is no easy way to delete the old keys, and this flag is usually not necessary except perhaps for TGS keys as it will allow existing valid TGTs to continue to work.

Example:

```
kadmin: cpw systest
Enter password for principal systest@ACME.COM:
Re-enter password for principal systest@ACME.COM:
Password for systest@ACME.COM changed.
kadmin:
```

Errors:

`KADM5_AUTH_MODIFY` (requires the `modify` privilege)

`KADM5_UNK_PRINC` (principal does not exist)

`KADM5_PASS_Q_*` (password policy violation errors)

`KADM5_PASS_REUSE` (password is in principal's password history)

`KADM5_PASS_TOOSOON` (current password minimum life not expired)

`get_principal` [`-terse`] *principal*

Gets the attributes of *principal*. Requires the `inquire` privilege, or that the principal that is running the program to be the same as the one being listed. With the `-terse` option, outputs fields as quoted tab-separated strings. Aliased by `getprinc`.

Examples:

```
kadmin: getprinc tlyu/admin
Principal: tlyu/admin@ACME.COM
Expiration date: [never]
Last password change: Thu Jan 03 12:17:46 CET 2008
Password expiration date: [none]
Maximum ticket life: 24855 days 03:14:07
Maximum renewable life: 24855 days 03:14:07
Last modified: Thu Jan 03 12:17:46 CET 2008 (root/admin@ACME.COM)
Last successful authentication: [never]
Last failed authentication: [never]
Failed password attempts: 0
Number of keys: 5
Key: vno 2, AES-256 CTS mode with 96-bit SHA-1 HMAC, no salt
Key: vno 2, AES-128 CTS mode with 96-bit SHA-1 HMAC, no salt
Key: vno 2, Triple DES cbc mode with HMAC/sha1, no salt
Key: vno 2, ArcFour with HMAC/md5, no salt
Key: vno 2, DES cbc mode with RSA-MD5, no salt
Attributes: REQUIRES_PRE_AUTH
Policy: [none]
kadmin: getprinc -terse tlyu/admin
"tlyu/admin@ACME.COM"   0       1199359066      0       2147483647
"root/admin@ACME.COM"   1199359066      128     2       0       "[none]"       21474836
47      0       0       0       5       1       2       18      0       1      2
17      0       1       2       16      0       1       2       23      0      12
       3       0
kadmin:
```

Errors:

`KADM5_AUTH_GET` (requires the get [inquire] privilege)

`KADM5_UNK_PRINC` (principal does not exist)

`list_principals` [*expression*]

Retrieves all or some principal names. *expression* is a shell-style glob expression that can contain the wild-card characters ?, *, and []'s. All principal names matching the expression are printed. If no expression is provided, all principal names are printed. If the expression does not contain an "@" character, an "@" character followed by the local realm is appended to the expression. Requires the `list` privilege. Aliased by `listprincs`, `get_principals`, and `getprincs`.

Examples:

```
kadmin: listprincs test*
test3@ACME.COM
test2@ACME.COM
test1@ACME.COM
testuser@ACME.COM
kadmin:
```

`add_policy` [*options*] *policy*
Adds the named policy to the policy database. Requires the `add` privilege. Aliased by `addpol`. The following options are available:

`-maxfailure` *maxnumber*
sets the maximum number of failures before the principal is locked after authentication failures in *failurecountinterval* time.

`-failurecountinterval` *failuretime*
sets the time after which the authentication failure count is reset 0. See the "Time Formats" section, below, for the valid time duration formats that you can specify for *failuretime*.

`-lockoutduration` *lockouttime*
sets the time in which the principal is locked from authenticating if *maxfailure* authentication failures occur within *failurecountinterval* time. See the "Time Formats" section, below, for the valid time duration formats that you can specify for *lockouttime*.

`-maxlife` *maxlife*
sets the maximum lifetime of a password. See the `Time Formats` section for the valid time duration formats that you can specify for *maxlife*.

`-minlife` *minlife*
sets the minimum lifetime of a password. See the `Time Formats` section for the valid time duration formats that you can specify for *minlife*.

`-minlength` *length*
sets the minimum length of a password.

`-minclasses` *number*
sets the minimum number of character classes allowed in a password. The valid values are:

`1`
only letters (himom)

`2`
both letters and numbers (hi2mom)

`3`
letters, numbers, and punctuation (hi2mom!)

`-history` *number*
sets the number of past keys kept for a principal.

Errors:
`KADM5_AUTH_ADD` (requires the `add` privilege)

`KADM5_DUP` (policy already exists)

`delete_policy` [`-force`] *policy*
Deletes the named policy. Unless the `-force` option is specified, prompts for confirmation before deletion. The command will fail if the policy is in use by any principals. Requires the `delete` privilege. Aliased by `delpol`.

Example:

```
kadmin: del_policy guests
Are you sure you want to delete the
policy "guests"? (yes/no): yes
Policy "guests" deleted.
kadmin:
```

Errors:
`KADM5_AUTH_DELETE` (requires the delete privilege)

`KADM5_UNK_POLICY` (policy does not exist)

`KADM5_POLICY_REF` (reference count on policy is not zero)

`modify_policy` [*options*] *policy*
Modifies the named policy. Options are as above for `add_policy`. Requires the `modify` privilege. Aliased by `modpol`.

Errors:
`KADM5_AUTH_MODIFY` (requires the `modify` privilege)

`KADM5_UNK_POLICY` (policy does not exist)

`get_policy` [`-terse`] *policy*
Displays the values of the named policy. Requires the `inquire` privilege. With the `-terse` flag, outputs the fields as quoted strings separated by tabs. Aliased by `getpol`.

Examples:

```
kadmin: get_policy admin
Policy: admin
Maximum password life: 180 days 00:00:00
Minimum password life: 00:00:00
Minimum password length: 6
Minimum number of password character classes: 2
Number of old keys kept: 5
Reference count: 17
```

```
Maximum password failures before lockout: 3
Password failure count reset interval: 180
Password lockout duration: 60
kadmin: get_policy -terse admin
admin admin    15552000    0    6    2    5    17 3 180 60
kadmin:
```

Errors:

`KADM5_AUTH_GET` (requires the `get` privilege)

`KADM5_UNK_POLICY` (policy does not exist)

`list_policies` [*expression*]

Retrieves all or some policy names. *expression* is a shell-style glob expression that can contain the wild-card characters ?, *, and []'s. All policy names matching the expression are printed. If no expression is provided, all existing policy names are printed. Requires the `list` privilege. Aliased by `listpols`, `get_policies`, and `getpols`.

Examples:

```
kadmin: listpols
test-pol dict-only once-a-min test-pol-nopw
kadmin: listpols t*
test-pol test-pol-nopw kadmin:
```

`ktadd` [`-k` *keytab*] [`-q`] [`-e` *keysaltlist*] [`-norandkey`] [[*principal* | `-glob` *princ-exp*]] [...]

Adds a principal or all principals matching *princ-exp* to a keytab. It randomizes each principal's key in the process, to prevent a compromised `admin` account from reading out all of the keys from the database. However, `kadmin.local` has the `-norandkey` option, which leaves the keys and their version numbers unchanged, similar to the Kerberos V4 `ext_srvtab` command. That allows users to continue to use the passwords they know to login normally, while simultaneously allowing scripts to login to the same account using a `keytab`. There is no significant security risk added since `kadmin.local` must be run by root on the KDC anyway.

`ktadd` requires the `inquire` and `changepw` privileges. An entry for each of the principal's unique encryption types is added, ignoring multiple keys with the same encryption type but different `salt` types. If the `-k` argument is not specified, the default keytab file, `/etc/krb5/krb5.keytab`, is used.

The "`-e` *enctype*`:salt`" option overrides the list of *enctypes* given in `krb5.conf(4)`, in the `permitted_enctypes` parameter. If "`-e` *enctype*`:salt`" is not used and `permitted_enctypes` is not defined in `krb5.conf(4)`, a key for each *enctype* supported by the system on which `kadmin` is run will be created and added to the `keytab`. Restricting the *enctypes* of keys in the `keytab` is useful when the system for which keys are being created does not support the same set of *enctypes* as the KDC. Note that `ktadd` modifies the *enctype* of the keys in the principal database as well.

If the `-q` option is specified, less status information is displayed. Aliased by `xst`. The `-glob` option requires the `list` privilege. Also, note that if you use `-glob` to create a keytab, you need to remove `/etc/krb5/kadm5.keytab` and create it again if you want to use `-p */admin` with `kadmin`.

`princ-exp`
princ-exp follows the same rules described for the `list_principals` command.

Example:

```
kadmin: ktadd -k /tmp/new-keytab nfs/chicago
Entry for principal nfs/chicago with kvno 2,
encryption type DES-CBC-CRC added to keytab
WRFILE:/tmp/new-keytab.
kadmin:
```

`ktremove` [`-k` *keytab*] [`-q`] *principal* [*kvno* | `all` | `old`]
Removes entries for the specified principal from a keytab. Requires no privileges, since this does not require database access. If `all` is specified, all entries for that principal are removed; if `old` is specified, all entries for that principal except those with the highest kvno are removed. Otherwise, the value specified is parsed as an integer, and all entries whose `kvno` match that integer are removed. If the `-k` argument is not specified, the default keytab file, `/etc/krb5/krb5.keytab`, is used. If the `-q` option is specified, less status information is displayed. Aliased by `ktrem`.

Example:

```
kadmin: ktremove -k /tmp/new-keytab nfs/chicago
Entry for principal nfs/chicago with kvno 2
removed from keytab
WRFILE:/tmp/new-keytab.
kadmin:
```

`quit`
Quits `kadmin`. Aliased by `exit` and `q`.

Time Formats

Various commands in `kadmin` can take a variety of time formats, specifying time durations or absolute times. The `kadmin` option variables *maxrenewlife*, *maxlife*, and *minlife* are time durations, whereas *expdate* and *pwexpdate* are absolute times.

Examples:

```
kadmin: modprinc -expire "12/31 7pm" jdb
kadmin: modprinc -maxrenewlife "2 fortnight" jdb
kadmin: modprinc -pwexpire "this sunday" jdb
kadmin: modprinc -expire never jdb
kadmin: modprinc -maxlife "7:00:00pm tomorrow" jdb
```

Note that times which do not have the “ago” specifier default to being absolute times, unless they appear in a field where a duration is expected. In that case, the time specifier will be interpreted as relative. Specifying “ago” in a duration can result in unexpected behavior.

The following time formats and units can be combined to specify a time. The time and date format examples are based on the date and time of July 2, 1999, 1:35:30 p.m.

Time Format	Examples
hh[:*mm*][:*ss*][am/pm/a.m./p.m.]	1p.m., 1:35, 1:35:30pm

Variable	Description
hh	hour (12-hour clock, leading zero permitted but not required)
mm	minutes
ss	seconds

Date Format	Examples
mm/dd[/*yy*]	07/02, 07/02/99
yyyy-*mm*-dd	1999-07-02
dd-*month*-*yyyy*	02-July-1999
month [,*yyyy*]	Jul 02, July 02,1999
dd *month*[*yyyy*]	02 JULY, 02 july 1999

Variable	Description
dd	day
mm	month
yy	year within century (00-38 is 2000 to 2038; 70-99 is 1970 to 1999)
yyyy	year including century
month	locale's full or abbreviated month name

Time Units	Examples
[+|- #] year	“-2 year”
[+|- #] month	“2 months”
[+|- #] fortnight	

[+|- #] week	
[+|- #] day	
[+|- #] hour	
[+|- #] minute	
[+|- #] min	
[+|- #] second	
[+|- #] sec	
tomorrow	
yesterday	
today	
now	
this	“this year”
last	“last saturday”
next	“next month”
sunday	
monday	
tuesday	
wednesday	
thursday	
friday	
saturday	
never	

You can also use the following time modifiers: `first`, `second`, `third`, `fourth`, `fifth`, `sixth`, `seventh`, `eighth`, `ninth`, `tenth`, `eleventh`, `twelfth`, and `ago`.

Environment Variables

See `environ(5)` for descriptions of the following environment variables that affect the execution of `kadmin`:

PAGER
: The command to use as a filter for paging output. This can also be used to specify options. The default is `more(1)`.

Files

`/var/krb5/principal`
Kerberos principal database.

`/var/krb5/principal.ulog`
The update log file for incremental propagation.

`/var/krb5/principal.kadm5`
Kerberos administrative database. Contains policy information.

`/var/krb5/principal.kadm5.lock`
Lock file for the Kerberos administrative database. This file works backwards from most other lock files (that is, `kadmin` will exit with an error if this file does *not* exist).

`/var/krb5/kadm5.dict`
Dictionary of strings explicitly disallowed as passwords.

`/etc/krb5/kadm5.acl`
List of principals and their `kadmin` administrative privileges.

`/etc/krb5/kadm5.keytab`
Keytab for `kadmind` principals: `kadmin`/*fqdn*, `changepw`/*fqdn*, and `kadmin`/`changepw`.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/security/kerberos-5
Interface Stability	Committed

See Also

`kpasswd(1)`, `more(1)`, `gkadmin(1M)`, `kadmind(1M)`, `kdb5_util(1M)`, `kdb5_ldap_util(1M)`, `kproplog(1M)`, `kadm5.acl(4)`, `kdc.conf(4)`, `krb5.conf(4)`, `attributes(5)`, `environ(5)`, `kerberos(5)`, `krb5envvar(5)`

History

The `kadmin` program was originally written by Tom Yu at MIT, as an interface to the OpenVision Kerberos administration program.

Diagnostics

The `kadmin` command is currently incompatible with the MIT `kadmind` daemon interface, so you cannot use this command to administer an MIT-based Kerberos database. However, clients running the Solaris implementation of Kerberos can still use an MIT-based KDC.

Name kadmind – Kerberos administration daemon

Synopsis

```
/usr/lib/krb5/kadmind [-nofork] [-m] [-port port-number] [-r realm]
     [-x db_args] [-P pid_file]
```

Description `kadmind` runs on the master key distribution center (KDC), which stores the principal and policy databases. `kadmind` accepts remote requests to administer the information in these databases. Remote requests are sent, for example, by `kpasswd(1)`, `gkadmin(1M)`, and `kadmin(1M)` commands, all of which are clients of `kadmind`. When you install a KDC, `kadmind` is set up in the `init` scripts to start automatically when the KDC is rebooted.

`kadmind` requires a number of configuration files to be set up for it to work:

`/etc/krb5/kdc.conf`
: The KDC configuration file contains configuration information for the KDC and the Kerberos administration system. `kadmind` understands a number of configuration variables (called relations) in this file, some of which are mandatory and some of which are optional. In particular, `kadmind` uses the `acl_file`, `dict_file`, `admin_keytab`, and `kadmind_port` relations in the [*realms*] section. Refer to the `kdc.conf(4)` man page for information regarding the format of the KDC configuration file.

`/etc/krb5/kadm5.keytab`
: `kadmind` requires a `keytab` (key table) containing correct entries for the `kadmin/`*fqdn*, `kadmin/changepw` and `kadmin/changepw` principals for every realm that `kadmind` answers requests. The `keytab` can be created with the `kadmin(1M)` or `kdb5_util(1M)` command. The location of the keytab is determined by the `admin_keytab` relation in the `kdc.conf`(4) file.

`/etc/krb5/kadm5.acl`
: `kadmind` uses an ACL (access control list) to determine which principals are allowed to perform Kerberos administration actions. The path of the ACL file is determined by the `acl_file` relation in the `kdc.conf` file. See `kdc.conf(4)`. For information regarding the format of the ACL file, refer to `kadm5.acl(4)`.

  The `kadmind` daemon will need to be restarted to reread the `kadm5.acl` file after it has been modified. You can do this, as root, with the following command:

  ```
  # svcadm restart svc:/network/security/kadmin:default
  ```

After `kadmind` begins running, it puts itself in the background and disassociates itself from its controlling terminal.

`kadmind` can be configured for incremental database propagation. Incremental propagation allows slave KDC servers to receive principal and policy updates incrementally instead of receiving full dumps of the database. These settings can be changed in the `kdc.conf(4)` file:

`sunw_dbprop_enable = [true | false]`
: Enable or disable incremental database propagation. Default is `false`.

`sunw_dbprop_master_ulogsize = N`
Specifies the maximum amount of log entries available for incremental propagation to the slave KDC servers. The maximum value that this can be is 2500 entries. Default value is 1000 entries.

The `kiprop/`<*hostname*>`@`<*REALM*> principal must exist in the master's `kadm5.keytab` file to enable the slave to authenticate incremental propagation from the master. In the principal syntax above, <*hostname*> is the master KDC's host name and <*REALM*> is the realm in which the master KDC resides.

Kerberos client machines can automatically migrate Unix users to the default Kerberos realm specified in the local `krb5.conf(4)`, if the user does not have a valid kerberos account already. You achieve this by using the `pam_krb5_migrate(5)` service module for the service in question. The Kerberos service principal used by the client machine attempting the migration needs to be validated using the `u` privilege in `kadm5.acl(4)`. When using the `u` privilege, `kadmind` validates user passwords using PAM, specifically using a `PAM_SERVICE` name of `k5migrate` by calling `pam_authenticate(3PAM)` and `pam_acct_mgmt(3PAM)`.

A suitable PAM stack configuration example for `k5migrate` would look like:

```
k5migrate       auth    required        pam_unix_auth.so.1
k5migrate       account required        pam_unix_account.so.1
```

Options The following options are supported:

`-nofork`
Specifies that `kadmind` does not put itself in the background and does not disassociate itself from the terminal. In normal operation, you should use the default behavior, which is to allow the daemon to put itself in the background.

`-m`
Specifies that the master database password should be retrieved from the keyboard rather than from the stash file. When using `-m`, the `kadmind` daemon receives the password prior to putting itself in the background. If used in combination with the `-d` option, you must explicitly place the daemon in the background.

`-port` *port-number*
Specifies the port on which the `kadmind` daemon listens for connections. The default is controlled by the `kadmind_port` relation in the `kdc.conf(4)` file.

`-P` *pid_file*
Specifies the file to which the PID of `kadmind` process should be written to after it starts up. This can be used to identify whether `kadmind` is still running and to allow `init` scripts to stop the correct process.

`-r` *realm*
Specifies the default realm that `kadmind` serves. If *realm* is not specified, the default *realm* of the host is used. `kadmind` answers requests for any realm that exists in the local KDC database and for which the appropriate principals are in its `keytab`.

`-x` *db_args*
Pass database-specific arguments to `kadmind`. Supported arguments are for LDAP and the `Berkeley-db2` plug-in. These arguments are:

`binddn`=*binddn*
LDAP simple bind DN for authorization on the directory server. Overrides the `ldap_kadmind_dn` parameter setting in `krb5.conf(4)`.

`bindpwd`=*bindpwd*
Bind password.

`dbname`=*name*
For the `Berkeley-db2` plug-in, specifies a name for the Kerberos database.

`nconns`=*num*
Maximum number of server connections.

`port`=*num*
Directory server connection port.

Files

`/var/krb5/principal`
Kerberos principal database.

`/var/krb5/principal.ulog`
The update log file for incremental propagation.

`/var/krb5/principal.kadm5`
Kerberos administrative database containing policy information.

`/var/krb5/principal.kadm5.lock`
Kerberos administrative database lock file. This file works backwards from most other lock files (that is, `kadmin` exits with an error if this file does not exist).

`/var/krb5/kadm5.dict`
Dictionary of strings explicitly disallowed as passwords.

`/etc/krb5/kadm5.acl`
List of principals and their `kadmin` administrative privileges.

`/etc/krb5/kadm5.keytab`
Keytab for `kadmin` principals: `kadmin/`*fqdn*, `changepw/`*fqdn*, and `kadmin/changepw`.

`/etc/krb5/kdc.conf`
KDC configuration information.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/security/kerberos-5

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Committed

See Also kpasswd(1), svcs(1), gkadmin(1M), kadmin(1M), kadmin(1M), kdb5_util(1M), kdb5_ldap_util(1M), kproplog(1M), svcadm(1M), pam_acct_mgmt(3PAM), pam_authenticate(3PAM), kadm5.acl(4), kdc.conf(4), krb5.conf(4), attributes(5), kerberos(5), krb5envvar(5), pam_krb5_migrate(5), smf(5)

Notes The Kerberos administration daemon (`kadmind`) is now compliant with the change-password standard mentioned in RFC 3244, which means it can now handle change-password requests from non-Solaris Kerberos clients.

The `kadmind` service is managed by the service management facility, smf(5), under the service identifier:

```
svc:/network/security/kadmin
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using svcadm(1M). The service's status can be queried using the svcs(1) command.

The `-d` and `-p` arguments are made obsolete with the `-nofork` and `-port` arguments, respectively. The `-d` and `-p` arguments might be removed in a future release of the Solaris operating system.

Name kcfd – kernel-level cryptographic framework daemon

Synopsis

```
kcfd
```

Description The `kcfd` daemon helps in managing CPU usage by cryptographic operations performed in software by kernel threads. The system utilization associated with these threads is charged to the `kcfd` process. It also does module verification for kernel cryptographic modules.

Only a privileged user can run this daemon.

The `kcfd` daemon is automatically invoked in run level 1, after `/usr` is mounted. A previously invoked `kcfd` daemon that is still running must be stopped before invoking another `kcfd` command.

Manually starting and restarting `kcfd` is not recommended. If it is necessary to do so, use the `cryptoadm`(1M) `start` and `stop` subcommands.

Exit Status The following exit values are returned:

`0` Daemon started successfully.

`> 1` Daemon failed to start.

Attributes See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os
Interface Stability	Committed

See Also `cryptoadm`(1M), `attributes`(5)

Name kclient – set up a machine as a Kerberos client

Synopsis

```
/usr/sbin/kclient [-n] [-R realm] [-a adminuser] [-c filepath]
    [-d dnsarg] [-f fqdn_list] [-h logical_host_name] [-k kdc_list]
    [-m master_kdc_list] [-p profile] [-s pam_service] [-T kdc_vendor]
```

Description By specifying the various command options, you can use the `kclient` utility to:

- Configure a machine as a Kerberos client for a specified realm and for KDC by setting up `krb5.conf(4)`.
- Add the Kerberos host principal to the local host's `keytab` file (`/etc/krb5/krb5.keytab`).
- Set up the machine to do kerberized NFS.
- Bring over a master `krb5.conf` copy from a specified pathname.
- Setup a machine to do server and/or host/domain name-to-realm mapping lookups by means of DNS.
- Configure a Kerberos client to use an MS Active Directory server. This generates a `keytab` file with the Kerberos client's service keys populated.
- Setup a Kerberos client that has no service keys. This is useful when the client does not require service keys, because the client does not wish to host a service that uses Kerberos for security.
- Configure a Kerberos client that is part of a cluster. This option requires the logical host name of the cluster so that the proper service keys are created and populated in the client's `keytab` file.
- Setup a Kerberos client to join an environment that consists of Kerberos servers that are non-Solaris and non-MS Active Directory servers.
- Configure `pam.conf(4)` to use Kerberos authentication for specified services.
- Configure the client as a simple NTP broadcast/multicast client.
- Specify custom domain/host name-to-realm name mappings.
- Setup the Kerberos client to use multiple KDC servers.

The `kclient` utility needs to be run on the client machine with root permission and can be run either interactively or non-interactively. In the non-interactive mode, the user feeds in the required inputs by means of a profile, command-line options, or a combination of profile and command-line options. The user is prompted for "required" parameter values (`realm` and `adminuser`), if found missing in the non-interactive run. The interactive mode is invoked when the utility is run without any command-line arguments.

Both the interactive and non-interactive forms of `kclient` can add the `host/fqdn` entry to the local host's `keytab` file. They also can require the user to enter the password for the administrative user requested, to obtain the Kerberos Ticket Granting Ticket (TGT) for `adminuser`. The `host/fqdn`, `nfs/fqdn`, and `root/fqdn` principals can be added to the KDC database (if not already present) before their possible addition to the local host's `keytab`.

The `kclient` utility assumes that the local host has been setup for DNS and requires the presence of a valid `resolv.conf(4)`. Also, `kclient` can fail if the localhost time is not synchronized with that of the KDC. For Kerberos to function the localhost time must be within five minutes of that of the KDC. It is advised that both systems run some form of time synchronization protocol, such as the Network Time Protocol (NTP). See the `ntpd` man page, delivered in the `SUNWntpu` package (not a SunOS man page).

Options The non-interactive mode supports the following options:

`-n`
Set up the machine for kerberized NFS. This involves making changes to `krb5*` security flavors in `nfssec.conf(4)`. This option will also add `nfs/fqdn` and `root/fqdn` entries to the local host's `keytab` file if the `-K` option has not been specified.

`-R` [*realm*]
Specifies the Kerberos realm.

`-k` *kdc_list*
The `-k` option specifies the KDC host names for the Kerberos client. *kdc_list* is a comma-separated list of KDCs. If the `-m` option is not used, it is assumed that the first (or only) host in *kdc_list* is the master KDC host name. Note that the list specified is used verbatim. This is helpful when specifying non-fully qualified KDC host names that can be canonicalized by DNS.

`-a` [*adminuser*]
Specifies the Kerberos administrative user.

`-T` *kdc_vendor*
Configure the Kerberos client to associate with a third party server. Valid *kdc_vendor* currently supported are:

`ms_ad`
Microsoft Active Directory

`mit`
MIT KDC server

`heimdal`
Heimdal KDC server

`shishi`
Shishi KDC server

Knowing the administrative password will be required to associate the client with the server if the `ms_ad` option is specified.

`-c` [*filepath*]
Specifies the pathname to the `krb5.conf(4)` master file, to be copied over to the local host. The path specified normally points to a master copy on a remote host and brought over to the local host by means of NFS.

`-d` [*dnsarg*]
Specifies the DNS lookup option to be used and specified in the `krb5.conf(4)` file. Valid *dnsarg* entries are: `none`, `dns_lookup_kdc`, `dns_lookup_realm` and `dns_fallback`. Any other entry is considered invalid. The latter three *dnsarg* values assume the same meaning as those described in `krb5.conf`. `dns_lookup_kdc` implies DNS lookups for the KDC and the other servers. `dns_lookup_realm` is for host/domain name-to-realm mapping by means of DNS. `dns_fallback` is a superset and does DNS lookups for both the servers and the host/domain name-to-realm mapping. A lookup option of `none` specifies that DNS is not be used for any kind of mapping lookup.

`-D` *domain_list*
Specifies the host and/or domain names to be mapped to the Kerberos client's default realm name. *domain_list* is a comma–separated list, for example "`example.com,host1.example.com`". If the `-D` option is not used, then only the client's domain is used for this mapping. For example, if the client is `host1.eng.example.com`, then the domain that is mapped to the `EXAMPLE.COM` realm is `example.com`.

`-K`
Configure the Kerberos client without service keys, which are usually stored in `/etc/krb5/krb5.keytab`. This is useful in the following scenarios:

- The client IP address is dynamically assigned and therefore does not host Kerberized services.
- Client has a static IP address, but does not want to host any Kerberized services.
- Client has a static IP address, but the local administrator does not currently have service keys available for the machine. It is expected that, at a later time, these keys will be installed on the machine.

`-f` [*fqdn_list*]
This option creates a service principal entry (host/nfs/root) associated with each of the listed fqdn's, if required, and subsequently adds the entries to the local host's `keytab`.

fqdn_list is a comma-separated list of one or more fully qualified DNS domain names.

This option is especially useful in Kerberos realms having systems offering kerberized services, but situated in multiple different DNS domains.

`-h` *logical_host_name*
Specifies that the Kerberos client is a node in a cluster. The *logical_host_name* is the logical host name given to the cluster. The resulting `/etc/krb5/krb5.conf` and `/etc/krb5/krb5.keytab` files must be manually copied over to the other members of the cluster.

`-m` *master_kdc_list*
This option specifies the master KDC host names to be used by the Kerberos client. *master_kdc_list* is a comma-separated list of the host names of master KDCs for the client. If the `-m` option is not used, then it is assumed that the first KDC host name listed with the `-k` option is the master KDC.

`-p` [*profile*]
Specifies the profile to be used to enable the reading in of the values of all the parameters required for setup of the machine as a Kerberos client.

The profile should have entries in the format:

PARAM <value>

Valid *PARAM* entries are: `REALM`, `KDC`, `ADMIN`, `FILEPATH`, `NFS`, `DNSLOOKUP`, `FQDN`, `NOKEY`, `NOSOL`, `LHN`, `KDCVENDOR`, `RMAP`, `MAS`, and `PAM`.

These profile entries correspond to the `-R` [*realm*], `-k` [*kdc*], `-a` [*adminuser*], `-c` [*filepath*], `-n`, `-d` [*dnsarg*], `-f` [*fqdn_list*], `-K`, `-h` [*logical_host_name*], `-T` [*kdc_vendor*], `-D` [*domain_list*], `-m` [*master_kdc*], and `-s` [*pam_service*] command-line options, respectively. Any other *PARAM* entry is considered invalid and is ignored.

The NFS profile entry can have a value of 0 (do nothing) or 1 (operation is requested). Any other value is considered invalid and is ignored.

Keep in mind that the command line options override the *PARAM* values listed in the profile.

`-s` *pam_service*:*auth_type*[,...]
Specifies that the PAM service names, listed in *pam_service*, are authenticated through Kerberos. Using this option updates `pam.conf(4)` to include a separate authentication stack with `pam_krb5(5)`. Examples of *pam_service* names are `sshd-kbdint`, `xscreensaver`, and so forth.

auth_type can be one of the following keywords:

`first`
Try authenticating through Kerberos first. If this fails try to authenticate through Unix.

`only`
Try to authenticate only through Kerberos.

`optional`
Try authenticating through Unix first. If this is successful try to authenticate through Kerberos.

Examples

EXAMPLE 1 Setting Up a Kerberos Client Using Command-Line Options

To setup a Kerberos client using the `clntconfig/admin` administrative principal for realm `'ABC.COM'`, `kdc` '`example1.com`' and that also does kerberized NFS, enter:

```
# /usr/sbin/kclient -n -R ABC.COM -k example1.com -a clntconfig
```

Alternatively, to set up a Kerberos client using the `clntconfig/admin` administrative principal for the realm '`EAST.ABC.COM`', `kdc` '`example2.east.abc.com`' and that also needs service principal(s) created and/or added to the local `keytab` for multiple DNS domains, enter:

EXAMPLE 1 Setting Up a Kerberos Client Using Command-Line Options *(Continued)*

```
# /usr/sbin/kclient -n -R EAST.ABC.COM -k example2.east.abc.com \
-f west.abc.com,central.abc.com -a clntconfig
```

Note that the `krb5` administrative principal used by the administrator needs to have only `add`, `inquire`, `change-pwd` and `modify` privileges (for the principals in the KDC database) in order for the `kclient` utility to run. A sample `kadm5.acl(4)` entry is:

```
clntconfig/admin@ABC.COM acmi
```

EXAMPLE 2 Setting Up a Kerberos Client Using the Profile Option

To setup a Kerberos client using the `clntconfig/admin` administrative principal for realm '`ABC.COM`', `kdc` '`example1.com`' and that also copies over the master `krb5.conf` from a specified location, enter:

```
# /usr/sbin/kclient -p /net/example1.com/export/profile.krb5
```

The contents of `profile.krb5`:

```
REALM ABC.COM
KDC example1
ADMIN clntconfig
FILEPATH /net/example1.com/export/krb5.conf
NFS 0
DNSLOOKUP none
```

EXAMPLE 3 Setting Up a Kerberos Client That Has a Dynamic IP Address

In this example a Kerberos client is a DHCP client that has a dynamic IP address. This client does not wish to host any Kerberized services and therefore does not require a `keytab` (`/etc/krb5/krb5.keytab`) file.

For this type of client the administrator would issue the following command to configure this machine to be a Kerberos client of the `ABC.COM` realm with the KDC server `kdc1.example.com`:

```
# /usr/sbin/kclient -K -R EXAMPLE.COM -k kdc1.example.com
```

Files

`/etc/krb5/kadm5.acl`
: Kerberos access control list (ACL) file.

`/etc/krb5/krb5.conf`
: Default location for the local host's configuration file.

`/etc/krb5/krb5.keytab`
: Default location for the local host's `keytab` file.

`/etc/nfssec.conf`
: File listing NFS security modes.

`/etc/resolv.conf`
DNS resolver configuration file.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/security/kerberos-5
Interface Stability	Committed

See Also `encrypt(1)`, `ksh(1)`, `ldapdelete(1)`, `ldapmodify(1)`, `ldapsearch(1)`, `dd(1M)`, `smbadm(1M)`, `kadm5.acl(4)`, `krb5.conf(4)`, `nfssec.conf(4)`, `pam.conf(4)`, `resolv.conf(4)`, `attributes(5)`, `pam_krb5(5)`

Notes `fqdn` stands for the Fully Qualified Domain Name of the local host. The `kclient` utility saves copies of both the `krb5.conf(4)` and `nfssec.conf(4)` files to files with corresponding names and `.sav` extensions. The optional copy of the `krb5.conf(4)` master file is neither encrypted nor integrity-protected and it takes place over regular NFS.

Name kdb5_ldap_util – Kerberos configuration utility

Synopsis

```
kdb5_ldap_util  [-D user_dn [-w passwd]] [-H ldap_uri] command
    [command_options]
```

Description The `kdb5_ldap_util` utility allows an administrator to manage realms, Kerberos services, and ticket policies. The utility offers a set of general options, described under OPTIONS, and a set of commands, which, in turn, have their own options. Commands and their options are described in their own subsections, below.

Options `kdb5_ldap_util` has a small set of general options that apply to the `kdb5_ldap_util` utility itself and a larger number of options that apply to specific commands. A number of these command-specific options apply to multiple commands and are described in their own section, below.

General Options The following general options are supported:

`-D` *user_dn*
Specifies the distinguished name (DN) of a user who has sufficient rights to perform the operation on the LDAP server.

`-H` *ldap_uri*
Specifies the URI of the LDAP server.

`-w` *passwd*
Specifies the password of *user_dn*. This option is not recommended.

Common Command-specific Options The following options apply to a number of `kdb5_ldap_util` commands.

`-subtrees` *subtree_dn_list*
Specifies the list of subtrees containing the principals of a realm. The list contains the DNs of the subtree objects separated by a colon.

`-sscope` *search_scope*
Specifies the scope for searching the principals under a subtree. The possible values are 1 or `one` (one level), 2 or `sub` (subtrees).

`-containerref` *container_reference_dn*
Specifies the DN of the container object in which the principals of a realm will be created. If the container reference is not configured for a realm, the principals will be created in the realm container.

`-maxtktlife` *max_ticket_life*
Specifies maximum ticket life for principals in this realm.

`-maxrenewlife` *max_renewable_ticket_life*
Specifies maximum renewable life of tickets for principals in this realm.

`-r` *realm*
Specifies the Kerberos realm of the database; by default the realm returned by `krb5_default_local_realm(3)` is used.

kdb5_ldap_util Commands

The `kdb5_ldap_util` utility comprises a set of commands, each with its own set of options. These commands are described in the following subsections.

The `create` Command

The `create` command creates a realm in a directory. The command has the following syntax:

```
create \
[-subtrees subtree_dn_list]
[-sscope search_scope]
[-containerref container_reference_dn]
[-k mkeytype]
[-m|-P password| -sf stashfilename]
[-s]
[-r realm]
[-maxtktlife max_ticket_life]
[-kdcdn kdc_service_list]
[-admindn admin_service_list]
[-maxrenewlife max_renewable_ticket_life]
[ticket_flags]
```

The `create` command has the following options:

`-subtree` *subtree_dn_list*
: See "Common Command-specific Options," above.

`-sscope` *search_scope*
: See "Common Command-specific Options," above.

`-containerref` *container_reference_dn*
: See "Common Command-specific Options," above.

`-k` *mkeytype*
: Specifies the key type of the master key in the database; the default is that given in `kdc.conf(4)`.

`-m`
: Specifies that the master database password should be read from the TTY rather than fetched from a file on the disk.

`-P` *password*
: Specifies the master database password. This option is not recommended.

`-sf` *stashfilename*
: Specifies the stash file of the master database password.

`-s`
: Specifies that the stash file is to be created.

`-maxtktlife` *max_ticket_life*
: See "Common Command-specific Options," above.

`-maxrenewlife` *max_renewable_ticket_life*
: See "Common Command-specific Options," above.

`-r` *realm*
See "Common Command-specific Options," above.

ticket_flags
Specifies the ticket flags. If this option is not specified, by default, none of the flags are set. This means all the ticket options will be allowed and no restriction will be set. See "Ticket Flags" for a list and descriptions of these flags.

The `modify` Command

The `modify` command modifies the attributes of a realm. The command has the following syntax:

```
modify \
[-subtrees subtree_dn_list]
[-sscope search_scope]
[-containerref container_reference_dn]
[-r realm]
[-maxtktlife max_ticket_life]
[-maxrenewlife max_renewable_ticket_life]
[ticket_flags]
```

The `modify` command has the following options:

`-subtree` *subtree_dn_list*
See "Common Command-specific Options," above.

`-sscope` *search_scope*
See "Common Command-specific Options," above.

`-containerref` *container_reference_dn*
See "Common Command-specific Options," above.

`-maxtktlife` *max_ticket_life*
See "Common Command-specific Options," above.

`-maxrenewlife` *max_renewable_ticket_life*
See "Common Command-specific Options," above.

`-r` *realm*
See "Common Command-specific Options," above.

ticket_flags
Specifies the ticket flags. If this option is not specified, by default, none of the flags are set. This means all the ticket options will be allowed and no restriction will be set. See "Ticket Flags" for a list and descriptions of these flags.

The `view` Command

The `view` command displays the attributes of a realm. The command has the following syntax:

```
view [-r realm]
```

The `view` command has the following option:

`-r` *realm*
See "Common Command-specific Options," above.

The `destroy` Command

The `destroy` command destroys a realm, including the master key stash file. The command has the following syntax:

```
destroy [-f] [-r realm]
```

The `destroy` command has the following options:

`-f`
If specified, `destroy` does not prompt you for confirmation.

`-r` *realm*
See "Common Command-specific Options," above.

The `list` Command

The `list` command displays the names of realms. The command has the following syntax:

```
list
```

The `list` command has no options.

The `stashsrvpw` Command

The `stashsrvpw` command enables you to store the password for service object in a file so that a KDC and Administration server can use it to authenticate to the LDAP server. The command has the following syntax:

```
stashsrvpw [-f filename] servicedn
```

The `stashsrvpw` command has the following option and argument:

`-f` *filename*
Specifies the complete path of the service password file. The default is:

```
/var/krb5/service_passwd
```

servicedn
Specifies the distinguished name (DN) of the service object whose password is to be stored in file.

The `create_policy` Command

The `create_policy` command creates a ticket policy in a directory. The command has the following syntax:

```
create_policy \
[-r realm]
[-maxtktlife max_ticket_life]
[-maxrenewlife max_renewable_ticket_life]
[ticket_flags]
policy_name
```

The `create_policy` command has the following options:

`-r` *realm*
See "Common Command-specific Options," above.

`-maxtktlife` *max_ticket_life*
See "Common Command-specific Options," above.

`-maxrenewlife` *max_renewable_ticket_life*
See "Common Command-specific Options," above.

ticket_flags
Specifies the ticket flags. If this option is not specified, by default, none of the flags are set. This means all the ticket options will be allowed and no restriction will be set. See "Ticket Flags" for a list and descriptions of these flags.

policy_name
Specifies the name of the ticket policy.

The `modify_policy` Command

The `modify_policy` command modifies the attributes of a ticket policy. The command has the following syntax:

```
modify_policy \
[-r realm]
[-maxtktlife max_ticket_life]
[-maxrenewlife max_renewable_ticket_life]
[ticket_flags]
policy_name
```

The `modify_policy` command has the same options and argument as those for the `create_policy` command.

The `view_policy` Command

The `view_policy` command displays the attributes of a ticket policy. The command has the following syntax:

```
view_policy [-r realm] policy_name
```

The `view_policy` command has the following options:

`-r` *realm*
See "Common Command-specific Options," above.

policy_name
Specifies the name of the ticket policy.

The `destroy_policy` Command

The `destroy_policy` command destroys an existing ticket policy. The command has the following syntax:

```
destroy_policy [-r realm] [-force] policy_name
```

The `destroy_policy` command has the following options:

`-r` *realm*
See "Common Command-specific Options," above.

`-force`
Forces the deletion of the policy object. If not specified, you will be prompted for confirmation before the policy is deleted. Enter `yes` to confirm the deletion.

policy_name
Specifies the name of the ticket policy.

The `list_policy` Command

The `list_policy` command lists the ticket policies in the default or a specified realm. The command has the following syntax:

```
list_policy [-r realm]
```

The `list_policy` command has the following option:

`-r` *realm*
See "Common Command-specific Options," above.

Ticket Flags

A number of `kdb5_ldap_util` commands have `ticket_flag` options. These flags are described as follows:

`{-|+}allow_dup_skey`
`-allow_dup_skey` disables user-to-user authentication for principals by prohibiting principals from obtaining a session key for another user. This setting sets the `KRB5_KDB_DISALLOW_DUP_SKEY` flag. `+allow_dup_skey` clears this flag.

`{-|+}allow_forwardable`
`-allow_forwardable` prohibits principals from obtaining forwardable tickets. This setting sets the `KRB5_KDB_DISALLOW_FORWARDABLE` flag. `+allow_forwardable` clears this flag.

`{-|+}allow_postdated`
`-allow_postdated` prohibits principals from obtaining postdated tickets. This setting sets the `KRB5_KDB_DISALLOW_POSTDATED` flag. `+allow_postdated` clears this flag.

`{-|+}allow_proxiable`
`-allow_proxiable` prohibits principals from obtaining proxiable tickets. This setting sets the `KRB5_KDB_DISALLOW_PROXIABLE` flag. `+allow_proxiable` clears this flag.

`{-|+}allow_renewable`
`-allow_renewable` prohibits principals from obtaining renewable tickets. This setting sets the `KRB5_KDB_DISALLOW_RENEWABLE` flag. `+allow_renewable` clears this flag.

`{-|+}allow_svr`
`-allow_svr` prohibits the issuance of service tickets for principals. This setting sets the `KRB5_KDB_DISALLOW_SVR` flag. `+allow_svr` clears this flag.

`{-|+}allow_tgs_req`
`-allow_tgs_req` specifies that a Ticket-Granting Service (TGS) request for a service ticket for principals is not permitted. This option is useless for most purposes. `+allow_tgs_req` clears this flag. The default is `+allow_tgs_req`. In effect, `-allow_tgs_req` sets the `KRB5_KDB_DISALLOW_TGT_BASED` flag on principals in the database.

{-|+}allow_tix
-allow_tix forbids the issuance of any tickets for principals. +allow_tix clears this flag. The default is +allow_tix. In effect, -allow_tix sets the KRB5_KDB_DISALLOW_ALL_TIX flag on principals in the database.

{-|+}needchange
+needchange sets a flag in the attributes field to force a password change; -needchange clears that flag. The default is -needchange. In effect, +needchange sets the KRB5_KDB_REQUIRES_PWCHANGE flag on principals in the database.

{-|+}password_changing_service
+password_changing_service sets a flag in the attributes field marking a principal as a password-change-service principal (a designation that is most often not useful). -password_changing_service clears the flag. That this flag has a long name is intentional. The default is -password_changing_service. In effect, +password_changing_service sets the KRB5_KDB_PWCHANGE_SERVICE flag on principals in the database.

{-|+}requires_hwauth
+requires_hwauth requires principals to preauthenticate using a hardware device before being allowed to kinit(1). This setting sets the KRB5_KDB_REQUIRES_HW_AUTH flag. -requires_hwauth clears this flag.

{-|+}requires_preauth
+requires_preauth requires principals to preauthenticate before being allowed to kinit(1). This setting sets the KRB5_KDB_REQUIRES_PRE_AUTH flag. -requires_preauth clears this flag.

Examples

EXAMPLE 1 Using create

The following is an example of the use of the create command.

```
# kdb5_ldap_util -D cn=admin,o=org -H ldaps://ldap-server1.mit.edu \
create -subtrees o=org -sscope SUB -r ATHENA.MIT.EDU
Password for "cn=admin,o=org":  password entered
Initializing database for realm 'ATHENA.MIT.EDU'
You will be prompted for the database Master Password.
It is important that you NOT FORGET this password.
Enter KDC database master key: master key entered
Re-enter KDC database master key to verify: master key re-enteredjjjjjj
```

EXAMPLE 2 Using modify

The following is an example of the use of the modify command.

```
# kdb5_ldap_util -D cn=admin,o=org -H ldaps://ldap-server1.mit.edu \
modify +requires_preauth -r ATHENA.MIT.EDU
Password for "cn=admin,o=org":  password entered
Password for "cn=admin,o=org":  password entered
```

EXAMPLE 3 Using `view`

The following is an example of the use of the `view` command.

```
# kdb5_ldap_util -D cn=admin,o=org -H ldaps://ldap-server1.mit.edu \
view -r ATHENA.MIT.EDU
          Password for "cn=admin,o=org":
                            Realm Name: ATHENA.MIT.EDU
                               Subtree: ou=users,o=org
                               Subtree: ou=servers,o=org
                           SearchScope: ONE
                  Maximum ticket life: 0 days 01:00:00
               Maximum renewable life: 0 days 10:00:00
                         Ticket flags: DISALLOW_FORWARDABLE REQUIRES_PWCHANGE
```

EXAMPLE 4 Using `destroy`

The following is an example of the use of the `destroy` command.

```
# kdb5_ldap_util -D cn=admin,o=org -H ldaps://ldap-server1.mit.edu \
destroy -r ATHENA.MIT.EDU
Password for "cn=admin,o=org": password entered
Deleting KDC database of 'ATHENA.MIT.EDU', are you sure?
(type 'yes' to confirm)? yes
OK, deleting database of 'ATHENA.MIT.EDU'...
```

EXAMPLE 5 Using `list`

The following is an example of the use of the `list` command.

```
# kdb5_ldap_util -D cn=admin,o=org -H ldaps://ldap-server1.mit.edu list
Password for "cn=admin,o=org": password entered
Re-enter Password for "cn=admin,o=org": password re-entered
ATHENA.MIT.EDU
OPENLDAP.MIT.EDU
MEDIA-LAB.MIT.EDU
```

EXAMPLE 6 Using `stashsrvpw`

The following is an example of the use of the `stashsrvpw` command.

```
# kdb5_ldap_util stashsrvpw -f \
/home/andrew/conf_keyfile cn=service-kdc,o=org
Password for "cn=service-kdc,o=org": password entered
Re-enter password for "cn=service-kdc,o=org": password re-entered
```

EXAMPLE 7 Using `create_policy`

The following is an example of the use of the `create_policy` command.

```
# kdb5_ldap_util -D cn=admin,o=org -H ldaps://ldap-server1.mit.edu \
create_policy -r ATHENA.MIT.EDU \
-maxtktlife "1  day" -maxrenewlife "1 week" \
```

EXAMPLE 7 Using `create_policy` *(Continued)*

```
-allow_postdated +needchange -allow_forwardable tktpolicy
Password for "cn=admin,o=org": password entered
```

EXAMPLE 8 Using `modify_policy`

The following is an example of the use of the `modify_policy` command.

```
# kdb5_ldap_util -D cn=admin,o=org -H ldaps://ldap-server1.mit.edu \
modify_policy -r ATHENA.MIT.EDU \
-maxtktlife "60 minutes" -maxrenewlife "10 hours" \
+allow_postdated -requires_preauth tktpolicy
Password for "cn=admin,o=org": password entered
```

EXAMPLE 9 Using `view_policy`

The following is an example of the use of the `view_policy` command.

```
# kdb5_ldap_util -D cn=admin,o=org -H ldaps://ldap-server1.mit.edu \
view_policy -r ATHENA.MIT.EDU tktpolicy
Password for "cn=admin,o=org": password entered
            Ticket policy: tktpolicy
      Maximum ticket life: 0 days 01:00:00
   Maximum renewable life: 0 days 10:00:00
             Ticket flags: DISALLOW_FORWARDABLE REQUIRES_PWCHANGE
```

EXAMPLE 10 Using `destroy_policy`

The following is an example of the use of the `destroy_policy` command.

```
# kdb5_ldap_util -D cn=admin,o=org -H ldaps://ldap-server1.mit.edu \
destroy_policy -r ATHENA.MIT.EDU tktpolicy
Password for "cn=admin,o=org": password entered
This will delete the policy object 'tktpolicy', are you sure?
(type 'yes' to confirm)? yes
** policy object 'tktpolicy' deleted.
```

EXAMPLE 11 Using `list_policy`

The following is an example of the use of the `list_policy` command.

```
# kdb5_ldap_util -D cn=admin,o=org -H ldaps://ldap-server1.mit.edu \
list_policy -r ATHENA.MIT.EDU
Password for "cn=admin,o=org": password entered
tktpolicy
tmppolicy
userpolicy
```

EXAMPLE 12 Using `setsrvpw`

The following is an example of the use of the `setsrvpw` command.

EXAMPLE 12 Using `setsrvpw` *(Continued)*

```
# kdb5_ldap_util setsrvpw -D cn=admin,o=org setsrvpw \
-fileonly -f /home/andrew/conf_keyfile cn=service-kdc,o=org
Password for "cn=admin,o=org": password entered
Password for "cn=service-kdc,o=org": password entered
Re-enter password for "cn=service-kdc,o=org": password re-entered
```

EXAMPLE 13 Using `create_service`

The following is an example of the use of the `create_service` command.

```
# kdb5_ldap_util -D cn=admin,o=org create_service \
-kdc -randpw -f /home/andrew/conf_keyfile cn=service-kdc,o=org
Password for "cn=admin,o=org": password entered
File does not exist. Creating the file /home/andrew/conf_keyfile...
```

EXAMPLE 14 Using `modify_service`

The following is an example of the use of the `modify_service` command.

```
# kdb5_ldap_util -D cn=admin,o=org modify_service \
-realm ATHENA.MIT.EDU cn=service-kdc,o=org
Password for "cn=admin,o=org": password entered
Changing rights for the service object. Please wait ... done
```

EXAMPLE 15 Using `view_service`

The following is an example of the use of the `view_service` command.

```
# kdb5_ldap_util -D cn=admin,o=org view_service \
cn=service-kdc,o=org
Password for "cn=admin,o=org": password entered
                      Service dn: cn=service-kdc,o=org
                    Service type: kdc
                Service host list:
                    Realm DN list: cn=ATHENA.MIT.EDU,cn=Kerberos,cn=Security
```

EXAMPLE 16 Using `destroy_service`

The following is an example of the use of the `destroy_service` command.

```
# kdb5_ldap_util -D cn=admin,o=org destroy_service \
cn=service-kdc,o=org
Password for "cn=admin,o=org": password entered
This will delete the service object 'cn=service-kdc,o=org', are you sure?
(type 'yes' to confirm)? yes
** service object 'cn=service-kdc,o=org' deleted.
```

EXAMPLE 17 Using `list_service`

The following is an example of the use of the `list_service` command.

EXAMPLE 17 Using `list_service` *(Continued)*

```
# kdb5_ldap_util -D cn=admin,o=org list_service
Password for "cn=admin,o=org": password entered
cn=service-kdc,o=org
cn=service-adm,o=org
cn=service-pwd,o=org
```

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	service/security/kerberos-5
Interface Stability	Volatile

See Also

`kinit(1)`, `kadmin(1M)`, `kdc.conf(4)`, `attributes(5)`

Name kdb5_util – Kerberos Database maintenance utility

Synopsis

```
/usr/sbin/kdb5_util [-d dbname] [-k mkeytype] [-kv mkeyVNO] [-m ]
     [-M mkeyname] [-P password] [-r realm] [-sf stashfilename]
     [-x db_args]... cmd
```

Description The `kdb5_util` utility enables you to create, dump, load, and destroy the Kerberos V5 database. You can also use `kdb5_util` to create a stash file containing the Kerberos database master key.

Options The following options are supported:

`-d` *dbname*
Specify the database name. `.db` is appended to whatever name is specified. You can specify an absolute path. If you do not specify the `-d` option, the default database name is `/var/krb5/principal`.

`-k` *mkeytype*
Specify the master key type. Valid values are `des3-cbc-sha1`, `des-cbc-crc`, `des-cbc-md5`, `des-cbc-raw`, `arcfour-hmac-md5`, `arcfour-hmac-md5-exp`, `aes128-cts-hmac-sha1-96`, and `aes256-cts-hmac-sha1-96`.

`-m`
Enter the master key manually.

`-M` *mkeyname*
Specify the master key name.

`-P` *password*
Use the specified *password* instead of the stash file.

`-r` *realm*
Use *realm* as the default database realm.

`-sf` *stashfile_name*
Specifies the stash file of the master database password.

`-x` *db_args*
Pass database-specific arguments to `kadmin`. Supported arguments are for LDAP and the `Berkeley-db2` plug-in. These arguments are:

`binddn`=*binddn*
LDAP simple bind DN for authorization on the directory server. Overrides the `ldap_kadmind_dn` parameter setting in `krb5.conf(4)`.

`bindpwd`=*bindpwd*
Bind password.

`dbname`=*name*
For the `Berkeley-db2` plug-in, specifies a name for the Kerberos database.

`nconns=`*num*
Maximum number of server connections.

`port=`*num*
Directory server connection port.

Operands The following operands are supported:

cmd
Specifies whether to create, destroy, dump, or load the database, or to create a stash file.

You can specify the following commands:

`create -s`
Creates the database specified by the `-d` option. You will be prompted for the database master password. If you specify `-s`, a stash file is created as specified by the `-f` option. If you did not specify `-f`, the default stash file name is `/var/krb5/.k5.realm`. If you use the `-f`, `-k`, or `-M` options when you create a database, then you must use the same options when modifying or destroying the database.

`destroy`
Destroys the database specified by the `-d` option.

`stash`
Creates a stash file. If `-f` was not specified, the default stash file name is `/var/krb5/.k5.realm`. You will be prompted for the master database password. This command is useful when you want to generate the stash file from the password.

`dump` [`-old`] [`-b6`] [`-b7`] [`-ov`] [`-r13`] [`-verbose`] [`-mkey_convert`] [`-new_mkey_file` *mkey_file*] [`-rev`] [`-recurse`] [*filename* [*principals*...]]
Dumps the current Kerberos and KADM5 database into an ASCII file. By default, the database is dumped in current format, "`kdb5_util load_dumpversion 6`". If *filename* is not specified or is the string "-", the dump is sent to standard output. Options are as follows:

`-old`
Causes the dump to be in the Kerberos 5 Beta 5 and earlier dump format ("`kdb5_edit load_dump version 2.0`").

`-b6`
Causes the dump to be in the Kerberos 5 Beta 6 format ("`kdb5_edit load_dump version 3.0`").

`-b7`
Causes the dump to be in the Kerberos 5 Beta 7 format ("`kdb5_util load_dump version 4`"). This was the dump format produced on releases prior to 1.2.2.

`-ov`
Causes the dump to be in `ovsec_adm_export` format.

`-r13`
Causes the dump to be in the Kerberos 5 1.3 format ("`kdb5_util load_dump version 5`"). This was the dump format produced on releases prior to 1.8.

`-verbose`
Causes the name of each principal and policy to be displayed as it is dumped.

`-mkey_convert`
Prompts for a new master key. This new master key will be used to re-encrypt the key data in the dumpfile. The key data in the database will not be changed.

`-new_mkey_file` *mkey_file*
The filename of a stash file. The master key in this stash file will be used to re-encrypt the key data in the dumpfile. The key data in the database will not be changed.

`-rev`
Dumps in reverse order. This might recover principals that do not dump normally, in cases where database corruption has occured.

`-recurse`
Causes the dump to walk the database recursively (`btree` only). This might recover principals that do not dump normally, in cases where database corruption has occurred. In cases of such corruption, this option will probably retrieve more principals than will the `-rev` option.

`load` [`-old`] [`-b6`] [`-b7`] [`-ov`] [`-r13`] [`-hash`] [`-verbose`] [`-update`] *filename dbname* [*admin_dbname*]
Loads a database dump from *filename* into *dbname*. Unless the `-old` or `-b6` option is specified, the format of the dump file is detected automatically and handled appropriately. Unless the `-update` option is specified, `load` creates a new database containing only the principals in the dump file, overwriting the contents of any existing database. The `-old` option requires the database to be in the Kerberos 5 Beta 5 or earlier format ("`kdb5_edit load_dump version 2.0`").

`-b6`
Requires the database to be in the Kerberos 5 Beta 6 format ("`kdb5_edit load_dump version 3.0`").

`-b7`
Requires the database to be in the Kerberos 5 Beta 7 format ("`kdb5_util load_dump version 4`").

`-ov`
Requires the database to be in `ovsec_adm_import` format. Must be used with the `-update` option.

`-hash`
Requires the database to be stored as a hash. If this option is not specified, the database will be stored as a `btree`. This option is not recommended, as databases stored in hash format are known to corrupt data and lose principals.

`-r13`
Causes the dump to be in the Kerberos 5 1.3 format ("`kdb5_util load_dump version 5`"). This was the dump format produced on releases prior to 1.8.

`-verbose`
Causes the name of each principal and policy to be displayed as it is dumped.

`-update`
Records from the dump file are added to or updated in the existing database. Otherwise, a new database is created containing only what is in the dump file and the old one is destroyed upon successful completion.

filename
Required argument that specifies a path to a file containing database dump.

dbname
Required argument that overrides the value specified on the command line or overrides the default.

admin_dbname
Optional argument that is derived from *dbname* if not specified.

`add_mkey` [`-e` *etype*] [`-s`]
Adds a new master key to the K/M (master key) principal. Existing master keys will remain. The `-e` *etype* option allows specification of the `enctype` of the new master key. The `-s` option stashes the new master key in a local stash file which will be created if it does not already exist.

`use_mkey` *mkeyVNO* [*time*]
Sets the activation time of the master key specified by *mkeyVNO*. Once a master key is active (that is, its activation time has been reached) it will then be used to encrypt principal keys either when the principal keys change, are newly created, or when the `update_princ_encryption` command is run. If the *time* argument is provided, that will be the activation time; otherwise the current time is used by default. The format of the optional time argument is that specified in the Time Formats section of the `kadmin(1M)` man page.

`list_mkeys`
List all master keys from most recent to earliest in K/M principal. The output will show the KVNO, `enctype` and `salt` for each `mkey`, similar to `kadmin getprinc` output. An asterisk (*) following an `mkey` denotes the currently active master key.

`purge_mkeys` [`-f`] [`-n`] [`-v`]
Delete master keys from the K/M principal that are not used to protect any principals. This command can be used to remove old master keys from a K/M principal once all principal keys are protected by a newer master key.

`-f`
Does not prompt user.

`-n`
Does a dry run, shows master keys that would be purged, but does not actually purge any keys.

`-v`
Verbose output.

`update_princ_encryption` [`-f`] [`-n`] [`-v`] [*princ-pattern*]
Update all principal records (or only those matching the *princ-pattern* glob pattern) to re-encrypt the key data using the active database master key, if they are encrypted using older versions, and give a count at the end of the number of principals updated. If the `-f` option is not given, ask for confirmation before starting to make changes. The `-v` option causes each principal processed (each one matching the pattern) to be listed, and an indication given as to whether it needed updating or not. The `-n` option causes the actions not to be taken, only the normal or verbose status messages displayed; this implies `-f`, since no database changes will be performed and thus there is little reason to seek confirmation.

Examples

EXAMPLE 1 Creating File that Contains Information about Two Principals

The following example creates a file named `slavedata` that contains the information about two principals, `jdb@ACME.COM` and `pak@ACME.COM`.

```
# /usr/krb5/bin/kdb5_util dump -verbose slavedata
jdb@ACME.COM pak@ACME.COM
```

Files

`/var/krb5/principal`
Kerberos principal database.

`/var/krb5/principal.kadm5`
Kerberos administrative database. Contains policy information.

`/var/krb5/principal.kadm5.lock`
Lock file for the Kerberos administrative database. This file works backwards from most other lock files (that is, `kadmin` exits with an error if this file does not exist).

`/var/krb5/principal.ulog`
The update log file for incremental propagation.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/security/kerberos-5
Interface Stability	Committed

See Also

`kpasswd(1)`, `gkadmin(1M)`, `kadmin(1M)`, `kadmind(1M)`, `kadmin.local(1M)`, `kdb5_ldap_util(1M)`, `kproplog(1M)`, `kadm5.acl(4)`, `kdc.conf(4)`, `attributes(5)`, `kerberos(5)`

Notes The global `-f` is made obsolete with the `-sf` argument for specifying a non-default stash file location. The global `-f` argument might be removed in a future release of the Solaris operating system. Use caution in specifying `-f` as it has different semantics in subcommands as distinguished from its use as a global argument.

Name kdcmgr – set up a Kerberos Key Distribution Center (KDC)

Synopsis

```
/usr/sbin/kdcmgr [-a admprincipal] [-e enctype]
     [-h] [-p pwfile] [-r realm] subcommand
```

Description Use the `kdcmgr` utility to do the following:

- Configure a master Key Distribution Center (KDC) server.
- Configure a slave KDC. This assumes that a master KDC has already been configured. The default propagation method configured is incremental propagation. See `kpropd(1M)`.
- Specify a list of slave KDCs to configure service principals and create access control list for those slaves on the master KDC.

If you specify no options, `kdcmgr` prompts you for required information, including a password to generate the master key and a password for the administrative principal. When you specify sufficient options, you are still prompted for these passwords, unless you specified the `-p` *pwfile* option.

The `kdcmgr` utility must be run by an administrator who is assigned the Kerberos Server Management rights profile or by root. The command must be run on the server from which it is invoked.

Note that `kdcmgr` requires the user to enter sensitive information, such as the password used to generate the database's master key and the password for the administrative principal. Great care must be taken to ensure that the connection to the server is secured over the network, by using a protocol such as `ssh(1)`.

You must also exercise great care when selecting the administrative and master key passwords. They should be derived from non-dictionary words and a long string of characters consisting of all of the following character classes:

- special characters (for example, !@#$%^&*)
- numerals (0-9)
- uppercase letters
- lowercase letters

Options The following options are supported:

`-a` *admprincipal*
When creating a master KDC, specifies the administrative principal, *admprincipal*, that will be created.

When creating a slave KDC, *admprincipal* is used to authenticate as the administrative principal.

If you omit `-a`, the suggested default administrative principal name is the output of `logname(1)` appended by `/admin`.

`-e` *enctype*
Specifies the encryption type to be used when creating the key for the master key, which is used to encrypt all principal keys in the database. The set of valid encryption types used here are described in `krb5.conf(4)` under the `permitted_enctypes` option. Note that the encryption type specified here must be supported on all KDCs or else they will not be able to decrypt any of the principal keys. Solaris 9 and earlier releases support only the `des-cbc-crc` encryption type for the master key. Therefore, if any of the master or slave KDCs are of these older releases, then `-e des-cbc-crc` would need to be specified on all KDCs configured with `kdcmgr`.

The default encryption type is `aes256-cts-hmac-sha1-96`.

`-h`
Displays usage information for `kdcmgr`.

`-p` *pwfile*
Provides the location of the password file that contains the password used to create the administrative principal and/or master key.

Warning: This option should be used with great care. Make sure that this *pwfile* is accessible only by a privileged user and on a local file system. Once the KDC has been configured, you should remove *pwfile*.

`-r` *realm*
Set the default realm for this server.

If the `-r` option is not specified, `kdcmgr` attempts to obtain the machine's local domain name by submitting the canonical form of the machine's host name to DNS and using the return value to derive the domain name. If successful, the domain name is converted to uppercase and proposed as the default realm name.

Subcommands The following subcommands are supported:

`create` [*master*]
`create` [`-m` *masterkdc*] slave
Creates a KDC. If no option is specified, an attempt to create a master KDC is made.

`create` [*master*]
Create a master KDC. Upon successful configuration the `krb5kdc(1M)` and `kadmind(1M)` are enabled on the machine.

`create` [`-m` *masterkdc*] slave
Configures a slave KDC. After configuration, the `krb5kdc(1M)` and `kpropd(1M)` services are enabled on the machine.

masterkdc specifies the master KDC to authenticate and with which to perform administrative tasks. If the `-m` option is not specified, you are prompted for a master KDC host name.

`destroy`
Remove all Kerberos configuration and database files associated with the KDC server. A confirmation is required before these files are deleted.

`status`
Determines the role of the KDC, master or slave, and outputs this and the state of such associated processes as:

- `krb5kdc(1M)`
- `kadmind(1M)`
- `kpropd(1M)`

The subcommand also displays information on incremental propagation if the configuration has this feature enabled, as well as any issues with dependent files.

Examples

EXAMPLE 1 Setting up a Master KDC

The following command configures a master KDC with the administrative principal `user1/admin` and with the realm name `EXAMPLE.COM`:

```
$ kdcmgr -a user1/admin -r EXAMPLE.COM create
```

Note that a password will be required to assign to the newly created **`user1/admin`** principal. The password for the master key will also need to be provided.

EXAMPLE 2 Setting up a Slave KDC

The following command configures a slave KDC, authenticates with the administrative principal `user1/admin`, specifies `kdc1` as the master, and uses the `EXAMPLE.COM` realm name:

```
$ kdcmgr -a user1/admin -r EXAMPLE.COM create -m kdc1 slave
```

Note that you must enter the correct password for `user1/admin` and that the master KDC must already have been created before entering this command. The correct password for the master key is also required.

Files

`/etc/krb5/krb5.conf`
Main Kerberos configuration file.

`/etc/krb5/kdc.conf`
KDC configuration, used by both master and slave servers.

`/etc/krb5/krb5.keytab`
Default location of the local host's service keys.

`/etc/krb5/kadm5.acl`
Kerberos administrative access control list (ACL).

`/etc/krb5/kadm5.keytab`
Service keys specific to `kadmind(1M)`.

`/var/krb5/principal`
Kerberos principal database.

`/var/krb5/principal.kadm5`
Kerberos policy database.

`/etc/krb5/kpropd.acl`
Used by slaves to indicate from which server to receive updates.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/security/kerberos-5
Interface Stability	See below

The command line interface (CLI) is Uncommitted. The CLI output is Not an Interface.

See Also `logname(1)`, `ssh(1)`, `kadmin(1M)`, `kadmind(1M)`, `kdb5_util(1M)`, `kdb5_ldap_util(1M)`, `kpropd(1M)`, `krb5kdc(1M)`, `ping(1M)`, `svcadm(1M)`, `kdc.conf(4)`, `krb5.conf(4)`, `attributes(5)`

Name kernel – UNIX system executable file containing basic operating system services

Synopsis

```
kernel-name [-asrvx] [-m smf_options] [-i altinit]
```

Description The operating system image, or kernel, is the collection of software comprising the image files (`unix` and `genunix`) and the modules loaded at any instant in time. The system will not function without a kernel to control it.

The kernel is loaded by the `boot(1M)` command in a machine-specific way. The kernel may be loaded from disk, CD-ROM, or DVD (`diskfull boot`) or over the network (`diskless boot`). In either case, the directories under `/platform` and `/kernel` must be readable and must contain executable code which is able to perform the required kernel service. If the `-a` flag is given, the user is able to supply different pathnames for the default locations of the kernel and modules. See `boot(1M)` for more information on loading a specific kernel.

The `moddir` variable contains a list of module directories separated by whitespace. moddir can be set in the `/etc/system` file. The minimal default is:

```
/platform/platform-name/kernel /kernel /usr/kernel
```

This default can be supplemented by a specific platform. It is common for many SPARC systems to override the default path with:

```
/platform/platform-name/kernel:/platform/hardware-class-name\
     /kernel:/kernel:/usr/kernel
```

where *platform-name* can be found using the `-i` option of `uname(1)`, and *hardware-class-name* can be found using the `-m` option of `uname(1)`.

The kernel configuration can be controlled using the `/etc/system` file (see `system(4)`).

`genunix` is the platform-independent component of the base kernel.

Options The following options are supported:

`-a`
: Asks the user for configuration information, such as where to find the system file, where to mount `root`, and even override the name of the kernel itself. Default responses will be contained in square brackets ([ ]), and the user may simply enter RETURN to use the default response (note that RETURN is labeled ENTER on some keyboards). To help repair a damaged `/etc/system` file, enter **`/dev/null`** at the prompt that asks for the pathname of the system configuration file. See `system(4)`.

`-i` *altinit*
: Select an alternative executable to be the primordial process. *altinit* must be a valid path to an executable. The default primordial process is `init(1M)`.

`-m` *smf_options*
: The *smf_options* include two categories of options to control booting behavior of the service management facility: recovery options and messages options.

Message options determine the type and amount of messages that smf(5) displays during boot. Service options determine the services which are used to boot the system.

Recovery options

debug
Prints standard per-service output and all `svc.startd` messages to log.

milestone=[milestone]
Boot with some SMF services temporarily disabled, as indicated by *milestone*. *milestone* can be “none”, “single-user”, “multi-user”, “multi-user-server”, or “all”. See the `milestone` subcommand of svcadm(1M).

Messages options

quiet
Prints standard per-service output and error messages requiring administrative intervention.

verbose
Prints standard per-service output with more informational messages.

`-r`
Reconfiguration boot. The system will probe all attached hardware devices and configure the logical namespace in `/dev`. See add_drv(1M) and rem_drv(1M) for additional information about maintaining device drivers.

`-s`
Boots only to init level 's'. See init(1M).

`-v`
Boots with verbose messages enabled. If this flag is not given, the messages are still printed, but the output is directed to the system logfile. See syslogd(1M).

`-x`
Does not boot in clustered mode. This option only has an effect when a version of Sun Cluster software that supports this option has been installed.

Examples See boot(1M) for examples and instructions on how to boot.

Files `/kernel`
Contains kernel components common to all platforms within a particular instruction set that are needed for booting the system.

`/platform/`*platform-name*`/kernel`
The platform-specific kernel components.

`/platform/`*hardware-class-name*`/kernel`
The kernel components specific to this hardware class.

`/usr/kernel`
Contains kernel components common to all platforms within a particular instruction set.

The directories in this section can potentially contain the following subdirectories:

drv
Loadable device drivers

exec
The modules that execute programs stored in various file formats.

fs
File system modules

misc
Miscellaneous system-related modules

sched
Operating system schedulers

strmod
System V STREAMS loadable modules

sys
Loadable system calls

SPARC cpu
Processor specific modules

tod
Time-Of-Day hardware interface modules

As only 64-bit SPARC platforms are supported, all SPARC executable modules are contained within `sparcv9` directories in the directories listed above.

x86 mach
x86 hardware support

Modules comprising the 32-bit x86 kernel are contained in the above directories, with the 64-bit x86 kernel components contained within `amd64` subdirectories.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/library/processor, system/library/processorx

See Also

`kmdb(1)`, `uname(1)`, `isainfo(1)`, `add_drv(1M)`, `boot(1M)`, `init(1M)`, `rem_drv(1M)`, `savecore(1M)`, `svc.startd(1M)`, `svcadm(1M)`, `syslogd(1M)`, `system(4)`, `attributes(5)`, `smf(5)`, `devfs(7FS)`

SPARC Only `monitor(1M)`

Diagnostics The kernel gives various warnings and error messages. If the kernel detects an unrecoverable fault, it will panic or halt.

Notes Reconfiguration boot will, by design, not remove `/dev` entries for some classes of devices that have been physically removed from the system.

Name keyserv – server for storing private encryption keys

Synopsis

```
keyserv [-c] [-d | -e] [-D] [-n] [-s sizespec]
```

Description `keyserv` is a daemon that is used for storing the private encryption keys of each user logged into the system. These encryption keys are used for accessing secure network services such as secure NFS.

Normally, root's key is read from the file `/etc/.rootkey` when the daemon is started. This is useful during power-fail reboots when no one is around to type a password.

`keyserv` does not start up if the system does not have a secure `rpc` domain configured. Set up the domain name by using the `/usr/bin/domainname` command. Usually the `svc:/system/identity:domain` service reads the domain from `/etc/defaultdomain`. Invoking the `domainname` command without arguments tells you if you have a domain set up.

The `/etc/default/keyserv` file contains the following default parameter settings. See Files.

`ENABLE_NOBODY_KEYS` Specifies whether default keys for `nobody` are used. `ENABLE_NOBODY_KEYS=NO` is equivalent to the `-d` command-line option. The default value for `ENABLE_NOBODY_KEYS` is `YES`.

Options The following options are supported:

`-c` Do not use disk caches. This option overrides any `-s` option.

`-D` Run in debugging mode and log all requests to `keyserv`.

`-d` Disable the use of default keys for `nobody`. See Files.

`-e` Enable the use of default keys for `nobody`. This is the default behavior. See Files.

`-n` Root's secret key is not read from `/etc/.rootkey`. Instead, `keyserv` prompts the user for the password to decrypt root's key stored in the `publickey` database and then stores the decrypted key in `/etc/.rootkey` for future use. This option is useful if the `/etc/.rootkey` file ever gets out of date or corrupted.

`-s` *sizespec* Specify the size of the extended Diffie-Hellman common key disk caches. The *sizespec* can be one of the following forms:

mechtype`=size` `size` is an integer specifying the maximum number of entries in the cache, or an integer immediately followed by the letter *M*, denoting the maximum size in MB.

`size` This form of *sizespec* applies to all caches.

Note that the `des` mechanism, `AUTH_DES`, does not use a disk cache.

Files `/etc/.rootkey`

`/etc/default/keyserv` Contains default settings. You can use command-line options to override these settings.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also `keylogin(1)`, `svcs(1)`, `keylogout(1)`, `svcadm(1M)`, `publickey(4)`, `attributes(5)`, `smf(5)`

Notes The `keyserv` service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/network/rpc/keyserv:default
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. The service's status can be queried using the `svcs(1)` command.

Name killall – kill all active processes

Synopsis

```
/usr/sbin/killall [signal]
```

Description `killall` is used by `shutdown(1M)` to kill all active processes not directly related to the shutdown procedure.

`killall` terminates all processes with open files so that the mounted file systems will be unbusied and can be unmounted.

`killall` sends *signal* (see `kill(1)`) to the active processes. If no *signal* is specified, a default of `15` is used.

The `killall` command can be run only by the super-user.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also `kill(1)`, `ps(1)`, `fuser(1M)`, `shutdown(1M)`, `signal(3C)`, `attributes(5)`

Name kmscfg – configure the PKCS#11 KMS provider

Synopsis

```
kmscfg
kmscfg -p[rofile] Profile_Name
kmscfg -a[gent] Agent_ID
kmscfg -i[paddr] Agent_Address
kmscfg -t[imeout] Transaction_Timeout
kmscfg -f[ailover] Failover_Limit
kmscfg -d[iscovery] Discovery_Freq
```

Description The `kmscfg` command is used to initialize the PKCS#11 KMS provider (`pkcs11_kms`) for use with the Solaris Cryptographic Framework. In order for the KMS provider to communicate with the Oracle Key Manager (OKM), it must have some configuration information available. This configuration data contains information such as the name of the profile to be used, the name of the OKM Agent, the IP address of an OKM appliance (KMA) and some other parameters (see `SYNOPSIS`).

By default, `kmscfg` stores the configuration information in `/var/user/$USERNAME/kms`. This directory will be created if it is not already present. If the configuration is already detected, the user will be given the option to override the existing data. The default location can be overriden by using the `KMSTOKEN_DIR` environment variable, which must be set prior to invoking `kmscfg`.

Prior to running `kmscfg`, the OKM administrator must have performed the required initialization and configuration steps on the appliance itself to setup the individual Profiles and Agents that PKCS11 KMS consumers will use. The instructions for configuring these profiles are available in the *Oracle Key Manager Administration Guide* on the Oracle website (`http://docs.oracle.com`).

Once the administrator has configured the KMA, the necessary identification information (profile name, agent ID, IP address) must be provided to be able to run `kmscfg` and initialize the provider on the Oracle Solaris client.

Options The options listed below are supported. Note that, if the profile, agent id, or KMA address are not specified on the command line, `kmscfg` prompts you to provide these items.

`-a` *Agent_ID*
The user agent ID as configured on the OKM to be used for the KMS token being configured. It is not unusual for the Profile and Agent ID to be the same, for example, `MyAgent`.

`-d` *Discovery_Freq*
Frequency in seconds with which the client will try to discover the availability of other KMAs in an OKM cluster. If not specified, *Discovery_Freq* defaults to 600 seconds (10 minutes).

`-f` *Failover_Limit*
The number of times communications to the KMA can fail before the client gives up. If not specified, *Failover_Limit* defaults to 3.

`-i` *Agent_Addr*
Address of the KMA. This can be an IPv4 address (*xxx.xxx.xxx.xxx*) or an IPv6 address. A fully qualified host name can also be used, as long as that name can be resolved by the name service configured on the client. If an OKM cluster is being used, the address of any member of the cluster can be specified.

`-p` *Profile_Name*
A name for the profile to be used for the KMS token being configured. Typically, the profile name and the Agent ID will be the same, though they are not required to be.

`-t` *Transaction_Timeout*
Timeout period for individual KMS commands, in seconds. If not specified, this value defaults to 10.

Exit Status After completing the requested operation, `kmscfg` exits with one of the following status values.

`0`
Successful termination.

`1`
Failure. The requested operation could not be completed.

Files `/var/user/$USERNAME/kms`
Default KMS token configuration directory.

`${KMSTOKEN_DIR}`
Alternate KMS token configuration directory.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	/system/library/security/crypto/pkcs11_kms
Interface Stability	Volatile

See Also `pktool(1)`, `attributes(5)`, `pkcs11_kms(5)`

Oracle Key Manager Administration Guide (`http://docs.oracle.com`)

Notes PKCS#11 clients require Oracle Key Manager Software Version 2.4 be installed on the OKM.

If PKCS#11 clients will use the same Agent ID from multiple systems, that agent should be created without the “One Time Passphrase” flag set. This option will not be available in OKM clusters with some members running versions of the OKM software prior to 2.4. Please refer to the *OKM Administration Guide* for assistance in creating Agents.

OKM Agents must have a Default Key Group assigned prior to being used to create keys with a PKCS#11 client. If a Default Key Group is not assigned to the Agent, operations will fail with a `CKR_PIN_INCORRECT` error. Please refer to the *OKM Administration Guide* for assistance in assigning key groups to agents.

Name kprop – Kerberos database propagation program

Synopsis

```
/usr/lib/krb5/kprop [-d] [-f file] [-p port-number]
     [-r realm] [-s keytab] [host]
```

Description `kprop` is a command-line utility used for propagating a Kerberos database from a master KDC to a slave KDC. This command must be run on the master KDC. See the *Solaris System Administration Guide, Vol. 6* on how to set up periodic propagation between the master KDC and slave KDCs.

To propagate a Kerberos database, the following conditions must be met:

- The slave KDCs must have an `/etc/krb5/kpropd.acl` file that contains the principals for the master KDC and all the slave KDCs.
- A keytab containing a host principal entry must exist on each slave KDC.
- The database to be propagated must be dumped to a file using `kdb5_util(1M)`.

Options The following options are supported:

`-d`	Enable debug mode. Default is debug mode disabled.
`-f` *file*	File to be sent to the slave KDC. Default is the `/var/krb5/slave_datatrans` file.
`-p` *port-number*	Propagate *port-number*. Default is port `754`.
`-r` *realm*	Realm where propagation will occur. Default *realm* is the local realm.
`-s` *keytab*	Location of the keytab. Default location is `/etc/krb5/krb5.keytab`.

Operands The following operands are supported:

host Name of the slave KDC.

Examples **EXAMPLE 1** Propagating the Kerberos Database

The following example propagates the Kerberos database from the `/tmp/slave_data` file to the slave KDC `london`. The machine `london` must have a host principal keytab entry and the `kpropd.acl` file must contain an entry for the all the KDCs.

```
# kprop -f /tmp/slave_data london
```

Files

`/etc/krb5/kpropd.acl`	List of principals of all the KDCs; resides on each slave KDC.
`/etc/krb5/krb5.keytab`	Keytab for Kerberos clients.
`/var/krb5/slave_datatrans`	Kerberos database propagated to the KDC slaves.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/security/kerberos-5

See Also kpasswd(1), svcs(1), gkadmin(1M), inetadm(1M), inetd(1M), kadmind(1M), kadmin.local(1M), kdb5_util(1M), svcadm(1M), kadm5.acl(4), kdc.conf(4), attributes(5), kerberos(5), smf(5)

Oracle Solaris 11.1 Administration: Security Services

Name kpropd – Kerberos propagation daemon for slave KDCs

Synopsis

```
/usr/lib/krb5/kpropd [-d] [-f temp_dbfile] [-F dbfile]
     [-p kdb_util] [-P port_number] [-r realm]
     [-s srv_tabfile] [-S] [-a acl_file]
```

Description The `kpropd` command runs on the slave KDC server. It listens for update requests made by `kprop(1M)` from the master KDC and periodically requests incremental updates from the master KDC.

When the slave receives a kprop request from the master, `kpropd` copies principal data to a temporary text file. Next, `kpropd` invokes `kdb5_util(1M)` (unless a different database utility is selected) to load the text file in database format.

When the slave periodically requests incremental updates, `kpropd` update its `principal.ulog` file with any updates from the master. `kproplog(1M)` can be used to view a summary of the update entry log on the slave KDC.

kpropd is not configured for incremental database propagation by default. These settings can be changed in the `kdc.conf(4)` file:

`sunw_dbprop_enable = [true | false]`	Enables or disables incremental database propagation. Default is `false`.
`sunw_dbprop_slave_poll = N[s, m, h]`	Specifies how often the slave KDC polls for any updates that the master might have. Default is `2m` (two minutes).

The `kiprop/`<*hostname*>`@`<*REALM*> principal must exist in the slave's `keytab` file to enable the master to authenticate incremental propagation requests from the slave. In this syntax, <*hostname*> is the slave KDC's host name and <*REALM*> is the realm in which the slave KDC resides.

Options The following options are supported:

`-d`	Enable debug mode. Default is debug mode disabled.
`-f` *temp_dbfile*	The location of the slave's temporary principal database file. Default is `/var/krb5/from_master`.
`-F` *dbfile*	The location of the slave's principal database file. Default is `/var/krb5/principal`.
`-p` *kdb_util*	The location of the Kerberos database utility used for loading principal databases. Default is `/usr/sbin/kdb5_util`.
`-P` *port_number*	Specifies the port number on which `kpropd` will listen. Default is 754 (service name: `krb5_prop`).
`-r` *realm*	Specifies from which Kerberos realm kpropd will receive information. Default is specified in `/etc/krb5/krb5.conf`.

`-s` *srv_tabfile*	The location of the service table file used to authenticate the `kpropd` daemon.
`-S`	Run the daemon in standalone mode, instead of having `inetd` listen for requests. Default is non-standalone mode.
`-a` *acl_file*	The location of the `kpropd`'s access control list to verify if this server can run the `kpropd` daemon. The file contains a list of principal name(s) that will be receiving updates. Default is `/etc/krb5/kpropd.acl`.

Files

`/var/krb5/principal`	Kerberos principal database.
`/var/krb5/principal.ulog`	The update log file.
`/etc/krb5/kdc.conf`	KDC configuration information.
`/etc/krb5/kpropd.acl`	List of principals of all the KDCs; resides on each slave KDC.
`/var/krb5/from_master`	Temporary file used by kpropd before loading this to the principal database.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/security/kerberos-5
Interface Stability	Committed

See Also `kdb5_util(1M)`, `kprop(1M)`, `kproplog(1M)`, `kdc.conf(4)`, `krb5.conf(4)`, `attributes(5)`, `kerberos(5)`

Notes The `kprop` service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/network/security/krb5_prop:default
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. Responsibility for initiating and restarting this service is delegated to `inetd(1M)`. Use `inetadm(1M)` to make configuration changes and to view configuration information for this service. The service's status can be queried using the `svcs(1)` command.

Name kproplog – display the contents of the Kerberos principal update log

Synopsis

```
/usr/sbin/kproplog [-h | -e num]
```

Description The `kproplog` displays the contents of the Kerberos principal update log to standard output. This command can be used to keep track of the incremental updates to the principal database, which is enabled by default. The `/var/krb5/principal.ulog` file contains the update log maintained by the `kadmind(1M)` process on the master KDC server and the `kpropd(1M)` process on the slave KDC servers. When updates occur, they are logged to this file. Subsequently any KDC slave configured for incremental updates will request the current data from the master KDC and update their `principal.ulog` file with any updates returned.

The `kproplog` command can only be run on a KDC server by someone with privileges comparable to the superuser. It will display update entries for that server only.

If no options are specified, the summary of the update log is displayed. If invoked on the master, all of the update entries are also displayed. When invoked on a slave KDC server, only a summary of the updates are displayed, which includes the serial number of the last update received and the associated time stamp of the last update.

Options The following options are supported:

`-h` Display a summary of the update log. This information includes the database version number, state of the database, the number of updates in the log, the time stamp of the first and last update, and the version number of the first and last update entry.

`-e` *num* Display the last *num* update entries in the log. This is useful when debugging synchronization between KDC servers.

`-v` Display individual attributes per update. If more than one `-v` is specified then very verbose output is displayed. An example of the output generated for one entry:

```
Update Entry
    Update serial # : 4
    Update operation : Add
    Update principal : test@EXAMPLE.COM
    Update size : 424
    Update committed : True
    Update time stamp : Fri Feb 20 23:37:42 2004
    Attributes changed : 6
          Principal
          Key data
          Password last changed
          Modifying principal
          Modification time
          TL data
```

Files `/var/krb5/principal.ulog` The update log file for incremental propagation.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/security/kerberos-5
Interface Stability	Committed

See Also `kpasswd(1)`, `gkadmin(1M)`, `kadmin(1M)`, `kadmind(1M)`, `kdb5_util(1M)`, `kprop(1M)`, `kpropd(1M)`, `kadm5.acl(4)`, `kdc.conf(4)`, `attributes(5)`, `kerberos(5)`

Name krb5kdc – KDC daemon

Synopsis

```
/usr/lib/krb5/krb5kdc [-d dbpath] [-r realm] [-m]
     [-k masterenctype] [-M masterkeyname]
     [-p port] [-n] [-x db_args]  [-P pid_file]
```

Description `krb5kdc` is the daemon that runs on the master and slave KDCs to process the Kerberos tickets. For Kerberos to function properly, `krb5kdc` must be running on at least one KDC that the Kerberos clients can access. Prior to running `krb5kdc`, you must initialize the Kerberos database using `kdb5_util`(1M). See the *Oracle Solaris 11.1 Administration: Security Services* for information regarding how to set up KDCs and initialize the Kerberos database.

Options The following options are supported:

`-d` *dbpath*
Specify the path to the database; default value is `/var/krb5`.

`-k` *masterenctype*
Specify the encryption type for encrypting the database. The default value is `des-cbc-crc`. `des3-cbc-sha1`, `arcfour-hmac-md5`, `arcfour-hmac-md5-exp`, `aes128-cts-hmac-sha1-96`, and `aes256-cts-hmac-sha1-96` are also valid.

`-m`
Specify that the master key for the database is to be entered manually.

`-M` *masterkeyname*
Specify the principal to retrieve the master Key for the database.

`-n`
Specify that `krb5kdc` should not detach from the terminal.

`-p` *port*
Specify the port that will be used by the KDC to listen for incoming requests.

`-P` *pid_file*
Tells the KDC to write its PID (followed by a newline) into *pid_file* after it starts up. This can be used to identify whether the KDC is still running and to allow `init` scripts to stop the correct process.

`-r` *realm*
Specify the realm name; default is the local realm name.

`-x` *db_args*
Pass database-specific arguments to `kadmin`. Supported arguments are for the LDAP plug-in. These arguments are:

`binddn=`*binddn*
Specifies the DN of the object used by the KDC server to bind to the LDAP server. This object should have the rights to read the realm container, principal container and the subtree that is referenced by the realm. Overrides the `ldap_kdc_dn` parameter setting in `krb5.conf`(4).

`bindpwd=`*bindpwd*
Specifies the password for the above-mentioned *binddn*. It is recommended not to use this option. Instead, the password can be stashed using the `stashsrvpw` command of `kdb5_ldap_util(1M)`.

`nconns=`*num*
Specifies the number of connections to be maintained per LDAP server.

`host=`*ldapuri*
Specifies, by an LDAP URI, the LDAP server to which to connect.

Files

`/var/krb5/principal.db`
Kerberos principal database.

`/var/krb5/principal.kadm5`
Kerberos administrative database. This file contains policy information.

`/var/krb5/principal.kadm5.lock`
Kerberos administrative database lock file. This file works backwards from most other lock files (that is, `kadmin` will exit with an error if this file does *not* exist).

`/etc/krb5/kdc.conf`
KDC configuration file. This file is read at startup.

`/etc/krb5/kpropd.acl`
File that defines the access control list for propagating the Kerberos database using `kprop`.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/security/kerberos-5

See Also

`kill(1)`, `kpasswd(1)`, `gkadmin(1M)`, `kadmind(1M)`, `kadmin.local(1M)`, `kdb5_util(1M)`, `kdb5_ldap_util(1M)`, `logadm(1M)`, `krb5.conf(4)`, `attributes(5)`, `krb5envvar(5)`, `kerberos(5)`,

Oracle Solaris 11.1 Administration: Security Services

Notes

The following signal has the specified effect when sent to the server process using the `kill(1)`command:

`SIGHUP`
`krb5kdc` closes and re-opens log files that it directly opens. This can be useful for external log-rotation utilities such as `logadm(1M)`. If this method is used for log file rotation, set the `krb5.conf(4)` `kdc_rotate` period relation to `never`.

Name ksslcfg – enable and configure SMF instance of Kernel SSL

Synopsis

```
ksslcfg create -f pkcs11 -T token_label -C certificate_label
     [-d softtoken_directory]
     -p password_file [-u username]
     [-h ca_certchain_file] [-c ciphersuites]
     [-t ssl_session_cache_timeout]
     [-z ssl_session_cache_size] [-v] -x proxy_port [host] ssl_port

ksslcfg create -f pkcs12 -i cert_and_key_pk12file
     -p password_file [-u username]
     [-c ciphersuites] [-t ssl_session_cache_timeout]
     [-z ssl_session_cache_size] [-v] -x proxy_port [host] ssl_port

ksslcfg create -f pem -i cert_and_key_pemfile
     -p password_file [-u username]
     [-c ciphersuites] [-t ssl_session_cache_timeout]
     [-z ssl_session_cache_size] [-v] -x proxy_port [host] ssl_port

ksslcfg delete [-v] [host] ssl_port

ksslcfg -V

ksslcfg -?
```

Description `ksslcfg` manages `smf(5)` instances for the Kernel SSL proxy module. An SSL-enabled web server can use the services of its Kernel SSL proxy to improve the performance of the HTTPS packets processing. It does so by creating an instance of the Kernel SSL service, specifying the SSL proxy port and parameters, and by listening on the proxy port.

The `create` subcommand creates an instance and enables the service for the given address and SSL port.

The `delete` subcommand disables the service for the given address and port, if it is enabled, and deletes the instance from the SMF repository.

`ksslcfg` can be run as root or by other users assigned to the Network Security profile. See `rbac(5)` and `user_attr(4)`.

After `ksslcfg` successfully configures the service in the kernel, the proxy application must be started, or restarted if it is already running.

You must run `ksslcfg` to configure your Kernel SSL proxy before you start your application.

`ksslcfg` allows you to specify an *ssl_port* operand, described under OPERANDS, and, with the `-x` option, a *proxy_port* value. When specified for use with the Kernel SSL proxy, these values cannot also be configured for the Solaris Network Cache and Acceleration (NCA) feature.

The Fault Managed Resource Identifier (FMRI) for the kernel SSL proxy instances is `svc://network/ssl/proxy`. `ksslcfg` creates an instance of that service unique to the

combination of host and SSL port. Instance FMRIs for particular proxy entries can be found with svcs(1) and used for dependencies of other services. The state of the service instance tracks in-kernel configuration only. It does not reflect the presence or state of the application listening on the proxy port.

Options The following options are supported:

`-c` *ciphersuites*
Set of ciphers a client is allowed to negotiate in a sorted order. The supported SSLv3 and TLS ciphers are listed below. Note that the names are case-insensitive.

```
rsa_rc4_128_sha
rsa_rc4_128_md5
rsa_aes_256_cbc_sha
rsa_aes_128_cbc_sha
rsa_3des_ede_cbc_sha
rsa_des_cbc_sha
```

`-f` *key_format*
Uses the certificate/key format specified in *key_format*. The supported options are `pkcs11`, `pkcs12`, and `pem`.

`-i` *key_and_certificate_file*
When `pkcs12` or `pem` is specified with the `-f` option, reads a key and a certificate of the web server from *key_and_certificate_file*. This file can also contain any intermediate CA certificates that form the certificate chain to the root CA for the server certificate. These certificates must follow the server certificate in the file and the order must be bottom up: lowest level CA certificate followed by the next higher level CA certificate, and so on.

`-C` *certificate_label*
PKCS#11 can store multiple certificates in single token. This option enables you to specify a single certificate, identified by *certificate_label*. This label must match the `CKA_LABEL` on the certificate object in the token specified by `-T`. This option is to be used only with `-f pkcs11`.

`-d` *softtoken_directory*
This option is applicable only with the `pkcs11` key format, when the token label is the Sun Software PKCS#11 softtoken. Use this option to override the default location of the PKCS#11 softtoken directory (`$HOME/.sunw`). See pkcs11_softtoken(5).

`-h` *ca_certchain_file*
When `pkcs11` is specified with the `-f` option, reads a set of intermediate CA certificates that form the certificate chain to the root CA for the server certificate (specified with the `-C` option), from *ca_certchain_file*. The file must be in PEM format.

`-p` *password_file*
Obtains the password used to encrypt the private key from *password_file*. When using the `pkcs11` option (see `-f`, above), the password is used to authenticate the user to the PKCS #11 token.

`-t` *ssl_session_cache_timeout*
The timeout value, in seconds, for an SSL session. It corresponds to `SSL3SessionTimeout` of the Sun ONE web server configuration or `SSLSessionCacheTimeout` of `mod_ssl`.

`-T` *token_label*
When `pkcs11` is specified with `-f`, uses the PKCS#11 token specified in *token_label*. Use `cryptoadm list -v` to display all PKCS#11 tokens available.

`-u` *username*
The username of the user who owns the password file. If omitted, the system will try to read the password file as root.

`-v`
Verbose mode.

`-V`
Displays the version.

`-x` *proxy_port*
The SSL proxy port. The port number is designated exclusively for clear-text HTTP communication between the web server and the kernel SSL proxy module. No external HTTP packets are delivered to this port.

`-z` *ssl_session_cache_size*
The maximum number of SSL sessions that can be cached. It corresponds to `SSLCacheEntries` of the Sun ONE web server configuration. When this option is not specified, the default is 5000 entries.

`-?`
Displays the usage of the command.

Operands [*host*] [*ssl_port*]
The address and the port of the web server for which the kernel SSL entry is created. If *host* is omitted, the entry will be used for all requests that arrived at the *ssl_port*, regardless of the destination address. Both a host name and an IP address are acceptable forms for *host*. *ssl_port* is required. Typically, this has a value of 443.

Examples **EXAMPLE 1** Create and Enable a Kernel SSL Instance

The following command creates and enables a Kernel SSL instance using a certificate and a key in PKCS#11 format.

```
# ksslcfg create -f pkcs11 -T "Sun Software PKCS#11 softtoken"  \
-C "Server-Cert" -p /some/directory/password -u webservd \
-x 8080 www.mysite.com 443

% svcs svc:/network/ssl/proxy
STATE          STIME    FMRI
online         Sep_27   svc:/network/ssl/proxy:kssl-www-mysite-com-443
```

EXAMPLE 2 Create and Enable a Default Instance for All Addresses

The following command creates and enables a default instance for all addresses from a certificate and key in a `pkcs#12` file.

```
# ksslcfg create -x 8888 -f pkcs12 -i /some/directory/keypair.p12 \
    -p /some/directory/password -u webservd 443
```

EXAMPLE 3 Create and Enable an Instance with Specific Cipher Suites

The following command creates and enables an instance with specific cipher suites.

```
# ksslcfg create -x 8080 -f pem \
-i /some/directory/keypair.pem -p /some/directory/password \
-c "rsa_rc4_128_md5,rsa_rc4_128_sha" \
209.249.116.195 443
```

EXAMPLE 4 Disable and Delete an Instance

The following command disables and deletes an instance.

```
# ksslcfg delete www.mysite.com 443
```

EXAMPLE 5 Establishing Dependency of Proxy Application

The sequence of commands shown below establishes a dependency of a proxy application on a KSSL instance. Note that he proxy application should only be started after the SSL kernel proxy instance has been started.

The following commands establish the dependency of an Apache 2.2 web server. KSSL has been configured to listen on SSL port 443 and a wildcard address.

```
# svccfg -s svc:/network/http:apache22
svc:/network/http:apache22> addpg kssl dependency
svc:/network/http:apache22> setprop kssl/entities = fmri:svc:/network/\
ssl/proxy:kssl-INADDR_ANY-443
svc:/network/http:apache22> setprop kssl/grouping = astring: require_all
svc:/network/http:apache22> setprop kssl/restart_on = astring: refresh
svc:/network/http:apache22> setprop kssl/type = astring: service
svc:/network/http:apache22> end
```

Following these commands, enable the web server:

```
# svcadm enable svc:/network/http:apache22
```

If the web server was already running, restart it:

```
# svcadm refresh svc:/network/http:apache22
# svcadm restart svc:/network/http:apache22
```

Exit Status

`0`
Successful completion.

`>0`
An error occurred.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os
Interface Stability	See below.

Command line options and the utility name are Committed. The command output, the FMRI service name (`svc://network/ssl/proxy`), and the FMRI instance's name format are Uncommitted

See Also

`svcprop(1)`, `svcs(1)`, `cryptoadm(1M)`, `svcadm(1M)`, `svccfg(1M)`, `user_attr(4)`, `attributes(5)`, `kssl(5)`, `pkcs11_softtoken(5)`, `rbac(5)`, `smf(5)`, `nca(7d)`

Chown, P., *Advanced Encryption Standard (AES) Ciphersuites for Transport Layer Security (TLS)*, RFC 3268, June 2002.

Notes

`ksslcfg create` without an host argument creates an `INADDR_ANY` `smf` instance. `ksslcfg delete` without an host argument deletes only the `INADDR_ANY` instance. `ksslcfg delete` needs a host argument to delete any non-`INADDR_ANY` instance.

On a system with `zones(5)` installed, the `ksslcfg` command can be used only in the global zone at this time.

Name kstat – display kernel statistics

Synopsis

```
kstat [-lpq] [-T u | d ] [-c class] [-m module]
     [-i instance] [-n name] [-s statistic]
     [interval [count]]

kstat [-lpq] [-T u | d ] [-c class]
     [module:instance:name:statistic]...
     [interval [count]]
```

Description The `kstat` utility examines the available kernel statistics, or kstats, on the system and reports those statistics which match the criteria specified on the command line. Each matching statistic is printed with its module, instance, and name fields, as well as its actual value.

Kernel statistics may be published by various kernel subsystems, such as drivers or loadable modules; each kstat has a module field that denotes its publisher. Since each module might have countable entities (such as multiple disks associated with the `sd`(7D) driver) for which it wishes to report statistics, the kstat also has an instance field to index the statistics for each entity; kstat instances are numbered starting from zero. Finally, the kstat is given a name unique within its module.

Each kstat may be a special kstat type, an array of name-value pairs, or raw data. In the name-value case, each reported value is given a label, which we refer to as the statistic. Known raw and special kstats are given statistic labels for each of their values by `kstat`; thus, all published values can be referenced as *module*:*instance*:*name*:*statistic*.

When invoked without any module operands or options, kstat will match all defined statistics on the system. Example invocations are provided below. All times are displayed as fractional seconds since system boot.

Options The tests specified by the following options are logically ANDed, and all matching kstats will be selected. A regular expression containing shell metacharacters must be protected from the shell by enclosing it with the appropriate quotes.

The argument for the `-c`, `-i`, `-m`, `-n`, and `-s` options may be specified as a shell glob pattern, or a Perl regular expression enclosed in '/' characters.

`-c` *class* Displays only kstats that match the specified class. *class* is a kernel-defined string which classifies the “type” of the kstat.

`-i` *instance* Displays only kstats that match the specified instance.

`-l` Lists matching kstat names without displaying values.

`-m` *module* Displays only kstats that match the specified module.

`-n` *name* Displays only kstats that match the specified name.

`-p` Displays output in parseable format. All example output in this document is given in this format. If this option is not specified, `kstat` produces output in a human-readable, table format.

`-q`	Displays no output, but return appropriate exit status for matches against given criteria.
`-s` *statistic*	Displays only kstats that match the specified statistic.
`-T d | u`	Displays a time stamp before each statistics block, either in `date(1)` format (`d`) or as an alphanumeric representation of the value returned by `time(2)` (`u`).

Operands The following operands are supported:

module:*instance*:*name*:*statistic*	Alternate method of specifying module, instance, name, and statistic as described above. Each of the module, instance, name, or statistic specifiers may be a shell glob pattern or a Perl regular expression enclosed by '/' characters. It is possible to use both specifier types within a single operand. Leaving a specifier empty is equivalent to using the '*' glob pattern for that specifier.
interval	The number of seconds between reports.
count	The number of reports to be printed.

Examples In the following examples, all the command lines in a block produce the same output, as shown immediately below. The exact statistics and values will of course vary from machine to machine.

EXAMPLE 1 Using the kstat Command

```
example$ kstat -p -m unix -i 0 -n system_misc -s 'avenrun*'
example$ kstat -p -s 'avenrun*'
example$ kstat -p 'unix:0:system_misc:avenrun*'
example$ kstat -p ':::avenrun*'
example$ kstat -p ':::/^avenrun_\d+min$/'

unix:0:system_misc:avenrun_15min        3
unix:0:system_misc:avenrun_1min 4
unix:0:system_misc:avenrun_5min 2
```

EXAMPLE 2 Using the kstat Command

```
example$ kstat -p -m cpu_stat -s 'intr*'
example$ kstat -p cpu_stat:::/^intr/

cpu_stat:0:cpu_stat0:intr       29682330
cpu_stat:0:cpu_stat0:intrblk    87
cpu_stat:0:cpu_stat0:intrthread 15054222
cpu_stat:1:cpu_stat1:intr       426073
cpu_stat:1:cpu_stat1:intrblk    51
cpu_stat:1:cpu_stat1:intrthread 289668
```

EXAMPLE 2 Using the kstat Command *(Continued)*

```
cpu_stat:2:cpu_stat2:intr       134160
cpu_stat:2:cpu_stat2:intrblk    0
cpu_stat:2:cpu_stat2:intrthread 131
cpu_stat:3:cpu_stat3:intr       196566
cpu_stat:3:cpu_stat3:intrblk    30
cpu_stat:3:cpu_stat3:intrthread 59626
```

EXAMPLE 3 Using the kstat Command

```
example$ kstat -p :::state ':::avenrun*'
example$ kstat -p :::state :::/^avenrun/

cpu_info:0:cpu_info0:state      on-line
cpu_info:1:cpu_info1:state      on-line
cpu_info:2:cpu_info2:state      on-line
cpu_info:3:cpu_info3:state      on-line
unix:0:system_misc:avenrun_15min        4
unix:0:system_misc:avenrun_1min 10
unix:0:system_misc:avenrun_5min 3
```

EXAMPLE 4 Using the kstat Command

```
example$ kstat -p 'unix:0:system_misc:avenrun*' 1 3
unix:0:system_misc:avenrun_15min        15
unix:0:system_misc:avenrun_1min 11
unix:0:system_misc:avenrun_5min 21

unix:0:system_misc:avenrun_15min        15
unix:0:system_misc:avenrun_1min 11
unix:0:system_misc:avenrun_5min 21

unix:0:system_misc:avenrun_15min        15
unix:0:system_misc:avenrun_1min 11
unix:0:system_misc:avenrun_5min 21
```

EXAMPLE 5 Using the kstat Command

```
example$ kstat -p -T d 'unix:0:system_misc:avenrun*' 5 2
Thu Jul 22 19:39:50 1999
unix:0:system_misc:avenrun_15min        12
unix:0:system_misc:avenrun_1min 0
unix:0:system_misc:avenrun_5min 11

Thu Jul 22 19:39:55 1999
unix:0:system_misc:avenrun_15min        12
unix:0:system_misc:avenrun_1min 0
unix:0:system_misc:avenrun_5min 11
```

EXAMPLE 6 Using the kstat Command

```
example$ kstat -p -T u 'unix:0:system_misc:avenrun*'
932668656
unix:0:system_misc:avenrun_15min        14
unix:0:system_misc:avenrun_1min 5
unix:0:system_misc:avenrun_5min 18
```

Exit Status The following exit values are returned:

`0` One or more statistics were matched.

`1` No statistics were matched.

`2` Invalid command line options were specified.

`3` A fatal error occurred.

Files `/dev/kstat` kernel statistics driver

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also `date(1)`, `sh(1)`, `time(2)`, `gmatch(3GEN)`, `kstat(3KSTAT)`, `attributes(5)`, `kstat(7D)`, `sd(7D)`, `kstat(9S)`

Notes If the pattern argument contains glob or Perl RE metacharacters which are also shell metacharacters, it will be necessary to enclose the pattern with appropriate shell quotes.

Name ktkt_warnd – Kerberos warning daemon

Synopsis

```
/usr/lib/krb5/ktkt_warnd
```

Description `ktkt_warnd` is a daemon on Kerberos clients that can warn users when their Kerberos tickets are about to expire or renew the tickets before they expire. It is invoked by `inetd` when a ticket-granting ticket (TGT) is obtained for the first time, such as after using the `kinit` command. `ktkt_warnd` can be configured through per user and system configuration files on the client. In the configuration files, you can specify that you be supplied notice of ticket expiration—through terminal, mail, or `syslog`—or to renew the TGT.

Files

`/etc/krb5/warn.conf`	Kerberos warning configuration file
`/var/user/$USER/krb-warn.conf`	Per-user Kerberos warning configuration file

If the user's configuration file does not exist, `/etc/krb5/warn.conf` will be used.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Committed

See Also `svcs(1)`, `inetadm(1M)`, `inetd(1M)`, `svcadm(1M)`, `warn.conf(4)`, `attributes(5)`, `kerberos(5)`, `smf(5)`

Notes The `ktkt_warnd` service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/network/security/ktkt_warn:default
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. Responsibility for initiating and restarting this service is delegated to `inetd(1M)`. Use `inetadm(1M)` to make configuration changes and to view configuration information for this service. The service's status can be queried using the `svcs(1)` command.

Name labeld – Trusted Extensions label daemon

Synopsis

```
/usr/lib/labeld
```

Description The `label` daemon is a system daemon which is started at boot time when Trusted Extensions is enabled. This daemon is started automatically and should not be invoked directly. It does not constitute a programming interface.

To enable Trusted Extensions, enter the following at the command line:

```
% svcadm enable svc:/system/labeld
```

Because Trusted Extensions affects the initialization and operation of zones, all zones must be removed before enabling Trusted Extensions. After enabling, the system should be rebooted to allow Trusted Extensions to come up properly initialized.

Enabling or disabling Trusted Extensions can only be done by a user or role with the `solaris.smf.manage.labels` authorization. (For example, a user or role that has either the Information Security or Object Label Management Rights Profile, or superuser.)

Other configuration steps are needed before using Trusted Extensions functionality. For more information, see the *Solaris Trusted Extensions Installation and Configuration Guide*.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/trusted
Interface Stability	Uncommitted

See Also `svcs(1)`, `svcadm(1M)`, `syslogd(1M)`, `attributes(5)`, `smf(5)`

Solaris Trusted Extensions Administrator's Procedures

Notes The `labels(5)` service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/system/labeld:default
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. The service's status can be queried using the `svcs(1)` command.

Name labelit – list or provide labels for file systems

Synopsis

```
labelit [-F FSType] [-V] [-o FSType-specific-options] special
     [operands]
```

Description The `labelit` utility is used to write or display labels on unmounted disk file systems.

Options The following options are supported:

`-F` *FSType* Specify the *FSType* on which to operate. The *FSType* should either be specified here or be determinable from `/etc/vfstab` by matching the *special* with an entry in the table. If no matching entry is found, the default file system type specified in `/etc/default/fs` will be used.

`-V` Echo complete command line. This option may be used to verify and validate the command line. Additional information obtained using a `/etc/vfstab` lookup is included in the output. The command is not executed.

`-o` Specify *FSType*-specific options. See the manual page for the `labelit` module specific to the file system type.

Operands The following operands are supported. If no operands are specified, `labelit` will display the value of the labels.

special The disk partition (for example, `/dev/rdsk/c0t3d0s6`). The device may not be on a remote machine.

operands *FSType*-specific operands. Consult the manual page of the *FSType*-specific `labelit` command for detailed descriptions.

Usage See `largefile(5)` for the description of the behavior of `labelit` when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

Exit Status The following exit values are returned:

`0` Write or display of labels was successful.

non-zero An error occurred.

Files

`/etc/vfstab` List of default parameters for each file system.

`/etc/default/fs` Default local file system type. Default values can be set for the following flags in `/etc/default/fs`. For example:

```
LOCAL=ufs
```

`LOCAL` The default partition for a command if no *FSType* is specified.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also `labelit_hsfs(1M)`, `labelit_udfs(1M)`, `labelit_ufs(1M)`, `volcopy(1M)`, `vfstab(4)`, `attributes(5)`, `largefile(5)`

Notes This utility may not be supported for all *FSTypes*.

Name labelit_hsfs – provide and print labels for hsfs file systems

Synopsis

```
/usr/sbin/labelit -F hsfs [generic_options]
     [-o specific_options] special
```

Description `labelit` can be used to provide labels for unmounted CD-ROM images (CD-ROMs may not be labeled, as they are read-only media).

generic_options are options supported by the generic `labelit` command.

If no *specific_options* are specified, `labelit` prints the current value of all label fields.

The *special* name should be the physical disk section (for example, `/dev/dsk/c0d0s6`).

Options

`-o` Use one or more of the following *name=value* pairs separated by commas (with no intervening spaces) to specify values for specific label fields. According to the ISO 9660 specification, only certain sets of characters may be used to fill in these labels. Thus, "d-characters" below refers to the characters 'A' through 'Z', the digits '0' through '9', and the '_' (underscore) character. "a-characters" below refers to 'A' through 'Z', '0' through '9', space, and the following characters: !"%&'()*+,-./:;<=>?_.

`absfile=` Abstract file identifier, d-characters, 37 characters maximum.

`applid=` Application identifier, d-characters, 128 characters maximum.

`bibfile=` Bibliographic file identifier, d-characters, 37 characters maximum.

`copyfile=` Copyright file identifier, d-characters, 128 maximum.

`prepid=` Data preparer identifier, d-characters, 128 maximum.

`pubid=` Publisher identifier, d-characters, 128 maximum.

`sysid=` System identifier, a-characters, 32 maximum.

`volid=` Volume identifier, d-characters, 32 maximum.

`volsetid=` Volume set identifier, d-characters, 128 maximum.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also `labelit(1M)`, `volcopy(1M)`, `attributes(5)`

Name labelit_udfs – provide and print labels for udf file systems

Synopsis

```
labelit -F  udfs [generic_options] [-o specific_options] special
     [fsname  volume]
```

Description The `labelit` command writes labels on an unmounted disk that contains a universal disk file (udf) system. These labels can be used to identify volumes.

Options The following options are supported:

generic_options Specify *generic_options* supported by the generic `labelit` command. See `labelit(1M)` for descriptions of supported options.

`-o` *specific_options* Specify udfs-file-system-specific options in a comma-separated list with no intervening spaces. The following *specific_options* are available:

`lvinfo1=`*string* Specify information to be inserted in the `LVInfo1` field of the Implementation Use Volume Descriptor. Information in `LVInfo1` is generally used to identify the person creating the file system. The maximum length of the string specified is 35 bytes.

`lvinfo2=`*string* Specify information to be inserted into the `LVInfo2` field of the Implementation Use Volume Descriptor. Information in `LVInfo2` is generally used to identify the organization responsible for creating the file system. The maximum length of the string specified is 35 bytes.

`lvinfo3=`*string* Specify information to be inserted into the `LVInfo3` field of the Implementation Use Volume Descriptor. Information in `LVInfo3` is generally used to identify the contact information for the medium. The maximum length of the string specified is 35 bytes.

Operands The following operands are supported:

special Specify *special* as the physical disk slice, for example, `/dev/rdsk/c0t0d0s6`. The device can not be on a remote machine.

fsname Specify *fsname* as the mount point, (for example, `root`, `u1`, and so forth), of the file system.

volume Specify *volume* as the physical volume name.

If none of the options (*fsname*, *volume*, *specific_options*) is specified, `labelit` prints the current values of *fsname*, *volume*, `LVInfo1`, `LVInfo2` and `LVInfo3`.

Exit Status The following exit values are returned:

0 Successful completion.

non-zero An error occurred.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/file-system/udfs

See Also `labelit(1M)`, `attributes(5)`

Name labelit_ufs – provide and print labels for ufs file systems

Synopsis `labelit -F ufs [`*generic_options*`] special [`*fsname volume*`]`

Description `labelit` is used to write labels on unmounted disk file systems. Such labels may be used to uniquely identify volumes and are used by volume-oriented programs such as `volcopy(1M)`.

Options The following option is supported:

generic_options options supported by the generic `labelit` command. See `labelit(1M)`.

Operands The following operands are supported:

special name should be the physical disk section (for example, `/dev/dsk/c0d0s6`). The device may not be on a remote machine.

fsname represents the mount point (for example, `root`, `u1`, and so on) of the file system.

volume may be used to represent the physical volume name.

If *fsname* and *volume* are not specified, `labelit` prints the current values of these labels. Both *fsname* and *volume* are limited to six or fewer characters.

Exit Status The following exit values are returned:

`0` Write or display of labels was successful.

non-zero An error occurred.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also `labelit(1M)`, `volcopy(1M)`, `attributes(5)`, `ufs(7FS)`

Name latencytop – report latency-related statistics in system and in applications

Synopsis

```
latencytop [-t interval] [-o log_file] [-k log_level]
     [-f [no]feature,...] [-l log_interval] [-h]
     [ -s pid=PID | pgid=PGID ]
```

Description LatencyTOP is an observability tool that reports statistics about latencies in the system and in applications. The tool reports statistics about where and what kind of latencies are happening in the system and in the applications that are running on the system. The statistics then can be used to improve performance throughput of applications and system, as you remove the identified latencies.

The tool analyzes system activity periodically and displays the data in the output window. Two types of latencies are tracked: an LWP going in and out of sleep and an LWP spinning order to acquire a synchronization object. The tool uses the Solaris DTrace framework to collect the statistics corresponding to these two scenarios of inactivity of the system and application LWPs.

The output window is divided into two sections. An upper part displays the system-wide statistics, while the lower part displays statistics about individual processes. The user can navigate the list of processes (using the left- and right-arrow keys) and select the list they are interested in. The tool will then display statistics about that selected process in the lower part of the window. If the t or T key is pressed, the tool displays the LWP-specific view of that selected process. The t or T key can be used to toggle between the process-view and the thread-view.

During execution, a user can force a refresh of the analysis by pressing the r or R key. The interval time is restored to the default or to a specified value (if -t was used). To quit the application, the user must press the q or Q key.

Options The following options are supported:

`-f`, `--feature` [no]*feature1*,[no]*feature2*,...
Enables/disables features in LatencyTOP. Features can be only one of the following:

[no]`filter`
Filter large interruptible latencies, for example, sleep. The default is `off`.

[no]`sched`
Monitors `sched` (PID=0). The default is `off`.

[no]`sobj`
Monitors synchronize objects. The default is `on`.

[no]`low`
Lower overhead by sampling small latencies. Enabling this feature will lower CPU utilization by estimating small latencies statistically. Use it for heavy workloads such as a very busy web server. The default is `off`.

`-h`
Displays the command's usage.

`-k` *log_level*
Specifies the level of logging in the log file. Valid values are:

`0` none (default)

`1` unknown

`2` all

`-l` [*log_interval*]
Writes data to the log file every *log_interval* seconds; *log_interval* must be greater than 60.

`-o` *log_file*
Specifies the log file where output will be written. The default log file is `/var/log/latencytop.log`.

`-s` pid=*PID* | pgid=*PGID*
Tracks only the specified process or the specified process group and displays data related only to that process or the process group.

`-t` *interval*
Specifies the interval, in seconds, at which the tool collects statistics from the system. The possible values are between 1 and 60; the default is 5 seconds.

Examples

EXAMPLE 1 Running the Tool

The following command launches the tool with default values for options.

```
% latencytop
```

EXAMPLE 2 Setting the Interval

The following command sets the sampling interval to two seconds.

```
% latencytop -t 2
```

EXAMPLE 3 Specifying the Log File

The following command sets the log file to `/tmp/latencytop.log`.

```
% latencytop -o /tmp/latencytop.log
```

EXAMPLE 4 Specifying the Log Level

The following command sets the log level to `all`.

```
% latencytop -l 2
```

EXAMPLE 5 Enabling Tracing of Latencies

The following command enables the tracing of latencies caused by synchronization objects.

```
% latencytop -f sobj
```

EXAMPLE 6 Displaying Data for a Process Group

The following command displays trace date for processes belonging to Process Group 630.

```
% latencytop -s pgid=630
```

Exit Status

`0`
Successful operation.

`1`
An error occurred.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	x86, SPARC
Availability	diagnostic/latencytop
Interface Stability	Volatile

See Also

`dtrace(1M)`, `kstat(1M)`, `attributes(5)`

Usage

You must have DTrace privileges to run LatencyTOP.

Name ldapaddent – create LDAP entries from corresponding /etc files

Synopsis

```
ldapaddent [-cpv] [-a authenticationMethod] [-b baseDN]
     -D bindDN [-w bind_password] [-j passwdFile] [-f filename]
     database

ldapaddent [-cpv] -a sasl/GSSAPI [-b baseDN] [-f filename]
     database

ldapaddent -d [-v] [-a authenticationMethod] [-D bindDN]
     [-w bind_password] [-j passwdFile] database

ldapaddent [-cpv] -h LDAP_server[:serverPort] [-M domainName]
     [-N profileName] [-P certifPath] [-a authenticationMethod]
     [-b baseDN] -D bindDN [-w bind_password] [-f filename]
     [-j passwdFile] database

ldapaddent [-cpv] -h LDAP_server[:serverPort] [-M domainName]
     [-N profileName] [-P certifPath] [-a authenticationMethod]
     [-b baseDN] [-f filename] database

ldapaddent -d [-v] -h LDAP_server[:serverPort] [-M domainName]
     [-N profileName] [-P certifPath] [-a authenticationMethod]
     [-b baseDN] -D bindDN [-w bind_password] [-j passwdFile]
     database
```

Description `ldapaddent` creates entries in LDAP containers from their corresponding `/etc` files. This operation is customized for each of the standard containers that are used in the administration of Solaris systems. The *database* argument specifies the type of the data being processed. Legal values for this type are one of `aliases`, `auto_*`, `bootparams`, `ethers`, `group`, `hosts` (including both IPv4 and IPv6 addresses), `ipnodes` (alias for `hosts`), `netgroup`, `netmasks`, `networks`, `passwd`, `shadow`, `protocols`, `publickey`, `rpc`, and `services`. In addition to the preceding, the *database* argument can be one of the RBAC-related files (see `rbac(5)`):

- `/etc/user_attr`
- `/etc/security/auth_attr`
- `/etc/security/prof_attr`
- `/etc/security/exec_attr`

By default, `ldapaddent` reads from the standard input and adds this data to the LDAP container associated with the database specified on the command line. An input file from which data can be read is specified using the `-f` option.

If you specify the `-h` option, `ldapaddent` establishes a connection to the server indicated by the option in order to obtain a `DUAProfile` specified by the `-N` option. The entries will be stored in the directory described by the configuration obtained.

By default (if the `-h` option is not specified), entries will be stored in the directory based on the client's configuration. To use the utility in the default mode, the Solaris LDAP client must be set up in advance.

The location where entries are to be written can be overridden by using the `-b` option.

If the entry to be added exists in the directory, the command displays an error and exits, unless the `-c` option is used.

Although, there is a `shadow` database type, there is no corresponding `shadow` container. Both the `shadow` and the `passwd` data is stored in the `people` container itself. Similarly, data from `networks` and `netmasks` databases are stored in the `networks` container.

The `user_attr` data is stored by default in the `people` container. The `prof_attr` and `exec_attr` data is stored by default in the `SolarisProfAttr` container.

You must add entries from the `passwd` database before you attempt to add entries from the `shadow` database. The addition of a `shadow` entry that does not have a corresponding `passwd` entry will fail.

The `passwd` database must precede the `user_attr` database.

For better performance, the recommended order in which the databases should be loaded is as follows:

- `passwd` database followed by `shadow` database
- `networks` database followed by `netmasks` database
- `bootparams` database followed by `ethers` database

Only the first entry of a given type that is encountered will be added to the LDAP server. The `ldapaddent` command skips any duplicate entries.

Options The `ldapaddent` command supports the following options:

`-a` *authenticationMethod*
Specify authentication method. The default value is what has been configured in the profile. The supported authentication methods are:

```
simple
sasl/CRAM-MD5
sasl/DIGEST-MD5
sasl/GSSAPI
tls:simple
tls:sasl/CRAM-MD5
tls:sasl/DIGEST-MD5
```

Selecting `simple` causes passwords to be sent over the network in clear text. Its use is strongly discouraged. Additionally, if the client is configured with a profile which uses no authentication, that is, either the `credentialLevel` attribute is set to `anonymous` or `authenticationMethod` is set to `none`, the user must use this option to provide an authentication method. If the authentication method is `sasl/GSSAPI`, *bindDN* and *bindPassword* is not required and the `hosts` and `ipnodes` fields of `/etc/nsswitch.conf` must be configured as:

```
hosts: dns files
ipnodes: dns files
```

See `nsswitch.conf(4)`.

`-b` *baseDN*
Create entries in the *baseDN* directory. *baseDN* is not relative to the client's default search base, but rather. it is the actual location where the entries will be created. If this parameter is not specified, the first search descriptor defined for the service or the default container will be used.

`-c`
Continue adding entries to the directory even after an error. Entries will not be added if the directory server is not responding or if there is an authentication problem.

`-D` *bindDN*
Create an entry which has write permission to the *baseDN*. When used with `-d` option, this entry only needs read permission.

`-d`
Dump the LDAP container to the standard output in the appropriate format for the given database.

`-f` *filename*
Indicates input file to read in an `/etc/` file format.

`-h` *LDAP_server*[:*serverPort*]
Specify an address (or a name) and an optional port of the LDAP server in which the entries will be stored. The current naming service specified in the `nsswitch.conf` file is used. The default value for the port is `389`, except when TLS is specified as the authentication method. In this case, the default LDAP server port number is `636`.

The format to specify the address and port number for an IPv6 address is:

`[`*ipv6_addr*`]:`*port*

To specify the address and port number for an IPv4 address, use the following format:

ipv4_addr:*port*

If the host name is specified, use the format:

host_name:*port*

`-j` *passwdFile*
Specify a file containing the password for the bind DN or the password for the SSL client's key database. To protect the password, use this option in scripts and place the password in a secure file. This option is mutually exclusive of the `-w` option.

`-M` *domainName*
The name of a domain served by the specified server. If not specified, the default domain name will be used.

`-N` *profileName*
Specify the `DUAProfile` name. A profile with such a name is supposed to exist on the server specified by `-h` option. Otherwise, a default `DUAProfile` will be used. The default value is `default`.

`-P` *certifPath*
The certificate path for the location of the certificate database. The value is the path where security database files reside. This is used for TLS support, which is specified in the `authenticationMethod` and `serviceAuthenticationMethod` attributes. The default is `/var/ldap`.

`-p`
Process the `password` field when loading password information from a file. By default, the `password` field is ignored because it is usually not valid, as the actual password appears in a `shadow` file.

`-w` *bindPassword*
Password to be used for authenticating the *bindDN*. If this parameter is missing, the command will prompt for a password. `NULL` passwords are not supported in LDAP.

When you use `-w` *bindPassword* to specify the password to be used for authentication, the password is visible to other users of the system by means of the `ps` command, in script files or in shell history.

If you supply "-" (hyphen) as a password, you will be prompted to enter a password.

`-v`
Verbose.

Operands

The following operands are supported:

database
The name of the database or service name. Supported values are: `aliases`, `auto_*`, `bootparams`, `ethers`, `group`, `hosts` (including IPv6 addresses), `netgroup`, `netmasks`, `networks`, `passwd`, `shadow`, `protocols`, `publickey`, `rpc`, and `services`. Also supported are `auth_attr`, `prof_attr`, `exec_attr`, `user_attr`, and `projects`.

Examples

EXAMPLE 1 Adding Password Entries to the Directory Server

The following example shows how to add password entries to the directory server:

```
example# ldapaddent -D "cn=directory manager" -w secret \
      -f /etc/passwd passwd
```

EXAMPLE 2 Adding Group Entries

The following example shows how to add `group` entries to the directory server using `sasl/CRAM-MD5` as the authentication method:

```
example# ldapaddent -D "cn=directory manager" -w secret \
     -a "sasl/CRAM-MD5" -f /etc/group group
```

EXAMPLE 3 Adding auto_master Entries

The following example shows how to add `auto_master` entries to the directory server:

```
example# ldapaddent -D "cn=directory manager" -w secret \
     -f /etc/auto_master auto_master
```

EXAMPLE 4 Dumping `passwd` Entries from the Directory to File

The following example shows how to dump `password` entries from the directory to a file `foo`:

```
example# ldapaddent -d passwd > foo
```

EXAMPLE 5 Adding Password Entries to a Specific Directory Server

The following example shows how to add password entries to a directory server that you specify:

```
example# ldapaddent -h 10.10.10.10:3890 \
-M another.domain.name -N special_duaprofile \
-D "cn=directory manager" -w secret \
-f /etc/passwd passwd
```

Exit Status

The following exit values are returned:

0
Successful completion.

>0
An error occurred.

Files

`/var/ldap/ldap_client_file`
`/var/ldap/ldap_client_cred`
Files containing the LDAP configuration of the client. These files are not to be modified manually. Their content is not guaranteed to be human readable. Use `ldapclient(1M)` to update these files.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/network/nis
Interface Stability	Committed

See Also

`ldaplist(1)`, `ldapmodify(1)`, `ldapmodrdn(1)`, `ldapsearch(1)`, `idsconfig(1M)`, `ldapclient(1M)`, `nsswitch.conf(4)`, `attributes(5)`

Oracle Solaris 11.1 Administration: Security Services

Caution Currently StartTLS is not supported by `libldap.so.5`, therefore the port number provided refers to the port used during a TLS open, rather than the port used as part of a StartTLS sequence. For example:

```
-h foo:1000 -a tls:simple
```

The preceding refers to a raw TLS open on host `foo` port 1000, not an open, StartTLS sequence on an unsecured port 1000. If port 1000 is unsecured the connection will not be made.

Name ldap_cachemgr – LDAP daemon to manage client configuration for LDAP based Network Information Service lookups

Synopsis

```
/usr/lib/ldap/ldap_cachemgr [-l log-file] [-g]
```

Description The `ldap_cachemgr` daemon is a process that provides an up-to-date configuration cache for LDAP naming services. It is started during multi-user boot.

The `ldap_cachemgr` utility provides caching for all parameters as specified and used by the LDAP naming service clients. The `ldap_cachemgr` utility uses the cache files which are originally created by executing the `ldapclient(1M)` utility, as cold start files. Updates to the cache files take place dynamically if profiles are used to configure the client. See the `init` option to `ldapclient(1M)`.

The `ldap_cachemgr` utility helps improve the performance of the clients that are using LDAP as the Naming service repository. In order for the LDAP naming services to function properly, the `ldap_cachemgr` daemon must be running. `ldap_cachemgr` also improves system security by making the configuration files readable by superuser only.

The cache maintained by this daemon is shared by all the processes that access LDAP Naming information. All processes access this cache through a door call. On startup, `ldap_cachemgr` initializes the cache from the cache files. See `ldapclient(1M)`. Thus, the cache survives machine reboots.

The `ldap_cachemgr` daemon also acts as its own administration tool. If an instance of `ldap_cachemgr` is already running, commands are passed transparently to the running version.

Options The following options are supported:

`-g`
: Print current configuration and statistics to standard output. This is the only option executable without superuser privileges.

`-l` *log-file*
: Cause `ldap_cachemgr` to use a log file other than the default `/var/ldap/cachemgr.log`.

Examples

EXAMPLE 1 Stopping and Restarting the `ldap_cachemgr` Daemon

The following example shows how to stop and to restart the `ldap_cachemgr` daemon.

```
example# svcadm disable network/ldap/client
example# svcadm enable network/ldap/client
```

EXAMPLE 2 Forcing `ldap_cachemgr` to Reread Configuration Files

The following example shows how to force `ldap_cachemgr` to reread the `/var/ldap/ldap_client_file` and `/var/ldap/ldap_client_cred` files

```
example# pkill -HUP ldap_cachemgr
```

Files

`/var/ldap/cachemgr.log`
Default log file.

`/var/ldap/ldap_client_file`
`/var/ldap/ldap_client_cred`
Files containing the LDAP configuration of the client. These files are not to be modified manually. Their content is not guaranteed to be human readable. Use `ldapclient(1M)` to update these files.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/network/nis

See Also

`ldapadd(1)`, `ldapdelete(1)`, `ldaplist(1)`, `ldapmodify(1)`, `ldapmodrdn(1)`, `ldapsearch(1)`, `pkill(1)`, `svcs(1)`, `idsconfig(1M)`, `ldapaddent(1M)`, `ldapclient(1M)`, `svcadm(1M)`, `signal.h(3HEAD)`, `resolv.conf(4)`, `attributes(5)`, `ldap(5)`, `smf(5)`

Notes

The `ldap_cachemgr` service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/network/ldap/client
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. The service's status can be queried using the `svcs(1)` command.

Name ldapclient – initialize LDAP client machine or output an LDAP client profile in LDIF format

Synopsis

```
/usr/sbin/ldapclient [-v | -q] init [-a profileName=profileName]
     [-a domainName=domain] [-a proxyDN=proxyDN]
     [-a proxyPassword=password]
     [-a authenticationMethod=authenticationMethod]
     [-a enableShadowUpdate=true | false]
     [-a adminDN=adminDN]
     [-a adminPassword=adminPassword]
     [-a certificatePath=path] [-d bindDN] [-w bindPassword]
     [-j passwdFile] [-y passwdFile]
     [-z adminrPasswdFile] LDAP_server[:port_number]

/usr/sbin/ldapclient [-v | -q] manual [-a attrName=attrVal]

/usr/sbin/ldapclient [-v | -q] mod [-a attrName=attrVal]

/usr/sbin/ldapclient [-v | -q] list

/usr/sbin/ldapclient [-v | -q] uninit

/usr/sbin/ldapclient [-v | -q] genprofile -a profileName=profileName
     [-a attrName=attrVal]
```

Description The `ldapclient` utility can be used to:

- initialize LDAP client machines
- restore the network service environment on LDAP clients
- list the contents of the LDAP client cache in human readable format.

The `init` form of the `ldapclient` utility is used to initialize an LDAP client machine, using a profile stored on an LDAP server specified by `LDAP_server`. The LDAP client will use the attributes in the specified profile to determine the configuration of the LDAP client. Using a configuration profile allows for easy installation of LDAP client and propagation of configuration changes to LDAP clients. The `ldap_cachemgr(1M)` utility will update the LDAP client configuration when its cache expires by reading the profile. For more information on the configuration profile refer to IETF document *A Configuration Schema for LDAP Based Directory User Agents*.

The `manual` form of the `ldapclient` utility is used to initialize an LDAP client machine manually. The LDAP client will use the attributes specified on the command line. Any unspecified attributes will be assigned their default values. At least one server must be specified in the `defaultServerList` or the `preferredServerList` attributes.The `domainName` attribute must be specified if the client's `domainName` is not set.

The `mod` form of the `ldapclient` utility is used to modify the configuration of an LDAP client machine that was setup manually. This option modifies only those LDAP client configuration attributes specified on the command line. The `mod` option should only be used on LDAP clients that were initialized using the `manual` option.

Regardless of which method is used for initialization, if a client is to be configured to use a proxy `credentialLevel`, proxy credentials must be provided using `-a proxyDN=`*proxyDN* and `-a proxyPassword=`*proxyPassword* options. However, if `-a proxyPassword=`*proxyPassword* is not specified, `ldapclient` will prompt for it. Note that `NULL` passwords are not allowed in LDAP. If a self `credentialLevel` is configured, `authenticationMethod` must be `sasl/GSSAPI`.

Similarily, if a client is to be configured to enable shadow information update and use a proxy credentialLevel, administrator credentials must be provided using `-a adminDN=`*adminDN* and `-a adminPassword=`*adminPassword*. However, the shadow information update does not need the administrator credentials if a self `credentialLevel` is configured.

The naming service-specific configuration properties are stored in the `svc:/network/ldap/client` SMF service. Modifying the SMF properties directly is not advised. Use this tool instead.

Other configuration might be modified during installation. It will be backed up to `/var/ldap/restore`. The files that are typically modified during initialization are:

- `/etc/nsswitch.conf`
- `/etc/defaultdomain` (if it exists)
- `/var/yp/binding/`‘`domainname`‘ (for a NIS [YP] client)

LDAP configuration is managed in location profiles; this affects when and how LDAP configuration is made to persist. Please refer to `netcfg(1M)` for more information about location profiles, and to the section “Interaction with Location Profiles” below for more details on how LDAP configuration should be managed when automatic network configuration is being used.

`ldapclient` does not set up a client to resolve hostnames using DNS. It simply copies `/etc/nsswitch.ldap` to `/etc/nsswitch.conf`. If you prefer to use DNS for host resolution, please refer to the DNS documentation for information on setting up DNS. See `resolv.conf(4)`. If you want to use `sasl/GSSAPI` as the authentication method, you have to use DNS for `hosts` and `ipnodes` resolution.

The `list` form of the `ldapclient` utility is used to list the LDAP client configuration. The output will be human readable. LDAP configuration files are not guaranteed to be human readable. Note that for security reason, the values for adminDN and adminPassword will not be displayed.

The `uninit` form of the `ldapclient` utility is used to uninitialize the network service environment, restoring it to the state it was in prior to the last execution of `ldapclient` using `init` or `manual`. The restoration will succeed only if the machine was initialized with the `init` or `manual` form of `ldapclient`, as it uses the backup files created by these options.

The `genprofile` option is used to write an LDIF formatted configuration profile based on the attributes specified on the command line to standard output. This profile can then be loaded into an LDAP server to be used as the client profile, which can be downloaded by means of the

`ldapclient init` command. Loading the LDIF formatted profile to the directory server can be done through `ldapadd(1)`, or through any server specific import tool. Note that the attributes `proxyDN`, `proxyPassword`, `certificatePath`, `domainName`, `enableShadowUpdate`, `adminDN`, and `adminPassword` are not part of the configuration profile and thus are not permitted.

You must have superuser privileges to run the `ldapclient` command, except with the `genprofile` option.

To access the information stored in the directory, clients can either authenticate to the directory, or use an unauthenticated connection. The LDAP client is configured to have a credential level of either `anonymous` or `proxy`. In the first case, the client does not authenticate to the directory. In the second case, client authenticates to the directory using a proxy identity for read access, and using a administrator identity for write access if `enableShadowUpdate` is configured. In the third case, client authenticates to the directory using a Kerberos principal that is mapped to an LDAP identity by the LDAP server. Refer to the chapter on implementing security in the *Oracle Solaris Administration: Naming and Directory Services* or your appropriate directory server documentation for identity mapping details.

If a client is configured to use an identity, you can configure which authentication method the client will use. The LDAP client supports the following authentication methods:

```
none
simple
sasl/CRAM-MD5
sasl/DIGEST-MD5
sasl/GSSAPI
tls:simple
tls:sasl/CRAM-MD5
tls:sasl/DIGEST-MD5
```

Note that some directory servers may not support all of these authentication methods. For `simple`, be aware that the bind password will be sent in the clear to the LDAP server. For those authentication methods using TLS (transport layer security), the entire session is encrypted. You will need to install the appropriate certificate databases to use TLS.

Interaction with Location Profiles

As previously mentioned, LDAP configuration is managed in location profiles (refer to `netcfg(1M)` for more information about location profiles). These profiles are either fixed, meaning the network configuration is being managed in the traditional way, or reactive, meaning the network configuration is being managed automatically, reacting to changes in the network environment according to policy rules specified in the profiles.

When a fixed location (there can currently be only one, the `DefaultFixed` location) is active, changes made to the SMF repository--including those made indirectly using tools such as `ldapclient`--will be applied to the location when it is disabled, and thus will be restored if that location is later re-enabled.

When a reactive location is active, changes should not be applied directly to the SMF repository; these changes will not be preserved in the location profile, and will thus be lost if the location is disabled, or if the system's network configuration, as managed by `svc:/network/physical:default` and `svc:/network/location:default`, is refreshed or restarted. Changes should instead be applied to the location itself, using the `netcfg(1M)` command; this will save the change to the location profile repository, and will also apply it to the SMF repository (if the change is made to the currently active location).

Support for LDAP configuration in a location is currently very limited. The user must specify a server name, in the `ldap-nameservice-servers` property, from which a complete configuration profile can be obtained. No other parameters may be specified. Further, if LDAP configuration is not included in a location, then the relevant LDAP services will be disabled when the location is enabled.

More sophisticated LDAP configurations can be accommodated by creating a script that issues appropriate `ldapclient` commands to configure LDAP; this script can then be used to create an ENM profile, which can be enabled/disabled as needed. Please refer to `netcfg(1M)` for more information about ENM profiles.

Commands

The following commands are supported:

`init`
: Initialize client from a profile on a server.

`manual`
: Manually initialize client with the specified attribute values.

`mod`
: Modify attribute values in the configuration file after a manual initialization of the client.

`list`
: Write the contents of the LDAP client cache to standard output in human readable form.

`uninit`
: Uninitialize an LDAP client, assuming that `ldapclient` was used to initialize the client.

`genprofile`
: Generate a configuration profile in LDIF format that can then be stored in the directory for clients to use, with the `init` form of this command.

Attributes

The following attributes are supported:

`adminDN`
: Specify the Bind Distinguished Name for the administrator identity that is used for shadow information update. This option is required if the credential level is `proxy`, and `enableShadowUpdate` is set to `true`. There is no default value.

`adminPassword`
: Specify the administrator password. This option is required if the credential level is `proxy`, and `enableShadowUpdate` is set to `true`. There is no default value.

`attributeMap`
Specify a mapping from an attribute defined by a service to an attribute in an alternative schema. This can be used to change the default schema used for a given service. The syntax of `attributeMap` is defined in the profile IETF draft. This option can be specified multiple times. The default value for all services is `NULL`. In the example,

```
attributeMap: passwd:uid=employeeNumber
```

the LDAP client would use the LDAP attribute `employeeNumber` rather than `uid` for the `passwd` service. This is a multivalued attribute.

`authenticationMethod`
Specify the default authentication method used by all services unless overridden by the `serviceAuthenticationMethod` attribute. Multiple values can be specified by using a semicolon-separated list. The default value is `none`. For those services that use `credentialLevel` and `credentialLevel` is `anonymous`, this attribute is ignored. Services such as `pam_ldap` will use this attribute, even if `credentialLevel` is anonymous. The supported authentication methods are described above. If the authenticationMethod is `sasl/GSSAPI`, the `hosts` and `ipnodes` of `/etc/nsswitch.conf` must be configured with DNS support, for example:

```
hosts: dns files
ipnodes: dns files
```

`bindTimeLimit`
The maximum time in seconds that a client should spend performing a bind operation. Set this to a positive integer. The default value is 30.

`certificatePath`
The certificate path for the location of the certificate database. The value is the path where security database files reside. This is used for TLS support, which is specified in the `authenticationMethod` and `serviceAuthenticationMethod` attributes. The default is `/var/ldap`.

`credentialLevel`
Specify the credential level the client should use to contact the directory. The credential levels supported are `anonymous`, `proxy`, and `self`. If a `proxy` credential level is specified, then the `authenticationMethod` attribute must be specified to determine the authentication mechanism. Also, if the credential level is `proxy` and at least one of the authentication methods require a bind DN, the `proxyDN` and `proxyPassword` attribute values must be set. In addition, if `enableShadowUpdate` is set to `true`, the `adminDN` and `adminPassword` values must be set. If a self credential level is specified, the `authenticationMethod` must be `sasl/GSSAPI`.

`defaultSearchBase`
Specify the default search base DN. There is no default. The `serviceSearchDescriptor` attribute can be used to override the `defaultSearchBase` for given services.

`defaultSearchScope=one | sub`
Specify the default search scope for the client's search operations. This default can be overridden for a given service by specifying a `serviceSearchDescriptor`. The default is one level search.

`defaultServerList`
A space separated list of server names or server addresses, either IPv4 or IPv6. If you specify server names, be sure that the LDAP client can resolve the name without the LDAP name service. You must resolve the LDAP servers' names by using either `files` or `dns`. If the LDAP server name cannot be resolved, your naming service will fail.

The port number is optional. If not specified, the default LDAP server port number 389 is used, except when TLS is specified in the authentication method. In this case, the default LDAP server port number is 636.

The format to specify the port number for an IPv6 address is:

```
[ipv6_addr]:port
```

To specify the port number for an IPv4 address, use the following format:

```
ipv4_addr:port
```

If the host name is specified, use the format:

```
host_name:port
```

If you use TLS, the LDAP server's hostname must match the hostname in the TLS certificate. Typically, the hostname in the TLS certificate is a fully qualified domain name. With TLS, the LDAP server host addresses must resolve to the hostnames in the TLS certificate. You must use `files` or `dns` to resolve the host address.

`domainName`
Specify the DNS domain name. This becomes the default domain for the machine. The default is the current domain name. This attribute is only used in client initialization.

`enableShadowUpdate=true | false`
Specify whether the client is allowed to update shadow information. If set to `true` and the credential level is `proxy`, `adminDN` and `adminPassword` must be specified.

`followReferrals=true | false`
Specify the referral setting. A setting of true implies that referrals will be automatically followed and false would result in referrals not being followed. The default is true.

`objectclassMap`
Specify a mapping from an `objectclass` defined by a service to an `objectclass` in an alternative schema. This can be used to change the default schema used for a given service. The syntax of `objectclassMap` is defined in the profile IETF draft. This option can be specified multiple times. The default value for all services is `NULL`. In the example,

```
objectclassMap=passwd:posixAccount=unixAccount
```

the LDAP client would use the LDAP `objectclass` of `unixAccount` rather than the `posixAccount` for the `passwd` service. This is a multivalued attribute.

`preferredServerList`
Specify the space separated list of server names or server addresses, either IPv4 or IPv6, to be contacted before servers specified by the `defaultServerList` attribute. If you specify server names, be sure that the LDAP client can resolve the name without the LDAP name service. You must resolve the LDAP servers' names by using either `files` or `dns`. If the LDAP server name cannot be resolved, your naming service will fail.

The port number is optional. If not specified, the default LDAP server port number 389 is used, except when TLS is specified in the authentication method. In this case, the default LDAP server port number is 636.

The format to specify the port number for an IPv6 address is:

```
[ipv6_addr]:port
```

To specify the port number for an IPv4 address, use the following format:

```
ipv4_addr:port
```

If the host name is specified, use the format:

```
host_name:port
```

If you use TLS, the LDAP server's hostname must match the hostname in the TLS certificate. Typically, the hostname in the TLS certificate is a fully qualified domain name. With TLS, the LDAP server host addresses must resolve to the hostnames in the TLS certificate. You must use `files` or `dns` to resolve the host address.

`profileName`
Specify the profile name. For `ldapclient init`, this attribute is the name of an existing profile which may be downloaded periodically depending on the value of the `profileTTL` attribute. For `ldapclient genprofile`, this is the name of the profile to be generated. The default value is `default`.

`profileTTL`
Specify the TTL value in seconds for the client information. This is only relevant if the machine was initialized with a client profile. If you do not want `ldap_cachemgr(1M)` to attempt to refresh the LDAP client configuration from the LDAP server, set `profileTTL` to 0 (zero). Valid values are either zero 0 (for no expiration) or a positive integer in seconds. The default value is 12 hours.

`proxyDN`
Specify the Bind Distinguished Name for the proxy identity. This option is required if the credential level is `proxy`, and at least one of the authentication methods requires a bind DN. There is no default value.

`proxyPassword`
Specify client proxy password. This option is required if the credential level is `proxy`, and at least one of the authentication methods requires a bind DN. There is no default.

`searchTimeLimit`
Specify maximum number of seconds allowed for an LDAP search operation. The default is 30 seconds. The server may have its own search time limit.

`serviceAuthenticationMethod`
Specify authentication methods to be used by a service in the form *servicename*:`authenticationmethod`, for example:

```
pam_ldap:tls:simple
```

For multiple authentication methods, use a semicolon-separated list. The default value is no service authentication methods, in which case, each service would default to the `authenticationMethod` value. The supported authentications are described above.

Three services support this feature: `passwd-cmd`, `keyserv`, and `pam_ldap`. The `passwd-cmd` service is used to define the authentication method to be used by `passwd(1)` to change the user's password and other attributes. The `keyserv` service is used to identify the authentication method to be used by the `chkey(1)` and `newkey(1M)` utilities. The `pam_ldap` service defines the authentication method to be used for authenticating users when `pam_ldap(5)` is configured. If this attribute is not set for any of these services, the `authenticationMethod` attribute is used to define the authentication method. This is a multivalued attribute.

`serviceCredentialLevel`
Specify credential level to be used by a service. Multiple values can be specified in a space-separated list. The default value for all services is `NULL`. The supported credential levels are: `anonymous` or `proxy`. At present, no service uses this attribute. This is a multivalued attribute.

`serviceSearchDescriptor`
Override the default base DN for LDAP searches for a given service. The format of the descriptors also allow overriding the default search scope and search filter for each service. The syntax of `serviceSearchDescriptor` is defined in the profile IETF draft. The default value for all services is `NULL`. This is a multivalued attribute. In the example,

```
serviceSearchDescriptor=passwd:ou=people,dc=a1,dc=acme,dc=com?one
```

the LDAP client would do a one level search in `ou=people,dc=a1,dc=acme,dc=com` rather than `ou=people,`*defaultSearchBase* for the `passwd` service.

Options The following options are supported:

`-a attrName=`*attrValue*
Specify `attrName` and its value. See `SYNOPSIS` for a complete list of possible attribute names and values.

`-D` *bindDN*
Specifies an entry that has read permission for the requested database.

`-j` *passwdFile*
Specify a file containing the password for the bind DN or the password for the SSL client's key database. To protect the password, use this option in scripts and place the password in a secure file. This option is mutually exclusive of the `-w` option.

`-q`
Quiet mode. No output is generated.

`-v`
Verbose output.

`-w` *bindPassword*
Password to be used for authenticating the bind DN. If this parameter is missing, the command will prompt for a password. `NULL` passwords are not supported in LDAP.

When you use `-w` *bindPassword* to specify the password to be used for authentication, the password is visible to other users of the system by means of the `ps` command, in script files, or in shell history.

If you supply "`-`" (hyphen) as a password, the command will prompt for a password.

`-y` *passwdFile*
Specify a file containing the password for the proxy DN. To protect the password, use this option in scripts and place the password in a secure file. This option is mutually exclusive of the `-a` *proxyPassword* option.

`-z` *adminrPasswdFile*
Specify a file containing the password for the `adminDN`. To protect the password, use this option in scripts and place the password in a secure file. This option is mutually exclusive of the `-a` *adminPassword* option.

Operands The following operand is supported:

LDAP_server
An address or a name for the LDAP server from which the profile will be loaded. The current naming service specified in the `nsswitch.conf` file is used. Once the profile is loaded, the `preferredServerList` and `defaultServerList` specified in the profile are used.

Examples **EXAMPLE 1** Setting Up a Client By Using the Default Profile Stored on a Specified LDAP Server

The following example shows how to set up a client using the default profile stored on the specified LDAP server. This command will only be successful if either the credential level in the profile is set to `anonymous` or the authentication method is set to `none`.

```
example# ldapclient init 172.16.100.1
```

EXAMPLE 2 Setting Up a Client By Using the simple Profile Stored on a Specified LDAP Server

The following example shows how to set up a client using the `simple` profile stored on the specified LDAP server. The domainname is set to `xyz.mycompany.com` and the proxyPassword is `secret`.

```
example# ldapclient init -a profileName=simple \
-a domainName=xyz.mycompany.com \
-a proxyDN=cn=proxyagent,ou=profile,dc=xyz,dc=mycompany,dc=com \
-a proxyPassword=secret '['fe80::a00:20ff:fea3:388']':386
```

EXAMPLE 3 Setting Up a Client Using Only One Server

The following example shows how to set up a client using only one server. The authentication method is set to none, and the search base is `dc=mycompany,dc=com`.

```
example# ldapclient manual -a authenticationMethod=none \
-a defaultSearchBase=dc=mycompany,dc=com \
-a defaultServerList=172.16.100.1
```

EXAMPLE 4 Setting Up a Client Using Only One Server That Does Not Follow Referrals

The following example shows how to set up a client using only one server. The credential level is set to `proxy`. The authentication method of is `sasl/CRAM-MD5`, with the option not to follow referrals. The domain name is `xyz.mycompany.com`, and the LDAP server is running on port number 386 at IP address `172.16.100.1`.

```
example# ldapclient manual \
-a credentialLevel=proxy \
-a authenticationMethod=sasl/CRAM-MD5 \
-a proxyPassword=secret \
-a proxyDN=cn=proxyagent,ou=profile,dc=xyz,dc=mycompany,dc=com \
-a defaultSearchBase=dc=xyz,dc=mycompany,dc=com \
-a domainName=xyz.mycompany.com \
-a followReferrals=false \
-a defaultServerList=172.16.100.1:386
```

EXAMPLE 5 Using genprofile to Set Only the defaultSearchBase and the Server Addresses

The following example shows how to use the `genprofile` command to set the `defaultSearchBase` and the server addresses.

```
example# ldapclient genprofile -a profileName=myprofile \
-a defaultSearchBase=dc=eng,dc=sun,dc=com \
-a "defaultServerList=172.16.100.1 172.16.234.15:386" \
> myprofile.ldif
```

EXAMPLE 6 Creating a Profile on IPv6 servers

The following example creates a profile on IPv6 servers

EXAMPLE 6 Creating a Profile on IPv6 servers *(Continued)*

```
example# ldapclient genprofile -a profileName=eng \
-a credentialLevel=proxy \
-a authenticationMethod=sasl/DIGEST-MD5 \
-a defaultSearchBase=dc=eng,dc=acme,dc=com \
-a "serviceSearchDescriptor=passwd:ou=people,dc=a1,dc=acme,dc=com?one"\
-a preferredServerList= '['fe80::a00:20ff:fea3:388']' \
-a "defaultServerList='['fec0::111:a00:20ff:fea3:edcf']' \
    '['fec0::111:a00:20ff:feb5:e41']'" > eng.ldif
```

EXAMPLE 7 Creating a Profile That Overrides Every Default Value

The following example shows a profile that overrides every default value.

```
example# ldapclient genprofile -a profileName=eng \
-a credentialLevel=proxy -a authenticationMethod=sasl/DIGEST-MD5 \
-a bindTimeLimit=20 \
-a defaultSearchBase=dc=eng,dc=acme,dc=com \
-a "serviceSearchDescriptor=passwd:ou=people,dc=a1,dc=acme,dc=com?one"\
-a serviceAuthenticationMethod=pam_ldap:tls:simple \
-a defaultSearchScope=sub \
-a attributeMap=passwd:uid=employeeNumber \
-a objectclassMap=passwd:posixAccount=unixAccount \
-a followReferrals=false -a profileTTL=6000 \
-a preferredServerList=172.16.100.30 -a searchTimeLimit=30 \
-a "defaultServerList=172.16.200.1 172.16.100.1 192.168.5.6" > eng.ldif
```

Exit Status

The following exit values are returned:

0 The command successfully executed.

1 An error occurred. An error message is output.

2 `proxyDN` and `proxyPassword` attributes are required, but they are not provided.

Files

`/var/ldap/ldap_client_cred`
`/var/ldap/ldap_client_file`
Contain the LDAP configuration of the client. These files are not to be modified manually. Their content is not guaranteed to be human readable. Use `ldapclient` to update them.

`/etc/defaultdomain`
System default domain name, matching the domain name of the data in the LDAP servers. See `defaultdomain(4)`.

`/etc/nsswitch.conf`
Configuration file for the name-service switch. See `nsswitch.conf(4)`.

`/etc/nsswitch.ldap`
Sample configuration file for the name-service switch configured with LDAP and files.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/network/nis
Interface Stability	Committed

See Also chkey(1), ldapadd(1), ldapdelete(1), ldaplist(1), ldapmodify(1), ldapmodrdn(1), ldapsearch(1), idsconfig(1M), ldapaddent(1M), ldap_cachemgr(1M), netcfg(1M), defaultdomain(4), nsswitch.conf(4), resolv.conf(4), attributes(5)

Caution Currently StartTLS is not supported by libldap.so.5, therefore the port number provided refers to the port used during a TLS open, rather than the port used as part of a StartTLS sequence. To avoid timeout delays, mixed use of TLS and non-TLS authentication mechanisms is not recommended.

For example:

```
-h foo:1000 -a authenticationMethod=tls:simple
```

...or:

```
defaultServerList= foo:1000
authenticationMethod= tls:simple
```

The preceding refers to a raw TLS open on host foo port 1000, not an open, StartTLS sequence on an unsecured port 1000. If port 1000 is unsecured the connection will not be made.

As a second example, the following will incur a significant timeout delay while attempting the connection to foo:636 with an unsecured bind.

```
defaultServerList= foo:636 foo:389
authenticationMethod= simple
```

Name ldmad – Logical Domains Agents daemon

Synopsis

```
/usr/lib/ldoms/ldmad
```

Description The `ldmad` daemon is part of the framework that enables Logical Domain agents to run on a Logical Domain. A Logical Domain agent is a component that interacts with the control domain to provide features or information.

`ldmad` is responsible for running agents on a Logical Domain. `ldmad` must be enabled to ensure the proper functionality of all features provided by the domain manager on the control domain. The daemon is started at boot time and has no configuration options.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/ldoms
Interface Stability	Unstable

See Also `svcs(1)`, `svcadm(1M)`, `syslog(3C)`, `syslog.conf(4)`, `attributes(5)`, `smf(5)`

Oracle VM Server for SPARC 2.2 Administration Guide

Errors `ldmad` uses `syslog(3C)` to report status and error messages. Error messages are logged with the `LOG_ERR` and `LOG_NOTICE` priorities. Informational messages are logged with the `LOG_INFO` priority. The default entries in the `/etc/syslog.conf` file specify that all `ldmad` error messages are written to the `/var/adm/messages` log.

Notes The `ldmad` service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/ldoms/agents:default
```

Administrative actions on this service, such as enabling, disabling, or requesting restart can be performed using `svcadm(1M)`. The service's status can be queried using the `svcs(1)` command.

Name link, unlink – link and unlink files and directories

Synopsis

```
/usr/sbin/link existing-file new-file

/usr/xpg4/bin/link existing-file new-file

/usr/sbin/unlink file
```

Description The `link` and `unlink` commands link and unlink files and directories. Only super-users can use these commands on directories.

Use `link` to create a new file that points to an existing file. The *existing-file* and *new-file* operands specify the existing file and newly-created files. See `OPERANDS`.

Note that the ZFS file system does not support links between directories.

`link` and `unlink` directly invoke the `link(2)` and `unlink(2)` system calls, performing exactly what they are told to do and abandoning all error checking. This differs from the `ln(1)` command. See `ln(1)`.

While linked files and directories can be removed using `unlink`, it is safer to use `rm(1)` and `rmdir(1)` instead. See `rm(1)` and `rmdir(1)`.

/usr/xpg4/bin/link If the existing file being hard linked is itself a symbolic link, then the newly created file (*new-file*) will be a hard link to the file referenced by the symbolic link, not to the symbolic link object itself (*existing-file*).

Operands The following operands are supported:

existing-file Specifies the name of the existing file to be linked.

file Specifies the name of the file to be unlinked.

new-file Specifies the name of newly created (linked) file.

Environment Variables See `environ(5)` for descriptions of the following environment variables that affect the execution of `link`: `LANG`, `LC_ALL`, `LC_CTYPE`, `LC_MESSAGES`, and `NLSPATH`.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

/usr/xpg4/bin/link

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/xopen/xcu4
Interface Stability	Committed
Standard	See `standards(5)`.

See Also ln(1), rm(1), link(2), unlink(2), attributes(5), environ(5), standards(5)

Name llc2_loop – loopback diagnostics to test the driver, adapter and network.

Synopsis

```
/usr/lib/llc2/llc2_loop2 [-v] ppa

/usr/lib/llc2/llc2_loop3 ppa sap frames

/usr/lib/llc2/llc2_loop3 ppa type frames

/usr/lib/llc2/llc2_loop4 [-v] ppa
```

Description

Loop 2 The `loop2` test sends a NULL XID frame to the broadcast (all 1's) destination MAC address. The source SAP (Service Access Point) value used is 0x04 (SNA's SAP). Therefore, if SNA is running on the system, the `loop2` test will fail. The destination SAP value is the NULL SAP (0x00). This test finds out who is listening and can receive frames sent out from a node. The verbose (`-v`) option displays the MAC address of responding nodes. All possible responders may not be displayed, since the `loop2` test only waits for responses for 2 seconds, but during this time 50-200 nodes may be displayed. The most likely error is:

```
Unexpected DLPI primitive x, expected y.
```

where $x = 5$ and $y = 6$. From `/usr/include/sys/dlpi.h`, the expected return value from one of the DLPI primitives is 6 (`DL_OK_ACK`), but instead a 5 (`DL_ERROR_ACK`) was received. This can occur for two reasons:

- The `loop2` command was issued to a non-existent PPA (Physical Point of Attachment).
- The SAP (0x04) is already in use (for example, the SNA subsystem is up).

Loop 3 The `loop3` test sends 1,495 byte Unnumbered Information (UI) frames to the NULL (all 0's) destination MAC address. This should be used along with data capture either on the local node or another node on the same LAN to verify the transmission of data. The *ppa* argument specifies the adapter on which to run the test. The *ppa* is the relative physical position of the adapter and may be ascertained by viewing the adapter configuration (see `llc2_config(1)`). For Token Ring or Ethernet, specify an even *sap* value from 2 through 254, or, for Ethernet only, any *type* value from 1519 (0x05ef) through 65535 (0xffff). It is advised to pick a value that is easily recognized when the data capture output is viewed. *frames* is the decimal number of 1,495 bytes packets to transmit. The test will only display a message if a failure occurs.

Loop 4 The `loop4` test sends a TEST frame (no information field) to the broadcast (all 1's) destination MAC address. The source SAP value used is 0x04 (SNA's SAP). Therefore, if SNA is running on the system, the `loop4` test will fail. The destination SAP value is the NULL SAP (0x00). This test finds out who is listening and can receive frames sent out from a node. The verbose (`-v`) option displays the MAC address of responding nodes. All possible responders may not be displayed since the `loop4` test only waits for responses for 2 seconds, but during this time 50-200 nodes may be displayed. The `loop4` test displays information similar to the following example if other nodes are listening and respond (verbose mode):

```
-Attaching
    -Binding
```

```
    -Sending TEST
    -Responders
       1-0000c0c12449
       2-08000e142990
       3-08000e142a51
       4-0000c0450044
       5-0000c0199e46
    -Unbinding
-Detaching
5 nodes responding
```

The errors displayed are the same as for `loop2`.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/network/llc2

See Also `llc2_config(1)`, `llc2(4)`, `attributes(5)`, `llc2(7D)`

Notes For information about how to start the service, see `llc2(7D)`

Name lldpadm – Link Layer Discovery Protocol administration tool

Synopsis

```
lldpadm set-agentprop -p prop[+|-]=value[,...] lldp_agent
lldpadm reset-agentprop -p prop[,...] lldp_agent
lldpadm show-agentprop [[-c] -o field,...] -p prop[,...] [lldp_agent]

lldpadm set-tlvprop -p prop[+|-]=value[,...] tlv_name
lldpadm reset-tlvprop -p prop[,...] tlv_name
lldpadm show-tlvprop [[-c] -o field,...] -p prop[,...] [tlv_name]

lldpadm set-agenttlvprop -p prop[+|-]=value[,...] -a lldp_agent
     tlv_name
lldpadm reset-agenttlvprop -p prop[,...] -a lldp_agent tlv_name
lldpadm show-agenttlvprop [[-c] -o field,...] -p prop[,...]
     [-a lldp_agent] [tlv_name]

lldpadm show-agent [-c] [-s] [-v] -o field,...] [-l|-r] [lldp_agent]
```

Description The `lldpadm` command is used to enable or disable a Link Layer Discovery Protocol (LLDP) agent on a physical datalink. `lldpadm` is also used to configure the behavior of an LLDP agent. The LLDP agent implements the LLDP protocol for a given physical datalink. LLDP is a one-way link layer protocol that allows an IEEE 802 LAN station to advertise the capabilities and current status of the system to other stations attached to the same LAN. The LLDP agent can also receive information about the capabilities and current status of the system associated with a remote station. LLDP agent can either be enabled for transmission only, for reception only, or for both.

Information to be exchanged is packed as a sequence of type, length, and value (TLVs), wherein the type field identifies the type of information, the length field indicates the length of the information field in octets, and the value field contains the information itself.

Operands Each `lldpadm` subcommand operates on one of the following objects:

lldp_agent

An LLDP agent implements the LLDP protocol for a given physical datalink that is connected to IEEE 802 LAN. The only supported physical links are the ones of media type Ethernet. Thus, LLDP can be enabled on all the links displayed in `dladm show-phys` output that are of media type Ethernet. The name of the *lldp_agent* is the name of the datalink itself.

tlv_name

Name of the TLV whose value can be modified. The supported modifiable TLVs are:

- `syscapab` and `mgmtaddr`. These form Global TLVs that are common to all the LLDP agents on the system.
- `pfc`, `ets`, and `appln`. Per-LLDP agent TLVs.

Sub-commands `lldpadm` supports the following subcommands.

`lldpadm set-agentprop|set-ap -p` *prop*[+|-]=*value*[,...] *lldp_agent*
Sets the value of one or more LLDP agent properties to the value specified. If the property takes multiple values then the value should be specified with a comma as the delimiter. The value is always made persistent and thus will be reapplied on system reboot or `lldpd(1M)` daemon restart. The list of properties supported and each property's possible values can be retrieved using `show-agentprop` subcommand.

`-p` *prop*[+|-]=*value*[,...], `--prop` *prop*[+|-]=*value*[,...]
A comma-separated list of properties to be set to the specified values. It also provides the following qualifiers to perform add and delete operations in addition to assignment.

`+` Adds the given value to the current list of value(s).

`-` Removes the given value from the current list of value(s).

`=` Makes a new assignment and removes all the current value(s).

See `EXAMPLES` for more information on how to use the qualifiers.

`lldpadm reset-agentprop|reset-ap -p` *prop*[,...] *lldp_agent*
Resets one or more properties to their default values. The default values for properties can be retrieved using `show-agentprop` subcommand.

`-p` *prop*[,...], `--prop` *prop*[,...]
A comma-separated list of properties to reset.

`lldpadm show-agentprop|show-ap` [`[-c] -o` *field*,...] `-p` *prop*[,...] [*lldp_agent*]
Show the current value of one or more properties, either for all of the LLDP agents or for the specified LLDP agent. Several properties of interest can be retrieved at a time by providing comma-separated property names to `-p` option. If the `-p` option is not specified, all available properties are displayed.

`-o` *field*[,...], `--output` *field*[,...]
A case-insensitive, comma-separated list of output fields to display. The field name must be one of the fields listed below, or the special value `all` to display all fields. For each LLDP agent, the following fields can be displayed:

`AGENT`
The name of the LLDP agent.

`PROPERTY`
The name of the property.

`PERM`
The read/write permissions of the property. The value shown will be `r` (read only), `w` (write only) or `rw` (read/write).

`VALUE`
The current value of the property. If the value is not set, it is shown as `--`. If it is unknown, the value is shown as `?`.

`DEFAULT`
The default value of the property. If the property has no default value, `--` is shown.

`POSSIBLE`
A comma-separated list of the values the property can have. If the values span a numeric range, *min* - *max* might be shown as shorthand. If the possible values are unknown or unbounded, `--` is shown.

`-c`, `--parsable`
Display using a stable machine-parsable format. The `-o` option is required with this option. See "Parsable Output Format", below.

`-p` *prop*[,...], `--prop` *prop*[,...]
A comma-separated list of properties to display.

For the supported list of agent properties, see "Agent Properties" section below.

`lldpadm set-tlvprop`|`set-tp -p` *prop*[+|-]=*value*[,...] *tlv_name*
Sets the value of one or more TLV properties to the value specified. If the property takes multiple values, the value should be specified with a comma as the delimiter. The value is always persisted and will be reapplied on system reboot or `lldpd(1M)` daemon restart. The list of properties supported and each property's possible values can be retrieved using `show-tlvprop` subcommand.

Note – The TLVs modified using this subcommand apply to all the LLDP agents running on the system.

`-p` *prop*[+|-]=*value*[,...], `--prop` *prop*[+|-]=*value*[,...]
See the description of this option under the `set-agentprop` subcommand, above.

`lldpadm reset-tlvprop`|`reset-tp -p` *prop*[,...] *tlv_name*
Resets one or more properties to their default values. The default values for properties can be retrieved using `show-tlvprop` subcommand.

`-p` *prop*[,...], `--prop` *prop*[,...]
A comma-separated list of properties to reset.

`lldpadm show-tlvprop`|`show-tp` [[`-c`] `-o` *field*,...] `-p` *prop*[,...] [*lldp_agent*]
Show the current value of one or more properties, either for all the TLVs or for a specified TLV. Several properties of interest can be retrieved at a time by providing comma-separated property names to `-p` option. If the `-p` option is not specified, all available properties are displayed.

`-o` *field*[,...], `--output` *field*[,...]
A case-insensitive, comma-separated list of output fields to display. The field name must be one of the fields listed below, or the special value `all` to display all fields. For each TLV, the following fields can be displayed:

`TLVNAME`
The name of the TLV.

`PROPERTY`
The name of the property.

`PERM`
The read/write permissions of the property. The value shown will be `r` (read only), `w` (write only) or `rw` (read/write).

`VALUE`
The current value of the property. If the value is not set, it is shown as `--`. If it is unknown, the value is shown as `?`.

`DEFAULT`
The default value of the property. If the property has no default value, `--` is shown.

`POSSIBLE`
A comma-separated list of the values the property can have. If the values span a numeric range, *min* - *max* might be shown as shorthand. If the possible values are unknown or unbounded, `--` is shown.

`-c`, `--parsable`
Display using a stable machine-parsable format. The `-o` option is required with this option. See "Parsable Output Format", below.

`-p` *prop*[,...], `--prop` *prop*[,...]
A comma-separated list of properties to display.

`lldpadm set-agenttlvprop`|`set-atp` `-p` *prop*[+|-]=*value*[,...] `-a` *lldp_agent tlv_name*
Sets the value of one or more TLV properties to the value specified. The `-a` option is mandatory and identifies the name of the agent for which the TLV property needs to be set. In this way, the TLV property modification is reflected only on the specified agent. If the property takes multiple values then the value should be specified with a comma as the delimiter. The value is always made persistent and will be reapplied on system reboot or `lldpd(1M)` daemon restart. The list of properties supported and each property's possible values can be retrieved using `show-agenttlvprop` subcommand.

`-p` *prop*[+|-]=*value*[,...], `--prop` *prop*[+|-]=*value*[,...]
See the description of this option under the `set-agentprop` subcommand, above.

`-a` *lldp_agent*, `--agent`=*lldp_agent*
The name of the LLDP agent for which TLV properties need to be displayed.

`lldpadm reset-agenttlvprop|reset-atp -p` *prop*[,...]=*value*[,...] `-a` *lldp_agent tlv_name*

Resets one or more properties to their default values. The `-a` option is mandatory and identifies the name of the agent for which the TLV property needs to be reset. The default values for properties can be retrieved using `show-agenttlvprop` subcommand.

`-p` *prop*[,...], `--prop` *prop*[,...]

A comma-separated list of properties to display.

`-a` *lldp_agent*, `--agent=`*lldp_agent*

The name of the LLDP agent for which TLV properties need to be displayed.

`lldpadm show-agenttlvprop|show-atp` [`[-c] -o` *field*,...] `-p` *prop*[,...] [`-a` *lldp_agent*] [*tlv_name*]

Show the current value of one or more properties, either for all of the TLVs or for a specified TLV. If a single LLDP agent is not specified (using `-a`), TLV properties for all LLDP agents are displayed. Several properties of interest can be retrieved at a time by providing comma-separated property names to the `-p` option. If the `-p` option is not specified, all available properties are displayed.

`-o` *field*[,...], `--output` *field*[,...]

A case-insensitive, comma-separated list of output fields to display. The field name must be one of the fields listed below, or the special value `all` to display all fields. For each LLDP agent, the following fields can be displayed:

`AGENT`

The name of the LLDP agent.

`TLVNAME`

The name of the TLV.

`PROPERTY`

The name of the property.

`PERM`

The read/write permissions of the property. The value shown will be `r` (read only), `w` (write only) or `rw` (read/write).

`VALUE`

The current value of the property. If the value is not set, it is shown as `--`. If it is unknown, the value is shown as `?`.

`DEFAULT`

The default value of the property. If the property has no default value, `--` is shown.

`POSSIBLE`

A comma-separated list of the values the property can have. If the values span a numeric range, *min* - *max* might be shown as shorthand. If the possible values are unknown or unbounded, `--` is shown.

`-c`, `--parsable`
Display using a stable machine-parsable format. The `-o` option is required with this option. See "Parsable Output Format", below.

`-p` *prop*[,...], `--prop` *prop*[,...]
A comma-separated list of properties to display.

`-a` *lldp_agent*, `--agent=`*lldp_agent*
The name of the LLDP agent for which TLV properties need to be displayed.

For the supported list of TLV properties that apply on a per-LLDP agent basis, see the "Per-LLDP agent TLV properties" section, below.

`lldpadm show-agent` [`-c`] [`-s`] [`-v`] `-o` *field*,...] [`-l` | `-r`] [*lldp_agent*]
Show the information advertised by the specified LLDP agent or information advertised by the adjacent neighbors to the specified LLDP agent. If no LLDP agent is specified, then the local or remote information will be displayed for all the LLDP agents. The information is displayed as a multi-line output, with each line containing information about a single TLV in the following format:

<Name of the TLV> : *<TLV Information expressed as a string>*

The following lines can be displayed:

```
                    Agent:
          Chassis ID Subtype:
                  Chassis ID:
             Port ID Subtype:
                     Port ID:
            Port Description:
                Time to Live:
                 System Name:
          System Description:
      Supported Capabilities:
        Enabled Capabilities:
          Management Address:
          Maximum Frame Size:
                Port VLAN ID:
                VLAN Name/ID:
         VNIC PortID/VLAN ID:
     Aggregation Information:
                 PFC Willing:
                     PFC Cap:
                     PFC MBC:
                  PFC Enable:
                  PFC Pending:  [displayed only when -l is used]
                 ETS Willing:
          ETS Configured CBS:
          ETS Configured TCS:
          ETS Configured PAT:
```

```
        ETS Configured BAT:
        ETS Configured TSA:
       ETS Recommended PAT:
       ETS Recommended BAT:
       ETS Recommended TSA:
Application(s)(ID/Sel/Pri):
                 EVB Mode:
         EVB GID (Station):
                EVB RRREQ:
               EVB RRSTAT:
          EVB GID (Bridge):
                EVB RRCAP:
                EVB RRCTR:
                    EVB R:
                  EVB RTE:
               EVB ROL RWD:
                  EVB RWD:
               EVB ROL RKA:
                  EVB RKA:
  Next Packet Transmission:  [displayed only when -l is used]
   Information Valid Until:  [displayed only when -r is used]
```

In preceding output, the ETS parameters are described as follows:

`ETS Configured CBS`	Indicates whether Credit-Based Shaper is supported.
`ETS Configured TCS`	Number of Traffic Classes supported.
`ETS Configured PAT`	Priority Assignment Table.
`ETS Configured BAT`	Bandwidth Assignment Table.
`ETS Configured TSA`	Transmission Selection Algorithm supported.
`ETS Recommended values`	Values that will be recommended to the peer.

`-c`, `--parsable`
: Display using a stable machine-parsable format. The `-o` option is required with this option. See "Parsable Output Format", below.

`-l`, `--local`
: Displays information advertised by the local LLDP agent. This option is mutually exclusive of the `-r` option.

`-r`, `--remote`
: Displays information advertised by the adjacent neighbors. This option is mutually exclusive of the `-l` option.

`-s`
Show the statistics for the specified LLDP agents or for all the LLDP agents on the system. Every LLDP agent maintains statistical counters that are used to count significant events in the transmit and receive state machines. These counters are defined to be 32-bit unsigned integers.

`-o` *field*[,...], `--output` *field*[,...]
A case-insensitive, comma-separated list of output fields to display. The field name must be one of the fields listed below, or the special value `all` to display all fields. For each TLV, the following fields can be displayed:

`AGENT`
The name of the LLDP agent.

`IFRAMES`
Count of all LLDP frames received by `AGENT`.

`IERR`
Count of all LLDPDUs received with one or more detectable errors.

`IDISCARD`
Count of all LLDPDUs received and then discarded for any of the following reasons:

- incorrectly formed LLDPDUs with respect to the first three mandatory TLVs
- insufficient space to store the incoming LLDPDU

`OFRAMES`
Count of all LLDP frames transmitted by `AGENT`.

`OLENERR`
Count of times the outgoing LLDPDU exceeded the length restrictions of 1500 bytes.

`TLVDISCARD`
Count of all TLVs received and then discarded because the TLVs did not adhere to the TLV usage rules as specified by the standard.

`TLVUNRECOG`
Count of all TLVs received that are not recognized by the LLDP agent.

`AGEOUT`
Count of the times that a neighbor's information has been deleted because of aging.

`-v`, `--verbose`
Displays detailed information.

Parsable Output Format

The `lldpadm show` subcommands have a `-c` option that displays output in a machine-parsable format. The output format is one or more lines of colon (:)-delimited fields. The fields displayed are specific to the subcommand used and are listed under the entry for the `-o` option for a given subcommand. Output includes only those fields requested by means of the `--o` option, in the order requested. Note that the `-o all` option, which displays all the fields for a given subcommand, cannot be used with the parsable output option.

When you request multiple fields, any literal colon characters are escaped by a backslash (\) before being output. Similarly, literal backslash characters are also escaped with a backslash. This escape format is parsable by using shell `read(1)` functions with the environment variable set as `IFS=`: Note that escaping is not done when you request only a single field.

Agent Properties

The following LLDP agent properties are supported:

`mode`
Configures the operation mode of the LLDP agent. Possible values are:

`txonly`
Enables LLDP for transmission only.

`rxonly`
Enables LLDP for receiving only.

`both`
Enables LLDP for both transmission and receiving.

`disable`
Disables LLDP on the LLDP agent.

Every LLDP packet (LLDPDU) transmitted by an LLDP agent contains multiple TLVs. The following four TLVs are mandatory and therefore included in all the LLDPDUs transmitted by an agent configured in the `txonly` or both mode:

`CHASSIS ID`
The value transmitted in the Chassis ID TLV is:

- subtype = 7 (locally assigned)
- Chassis ID = `hostid(1)`

`PORT ID`
The value transmitted in the Port ID TLV is:

- subtype = 3 (MAC address)
- Port ID = primary MAC address of the agent

`TTL`
The duration for which this packet is valid. The default value is 30 seconds.

`End of PDU TLV`
End of PDU indicator.

Optional TLVs that can be advertised are configured using the following properties:

`basic-tlv`
Configures the Basic Management TLVs that should be advertised by the LLDP agent. The possible values are:

`portdesc`
Alphanumeric string that identifies the datalink. Value set to the linkname.

`sysname`
Alphanumeric string that identifies the system. Value set to the output of `'uname -n'`.

`sysdesc`
Alphanumeric string that describes the system. Value set to the output of `'uname -a'`.

`syscapab`
Indicates the systems supported and enabled capabilities.

`mgmtaddr`
Indicates the IP address of the system that can be used by network management.

`dot1-tlv`
Configures the IEEE 802.1 Organizationally Specific TLVs that should be advertised by the LLDP agent. The possible values are:

`vlanname`
Indicates the names and IDs of all the VLANS configured on the datalink.

`pvid`
Indicates the default VLAN ID associated with the given datalink. It corresponds to the `default_tag` datalink property that is managed by means of the `dladm(1M)` utility.

`linkaggr`
Indicates whether underlying datalink is in an aggregation or is capable of being part of an aggregation.

`pfc`
Indicates whether underlying datalink supports PFC (Priority Flow Control) and the priorities for which the PFC pause frame is enabled. Also indicates whether the local endpoint is willing to negotiate the PFC configuration.

`ets-cfg`
Indicates the ETS (Enhanced Transmission Selection) configuration on the host when the underlying physical link supports ETS feature. Also indicates whether the local endpoint is willing to negotiate the ETS configuration.

`appln`
Indicates the priority that will be used by an application.

`dot3-tlv`
Configures the IEEE 802.3 Organizationally Specific TLVs that should be advertised by the LLDP agent. The possible values are:

`max-framesize`
Indicates the maximum supported frame size for the underlying datalink.

`virt-tlv`
Configures the Solaris's Virtualization Specific TLVs that should be advertised by the LLDP agent. The possible values are:

`vnic`
Indicates the MAC address of the Virtual NIC created on top of the underlying physical link. Also indicates any VLAN id associated with the VNIC. See `dladm(1M)` for more information on VNIC.

Global TLVs and Their Properties

The following Global TLV properties are supported:

`syscapab` (TLV name)
This property can one of the following values:

`supported`
Indicates the supported capabilities on the system. The default supported capabilities are: `bridge`, `router`, and `station`.

`enabled`
Indicates the enabled capabilities on the system. The enabled capabilities must be a subset of the supported capabilities.

`mgmtaddr` (TLV name)
This property can have the following value:

`ipaddr`
The IP address(es), either IPv4 or IPv6, associated with the local LLDP agent that will be used to reach higher layer entities to assist discovery by network management.

Per-Agent TLVs and Their Properties

The following Agent TLV properties are supported:

`appln` (TLV name)
This property can have the following value:

`apt`
Configures the Application Priority Table for an Application TLV. One can add or remove entries from this table using the + and - qualifiers. Each entry in the table indicates the application and the priority that will be used for that application. Its value is of the form:

id/*selector*/*priority*

The meaning of the *id* is determined by the *selector* field. The *selector* field can be any one of the following:

- `1` — *id* indicates an Ethertype (an L2 protocol), therefore *id*'s value should be greater than 1536
- `2` — *id* indicates a port number over TCP or SCTP
- `3` — *id* indicates a port number over UDP or DCCP
- `4` — *id* indicates a port number over TCP, SCTP, UDP, or DCCP

The *priority* indicates the priority value (0-7) that will be used for given application.

`ets` (TLV name)
This property can have the following value:

`willing`
Indicates whether the host is willing to accept the peer's ETS recommendation. This property is likely to change in the future.

`pfc` (TLV name)
This property can have the following value:

`willing`
Configures the willingness to accept the configuration from the remote peer and change the operational configuration on the host locally for a Priority-based Flow Control TLV. Its value can be `on` (default) or `off`. This property is subject to change in future releases.

Authorizations The following subcommands require solaris.network.lldp authorization:

- `set-agentprop`
- `reset-agentprop`
- `set-tlvprop`
- `reset-tlvprop`
- `set-agenttlvprop`
- `reset-agenttlvprop`

The various `show-*` subcommands do not need any authorization.

Examples

EXAMPLE 1 Enabling LLDP Protocol on an LLDP Agent

The following command enables the LLDP protocol on an LLDP agent for both transmission and reception of LLDPDUs.

```
# lldpadm set-agentprop -p mode=both net0
```

EXAMPLE 2 Disabling LLDP Protocol on an LLDP Agent

The following command disables the LLDP protocol on an LLDP agent.

```
# lldpadm set-agentprop -p mode=disable net0
```

EXAMPLE 3 Configuring TLVs

The following command configures transmission of the Port Description and System Name TLV.

```
# lldpadm set-agentprop -p basic-tlv=portdesc,sysname net0
```

The following command configures transmission of a VLAN Name TLV and a Link Aggregation TLV.

```
# lldpadm set-agentprop -p dot1-tlv=vlanname net0
# lldpadm set-agentprop -p dot1-tlv+=linkaggr net0
```

EXAMPLE 3 Configuring TLVs *(Continued)*

The following command configures transmission of all `dot3-tlv`s.

```
# lldpadm set-agentprop -p dot3-tlv=all net0
```

All the above `lldpadm` invocations can be combined into the following, single invocation.

```
# lldpadm set-agentprop -p basic-tlv=portdesc,sysname,\
dot1-tlv=vlanname,linkaggr,dot3-tlv=all net0
```

EXAMPLE 4 Disabling Transmission

The following command disables the transmission of all `dot1-tlv`s out of an LLDP agent.

```
# lldpadm set-agentprop -p dot1-tlv=none net0
```

The following command is equivalent to the preceding.

```
# lldpadm reset-agentprop -p dot1-tlv net0
```

EXAMPLE 5 Configuring Enabled Capabilities

The following command configures the enabled capabilities on a system.

```
# lldpadm set-tlvprop -p enabled=router syscapab
```

With this configuration, when an LLDP agent is enabled for advertising a System Capabilities TLV, the adjacent neighbors would learn of the local system's capabilities.

EXAMPLE 6 Configuring a Management Address for Subsequent Advertising

The following command configures the management address that will be advertised by means of the Management Address TLV.

```
# lldpadm set-tlvprop -p ipaddr=192.168.1.2 mgmtaddr
```

Note that this address would be identified as an address associated with the local LLDP agent that will be used to reach higher layer entities to assist discovery by network management.

EXAMPLE 7 Configuring an Application TLV

The following sequence of commands configures the application TLV to advertise the priority that will be used by FCoE.

```
# lldpadm set-agenttlvprop -p apt=8906/1/4 -a net0 appln
# lldpadm show-agenttlvprop -a net0 appln
AGENT    TLVNAME PROPERTY  PERM VALUE        DEFAULT        POSSIBLE
net0     appln   apt         rw 8906/1/4     --             --
```

EXAMPLE 8 Show Local Information Advertised by LLDP Agent

The following commands show, respectively, brief and detailed local information advertised by an LLDP agent.

Brief information:

```
# lldpadm show-agent -l net0
AGENT               CHASSISID           PORTID
net0                004bb87f            00:14:4f:01:77:5d
```

Detailed information:

```
# lldpadm show-agent -lv net0
                    Agent: net0
       Chassis ID Subtype: Local(7)
               Chassis ID: 004bb87f
          Port ID Subtype: MacAddress(3)
                  Port ID: 00:14:4f:01:77:5d
         Port Description: net0
             Time to Live: 81 (seconds)
              System Name: hosta.example.com
       System Description: SunOS 5.11 dcb-clone-x-01-19-11 i86pc
    Supported Capabilities: bridge, router, station
      Enabled Capabilities: router
        Management Address: 192.168.1.2
        Maximum Frame Size: 3000
              Port VLAN ID: --
              VLAN Name/ID: vlan25/25
     VNIC PortID/VLAN ID : --
   Aggregation Information: Capable, Not Aggregated
               PFC Willing: --
                   PFC Cap: --
                   PFC MBC: --
                PFC Enable: --
               PFC Pending: --
       ETS Configured CBS: 0
       ETS Configured TCS: 8
       ETS Configured PAT: 0,1,2,3,4,5,6,7
       ETS Configured BAT: 70,30,0,0,0,0,0,0
       ETS Configured TSA: 2,2,2,2,2,2,2,2
      ETS Recommended PAT: --
      ETS Recommended BAT: --
      ETS Recommended TSA: --
Application(s)(ID/Sel/Pri): --
                 EVB Mode: Station
         EVB GID (Station): Not Supported
                EVB RRREQ: Not Requested
               EVB RRSTAT: RR Unknown
          EVB GID (Bridge): Not Supported
```

EXAMPLE 8 Show Local Information Advertised by LLDP Agent *(Continued)*

```
                EVB RRCAP: Not Supported
                EVB RRCTR: Not Enabled
                    EVB R: 3
                  EVB RTE: 20
              EVB ROL RWD: Local
                  EVB RWD: 20
              EVB ROL RKA: Local
                  EVB RKA: 20
  Next Packet Transmission: 18 (seconds)
```

The above ETS-related information indicates that CBS is not supported, 8 Traffic Classes are supported, there is a 1-to-1 mapping between the priority and the Traffic Class, the bandwidth allocation among the Traffic Classes are 70,30, 000000, and the transmission algorithm used for all the Traffic Classes is ETS.

EXAMPLE 9 Show Remote Information about Adjacent Devices for an LLDP Agent

The following commands show, respectively, brief and detailed remote information about adjacent devices for a given LLDP agent.

Brief information:

```
# lldpadm show-agent -r net0
AGENT      SYSNAME      CHASSISID       PORTID
net0       hostb        0083b390        00:14:4f:01:59:ab
```

Detailed information:

```
# lldpadm show-agent -rv net0
                    Agent: net0
        Chassis ID Subtype: Local(7)
                Chassis ID: 0083b390
           Port ID Subtype: MacAddress(3)
                   Port ID: 00:14:4f:01:59:ab
          Port Description: net0
              Time to Live: 121 (seconds)
               System Name: hostb.example.com
        System Description: SunOS 5.11 dcb-clone-x-01-19-11 i86pc
    Supported Capabilities: bridge, router, station
      Enabled Capabilities: router
        Management Address: 192.168.1.3
        Maximum Frame Size: 3000
              Port VLAN ID: --
              VLAN Name/ID: vlan25/25
      VNIC PortID/VLAN ID : 02:08:20:72:71:31
   Aggregation Information: Capable, Not Aggregated
               PFC Willing: --
```

EXAMPLE 9 Show Remote Information about Adjacent Devices for an LLDP Agent *(Continued)*

```
                  PFC Cap: --
                  PFC MBC: --
               PFC Enable: --
              PFC Pending: --
       ETS Configured CBS: 0
       ETS Configured TCS: 8
       ETS Configured PAT: 0,1,2,3,4,5,6,7
       ETS Configured BAT: 100,0,0,0,0,0,0,0
       ETS Configured TSA: 2,2,2,2,2,2,2,2
      ETS Recommended PAT: --
      ETS Recommended BAT: --
      ETS Recommended TSA: --
Application(s)(ID/Sel/Pri): --
                 EVB Mode: Bridge
        EVB GID (Station): Not Supported
                EVB RRREQ: Not Requested
               EVB RRSTAT: RR Unknown
         EVB GID (Bridge): Not Supported
                EVB RRCAP: Not Supported
                EVB RRCTR: Not Enabled
                    EVB R: 3
                  EVB RTE: 20
              EVB ROL RWD: Local
                  EVB RWD: 20
              EVB ROL RKA: Local
                  EVB RKA: 20
  Information Valid Until: 117 (seconds)
```

EXAMPLE 10 Show LLDP Agent Statistics

The following command displays LLDP agent statistics.

```
# lldpadm show-agent -s net0
AGENT IFRAMES IERR IDISCARD OFRAMES OLENERR TLVDISCARD TLVUNRECOG AGEOUT
net0       44    0        0      57       0          0          0      0
```

Attributes

See `attributes(5)` for descriptions of the following attributes:

`/sbin`

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	service/network/lldp
Interface Stability	Committed

See Also hostid(1), read(1), uname(1), dladm(1M), lldpd(1M), svcadm(1M), attributes(5)

IEEE Std 802.1AB-2009, *IEEE Standard for Local and Metropolitan Area Networks: Station and Media Access Control Connectivity Discovery*

IEEE Draft 802.1Qbb, *Virtual Bridged Local Area Networks - Amendment : Priority-based Flow Control*

IEEE Draft 802.1Qaz, *Virtual Bridged Local Area Networks - Amendment XX: Enhanced Transmission Selection for Bandwidth Sharing between Traffic Classes*

Notes The lldpd(1M) daemon that implements the LLDP protocol must be first enabled before using the `lldpadm` command. The `lldpd` daemon is controlled through the service management facility (SMF) service instance:

```
svc:/network/lldp:default
```

Use svcadm(1M) to enable this service.

Name lldpd – Link Layer Discovery Protocol daemon

Synopsis

```
/usr/lib/lldpd

svc:/network/lldp:default
```

Description `lldpd` is a system daemon that implements the Link Layer Discovery Protocol as specified in IEEE Std. 802.1AB. The daemon also manages LLDP agents on physical datalinks. Please see `lldpadm`(1M) for more information on administering LLDP on LLDP agents.

The `lldpd` daemon is controlled through the service management facility (SMF) service instance:

```
svc:/network/lldp:default
```

This means that `svcadm`(1M) must be used to start, stop, restart, and refresh this daemon. This daemon is enabled by default. To enable LLDP on any of the physical datalinks, this daemon should be running.

The `lldpd` daemon is a project private interface and has no user-accessible options.

Service Properties The following properties control the behavior of LLDP protocol on each of the LLDP agents. These are private properties that are subject to change.

`msgTxInterval`
: Indicates the time interval between the transmission of two consecutive LLDPDUs during normal transmission periods. The default value is 30 seconds and the possible values for this property is 1 through 3600.

`msgFastTx`
: Indicates the time interval between the transmission of two consecutive LLDPDUs during fast transmission periods. The default value is 1 second and the possible values for this property is 1 through 3600.

  Note – Fast transmission of LLDPDUs occur when a new neighbor is detected or some local configuration has changed. During this period LLDPDUs are transmitted at shorter time intervals than during the normal operation. This helps in achieving quick convergence of information.

`msgTxHold`
: The value of the TTL in the LLDPDU transmitted by the LLDP agent is determined by multiplying `msgTxHold` by `msgTxInterval`. So, if this value is set to 2, then the adjacent stations would wait for twice the `msgTxInterval` before aging the entries from this station. The default value is 4 and the possible values for this property is 1 through 100.

`reinitDelay`
: Indicates the delay before reinitialization of LLDP state machines. The default value is 2 seconds and the possible values for this property is 1 through 100.

`txCreditMax`
The maximum number of consecutive LLDPDUs that can be transmitted at any time. The default value is 5 and the possible values for this property is 1 through 10.

`txFastInit`
Indicates the number of LLDPDUs that are transmitted during a fast transmission period. The default value is 4 and the possible values for this property is 1 through 8.

`snmp`
This boolean property turns SNMP support on or off.

Whenever any of the above properties are modified, `svcadm(1M)` must be used to refresh the daemon. On refresh, the daemon rereads these service properties and applies the new values. Note that none of the LLDP agents are restarted on refresh. The only action is that the configuration is reread.

The following property controls whether LLDP agents should be automatically enabled on all physical ports on system.

auto-enable-agents

`yes`
If LLDP is not enabled on any port on the system, enable LLDP on all ports into `both` mode. If LLDP is explicitly enabled in `rxonly` mode, or in `txonly` mode, or in `both` mode on a port, do not automatically enable LLDP on such port. This option is particularly useful when one wants to deploy LLDP in a large datacenter environment.

`force`
Regardless of whether LLDP agent is enabled on ports in `rxonly` mode or `txonly` mode or disabled, enable LLDP agents on all ports into the `both` mode.

`no`
Do not automatically enable LLDP agents on any port. This option does not mean that it will disable LLDP agents on all ports. Rather with this value, LLDP is required to be explicitly enabled on the desired ports. Please see `lldpadm(1M)` on how to enable LLDP agent explicitly.

The `auto-enable-agents` property is set to `yes` by default, which makes LLDP automatically enabled in `both` mode on all physical ports on the system. Whenever this property gets modified, `svcadm(1M)` must be used to restart the daemon. Only refresh of the daemon will not cause the restart of LLDP agents.

Examples

EXAMPLE 1 Enabling the Daemon

The following command enables the `lldpd` daemon.

```
# svcadm enable network/lldp
# svcs network/lldp
STATE          STIME    FMRI
online         Jan_26   svc:/network/lldp:default
```

EXAMPLE 2 Disabling the Daemon

The following command disables the `lldpd` daemon.

```
# svcadm disable network/lldp
# svcs network/lldp
STATE          STIME    FMRI
disabled       5:11:35  svc:/network/lldp:default
```

EXAMPLE 3 Listing All Service Properties

The following command lists all service properties.

```
# svcprop -p lldp network/lldp:default
lldp/msgFastTx count 1
lldp/msgTxHold count 4
lldp/reinitDelay count 2
lldp/snmp boolean false
lldp/stability astring Evolving
lldp/txCreditMax count 5
lldp/txFastInit count 4
lldp/msgTxInterval count 30
```

EXAMPLE 4 Modifying the Service Property, Refreshing the Daemon

The following command sequence changes the LLDPDU transmission interval to every 15 seconds.

```
# svccfg -s lldp:default setprop lldp/msgTxInterval = 15
# svcadm refresh lldp:default
# svcprop -p lldp/msgTxInterval lldp:default
15
```

EXAMPLE 5 Modifying auto-enable-agents, Restarting Daemon

The following command sequence modifies the `auto-enable-agents` property and restarts the LLDP daemon.

```
# svccfg -s lldp:default setprop lldp/auto-enable-agents = no
# svcadm restart lldp:default
```

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	service/network/lldp
Interface Stability	Project Private

See Also `lldpadm(1M)`, `svcadm(1M)`, `attributes(5)`

IEEE Std 802.1AB-2009, *IEEE Standard for Local and Metropolitan Area Networks: Station and Media Access Control Connectivity Discovery*

Name lms – allow applications to access the Intel Active Management Technology

Synopsis `lms`

Description The Local Manageability Service, `lms`, allows applications to access the Intel Active Management Technology (Intel AMT) ME (Management Engine) using the local HECI interface. LMS is dependent on the HECI driver.

To use `lms`, ensure that the Intel AMT Manageability Interface driver is installed and connected to the ME.

`lms` is intended to be run as a daemon. Messages from the service are sent to the `syslog`. LMS messages are marked with a source of "`LMS`".

For an example of how to enable the LMS service using SMF, see `EXAMPLES`.

Examples **EXAMPLE 1** Enabling the LMS service using SMF

The following example enables the LMS service using SMF:

```
svcadm enable network/lms
```

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	x86
Availability	system/management/intel-amt
Interface Stability	Volatile

See Also `attributes(5)`, `smf(5)`

Name locator – location indicator control

Synopsis

```
/usr/sbin/locator [-f | -n]
```

Description The locator command sets or queries the state of the system locator if such a device exists.

Without options, the locator command reports the current state of the system.

The privileges required to use this command are hardware dependent. Typically, only the super user can get or set a locator.

Options The following options are supported:

`-f` Turns the locator off.

`-n` Turns the locator on.

Examples **EXAMPLE 1** Using the `locator` Command on a Platform Which Has a System Locator LED

When issued on a platform which has a system locator LED, the following command turns the locator on:

```
# locator -n
# locator
The 'system' locator is on
```

EXAMPLE 2 Using the `locator` Command on a Platform Which Does Not Have a System Locator LED

When issued on a platform which does not have a system locator LED, the following command attempts to turn the locator on. The command returns an error message.

```
# locator -n
'system' locator not found
```

Exit Status The following exit values are returned:

`0` Successful completion.

`1` Invalid command line input.

`2` The requested operation failed.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/picl

See Also `attributes(5)`

Name lockd – network lock daemon

Synopsis

```
/usr/lib/nfs/lockd [-g graceperiod] [-l listen_min_backlog]
     [-t timeout] [nthreads]
```

Description The `lockd` utility is part of the NFS lock manager, which supports record locking operations on NFS files in NFSv2 and NFSv3. See `fcntl(2)` and `lockf(3C)`. The lock manager provides the following two functions:

- It forwards `fcntl(2)` locking requests for NFS mounted file systems to the lock manager on the NFS server.
- It generates local file locking operations in response to requests forwarded from lock managers running on NFS client machines.

State information kept by the lock manager about these locking requests can be lost if the `lockd` is killed or the operating system is rebooted. Some of this information can be recovered as follows. When the server lock manager restarts, it waits for a grace period for all client-site lock managers to submit reclaim requests. Client-site lock managers, on the other hand, are notified by the status monitor daemon, `statd(1M)`, of the restart and promptly resubmit previously granted lock requests. If the lock daemon fails to secure a previously granted lock at the server site, then it sends `SIGLOST` to a process.

Administrators can make changes to the startup parameters for `lockd` by logging in as root and using the `sharectl(1M)` command.

SMF Management The `lockd` service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/network/nfs/nlockmgr
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. The service's status can be queried using the `svcs(1)` command.

If it is disabled, it will be enabled by `mount_nfs(1M)`, `share_nfs(1M)`, and `automountd(1M)` unless its `application/auto_enable` property is set to `false`.

The `sharectl(1M)` command is used to manipulate the startup SMF parameters for `lockd`. Currently supported parameters are as follows:

`lockd_listen_backlog=`*num*
: Set connection queue length for `lockd` over a connection-oriented transport. The default and minimum value is `32`. Equivalent to `-l` option.

`lockd_servers=`*num*
: Maximum number of concurrent `lockd` requests. The default is `1024`. Equivalent to the *nthreads* operand.

`lockd_retransmit_timeout=`*num*
Retransmit timeout, in seconds, before `lockd` retries. The default is `5`. Equivalent to `-t` option.

`grace_period=`*num*
Grace period, in seconds, that all clients (both NLM and NFSv4) have to reclaim locks after a server reboot. This parameter also controls the NFSv4 lease interval. The default is `90`. Equivalent to `-g` option.

See `EXAMPLES`, below.

Options The following options are supported:

`-g` *graceperiod*
Deprecated in favor of `grace_period`. Specify the number of seconds that all clients (both NLM and NFSv4) have to reclaim locks after the server reboots. It also controls the NFSv4 lease interval. This option is equivalent to the `grace_period` property described above.

`-l` *listen_min_backlog*
Specify the listener backlog (*listen_min_backlog*). *listen_min_backlog* is the number connect requests that are queued and waiting to be processed before new connect requests start to get dropped. Equivalent of the `lockd_listen_backlog` property described above.

`-t` *timeout*
Specify the number of seconds to wait before retransmitting a lock request to the remote server. The default value is 5 seconds. Equivalent of the `lockd_retransmit_timeout` property described above.

Operands *nthreads*
Specify the maximum number of concurrent threads that the server can handle. This concurrency is achieved by up to *nthreads* threads created as needed in the kernel. *nthreads* should be based on the load expected on this server. If *nthreads* is not specified, the maximum number of concurrent threads will default to 1024. Equivalent of the `lockd_servers` property described above.

Examples **EXAMPLE 1** Setting a `lockd` Property

The following command sets `lockd_listen_backlog` to a new value:

```
# sharectl set -p lockd_listen_backlog=40 nfs
```

The `lockd_listen_backlog` and other `lockd` properties are described under `NOTES`, below.

EXAMPLE 2 Getting a `lockd` Property Value

The following command retrieves the value of the `lockd_listen_backlog` property.

```
% sharectl get -p lockd_listen_backlog nfs
lockd_listen_backlog=40
```

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/file-system/nfs

See Also svcs(1), automountd(1M), clear_locks(1M), mount_nfs(1M), share(1M), share_nfs(1M), sharectl(1M), statd(1M), svcadm(1M), fcntl(2), lockf(3C), attributes(5), smf(5)

Notes The lockd daemon does not need to be running for NFSv4.

The lockd daemon might not exist in a future release of Solaris.

Name lockfs – change or report file system locks

Synopsis `/usr/sbin/lockfs [-adefhnuw] [-c` *string*`] [`*file-system*`]...`

Description `lockfs` is used to change and report the status of file system locks. `lockfs` reports the lock status and unlocks the file systems that were improperly left locked.

Using `lockfs` to lock a file system is discouraged because this requires extensive knowledge of SunOS internals to be used effectively and correctly.

When invoked with no arguments, `lockfs` lists the UFS file systems that are locked. If *file-system* is not specified, and `-a` is specified, `lockfs` is run on all mounted, UFS type file systems.

Options The options are mutually exclusive: `wndheuf`. If you do specify more than one of these options on a `lockfs` command line, the utility does not protest and invokes only the last option specified. In particular, you cannot specify a flush (`-f`) and a lock (for example, `-w`) on the same command line. However, all locking operations implicitly perform a flush, so the `-f` is superfluous when specifying a lock.

You must be super-user to use any of the following options, with the exception of `-a`, `-f` and `-v`.

The following options are supported.

`-a`
Apply command to all mounted, UFS type file systems. *file-system* is ignored when `-a` is specified.

`-c` *string*
Accept a string that is passed as the comment field. The `-c` only takes affect when the lock is being set using the `-d`, `-h`, `-n`, `-u`, or `-w` options.

`-d`
Delete-lock (`dlock`) the specified *file-system*. dlock suspends access that could remove directory entries.

`-e`
Error-lock (`elock`) the specified *file-system*. elock blocks all local access to the locked file system and returns `EWOULDBLOCK` on all remote access. File systems are elocked by UFS on detection of internal inconsistency. They may only be unlocked after successful repair by `fsck`, which is usually done automatically (see `mount_ufs(1M)`). elocked file systems can be unmounted.

`-f`
Force a synchronous flush of all data that is dirty at the time `fsflush` is run to its backing store for the named file system (or for all file systems.)

It is a more reliable method than using sync(1M) because it does not return until all possible data has been pushed. In the case of UFS filesystems with logging enabled, the log is also rolled before returning. Additional data can be modified by the time `fsflush` exits, so using one of the locking options is more likely to be of general use.

`-h`
Hard-lock (`hlock`) the specified *file-system*. hlock returns an error on every access to the locked file system, and cannot be unlocked. hlocked file systems can be unmounted.

`-n`
Name-lock (`nlock`) the specified *file-system*. nlock suspends accesses that could change or remove existing directories entries.

`-u`
Unlock (`ulock`) the specified *file-system*. ulock awakens suspended accesses.

`-v`
Enable verbose output.

`-w`
Write-lock (`wlock`) the specified *file-system*. wlock suspends writes that would modify the file system. Access times are not kept while a file system is write-locked.

Operands The following operands are supported.

file-system
A list of path names separated by whitespace. Note that *file-system* can be a directory rather than the specific name of a file system, such as `/` or `/usr`. For example, if you specify `/export/home` as an argument to a `lockfs` command and `/export/home` is mounted on the root (`/`) file system, the `lockfs` command will take effect on the root file system.

Usage See largefile(5) for the description of the behavior of `lockfs` when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

Examples **EXAMPLE 1** Using `lockfs -a`

In the following examples, *filesystem* is the pathname of the mounted-on directory (mount point). `Locktype` is one of "`write`," "*name*," "*delete*," "*hard*," or "*unlock*". When enclosed in parenthesis, the lock is being set. `Comment` is a string set by the process that last issued a lock command.

The following example shows the `lockfs` output when only the `-a` option is specified.

```
example#  /usr/sbin/lockfs -a
```

Filesystem	Locktype	Comment
/	unlock	

EXAMPLE 1 Using `lockfs -a` *(Continued)*

/var	unlock

```
example#
```

EXAMPLE 2 Using `lockfs -w`

The following example shows the `lockfs` output when the `-w` option is used to write lock the `/var` file system and the comment string is set using the `-c` option. The `-a` option is then specified on a separate command line.

```
example#  /usr/sbin/lockfs -w -c "lockfs: write lock example" /var
example#  /usr/sbin/lockfs -a
```

Filesystem	Locktype	Comment
/	unlock	
/var	write	lockfs: write lock example

```
example#
```

EXAMPLE 3 Using `lockfs -u`

The following example shows the `lockfs` output when the `-u` option is used to unlock the `/var` file system and the comment string is set using the `-c` option.

```
example#  /usr/sbin/lockfs -uc "lockfs: unlock example" /var
example#  /usr/sbin/lockfs /var
```

Filesystem	Locktype	Comment
/var	unlock	lockfs: unlock example

```
example#
```

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also

`kill(1)`, `mount_ufs(1M)`, `sync(1M)`, `attributes(5)`, `largefile(5)`, `ufs(7FS)`,

Oracle Solaris Administration: Common Tasks

Diagnostics *file system*`: Not owner`
You must be root to use this command.

file system `:Deadlock condition detected/avoided`
A file is enabled for accounting or swapping, on *file system*.

file system`: Device busy`
Another process is setting the lock on *file system*.

Name lockstat – report kernel lock and profiling statistics

Synopsis

```
lockstat [-ACEHI] [-e event_list] [-i rate]
     [-b | -t | -h | -s depth] [-n nrecords]
     [-l lock [, size]] [-d duration]
     [-f function [, size]] [-T] [-ckgwWRpP] [-D count]
     [-o filename] [-x opt [=val]] command [args]
```

Description The `lockstat` utility gathers and displays kernel locking and profiling statistics. `lockstat` allows you to specify which events to watch (for example, spin on adaptive mutex, block on read access to rwlock due to waiting writers, and so forth) how much data to gather for each event, and how to display the data. By default, `lockstat` monitors all lock contention events, gathers frequency and timing data about those events, and displays the data in decreasing frequency order, so that the most common events appear first.

`lockstat` gathers data until the specified command completes. For example, to gather statistics for a fixed-time interval, use `sleep(1)` as the command, as follows:

```
example# lockstat sleep 5
```

When the `-I` option is specified, `lockstat` establishes a per-processor high-level periodic interrupt source to gather profiling data. The interrupt handler simply generates a `lockstat` event whose caller is the interrupted PC (program counter). The profiling event is just like any other `lockstat` event, so all of the normal `lockstat` options are applicable.

`lockstat` relies on DTrace to modify the running kernel's text to intercept events of interest. This imposes a small but measurable overhead on all system activity, so access to `lockstat` is restricted to super-user by default. The system administrator can permit other users to use `lockstat` by granting them additional DTrace privileges. Refer to the *Solaris Dynamic Tracing Guide* for more information about DTrace security features.

Options The following options are supported:

Event Selection If no event selection options are specified, the default is `-C`.

`-A`
: Watch all lock events. `-A` is equivalent to `-CH`.

`-C`
: Watch contention events.

`-E`
: Watch error events.

`-e` *event_list*
: Only watch the specified events. *event list* is a comma-separated list of events or ranges of events such as 1,4-7,35. Run `lockstat` with no arguments to get a brief description of all events.

`-H`
Watch hold events.

`-I`
Watch profiling interrupt events.

`-i` *rate*
Interrupt rate (per second) for `-I`. The default is 97 Hz, so that profiling doesn't run in lockstep with the clock interrupt (which runs at 100 Hz).

Data Gathering

`-x` *arg*[=*val*]
Enable or modify a DTrace runtime option or D compiler option. The list of options is found in `dtrace(1M)`. Boolean options are enabled by specifying their name. Options with values are set by separating the option name and value with an equals sign (=).

Data Gathering (Mutually Exclusive)

`-b`
Basic statistics: lock, caller, number of events.

`-h`
Histogram: Timing plus time-distribution histograms.

`-s` *depth*
Stack trace: Histogram plus stack traces up to *depth* frames deep.

`-t`
Timing: Basic plus timing for all events [default].

Data Filtering

`-d` *duration*
Only watch events longer than *duration*.

`-f` *func[,size]*
Only watch events generated by *func*, which can be specified as a symbolic name or hex address. *size* defaults to the ELF symbol size if available, or `1` if not.

`-l` *lock[,size]*
Only watch *lock*, which can be specified as a symbolic name or hex address. `size` defaults to the ELF symbol size or `1` if the symbol size is not available.

`-n` *nrecords*
Maximum number of data records.

`-T`
Trace (rather than sample) events [off by default].

Data Reporting

`-c`
Coalesce lock data for lock arrays (for example, `pse_mutex[]`).

`-D` *count*
Only display the top *count* events of each type.

`-g`
Show total events generated by function. For example, if `foo()` calls `bar()` in a loop, the work done by `bar()` counts as work generated by `foo()` (along with any work done by `foo()` itself). The `-g` option works by counting the total number of stack frames in which each function appears. This implies two things: (1) the data reported by `-g` can be misleading if the stack traces are not deep enough, and (2) functions that are called recursively might show greater than 100% activity. In light of issue (1), the default data gathering mode when using `-g` is `-s 50`.

`-k`
Coalesce PCs within functions.

`-o` *filename*
Direct output to *filename*.

`-P`
Sort data by (*count* * *time*) product.

`-p`
Parsable output format.

`-R`
Display rates (events per second) rather than counts.

`-W`
Whichever: distinguish events only by caller, not by lock.

`-w`
Wherever: distinguish events only by lock, not by caller.

Display Formats The following headers appear over various columns of data.

`Count` or `ops/s`
Number of times this event occurred, or the rate (times per second) if `-R` was specified.

`indv`
Percentage of all events represented by this individual event.

`genr`
Percentage of all events generated by this function.

`cuml`
Cumulative percentage; a running total of the individuals.

`rcnt`
Average reference count. This will always be `1` for exclusive locks (mutexes, spin locks, rwlocks held as writer) but can be greater than `1` for shared locks (rwlocks held as reader).

`nsec`
Average duration of the events in nanoseconds, as appropriate for the event. For the profiling event, duration means interrupt latency.

`Lock`
Address of the lock; displayed symbolically if possible.

`CPU+PIL`
CPU plus processor interrupt level (PIL). For example, if CPU 4 is interrupted while at PIL 6, this will be reported as `cpu[4]+6`.

`Caller`
Address of the caller; displayed symbolically if possible.

Examples

EXAMPLE 1 Measuring Kernel Lock Contention

```
example# lockstat sleep 5
Adaptive mutex spin: 2210 events in 5.055 seconds (437 events/sec)

Count indv cuml rcnt     nsec Lock                 Caller
------------------------------------------------------------------------
  269  12%  12% 1.00     2160 service_queue       background+0xdc
  249  11%  23% 1.00       86 service_queue       qenable_locked+0x64
  228  10%  34% 1.00      131 service_queue       background+0x15c
   68   3%  37% 1.00       79 0x30000024070       untimeout+0x1c
   59   3%  40% 1.00      384 0x300066fa8e0       background+0xb0
   43   2%  41% 1.00       30 rqcred_lock         svc_getreq+0x3c
   42   2%  43% 1.00      341 0x30006834eb8       background+0xb0
   41   2%  45% 1.00      135 0x30000021058       untimeout+0x1c
   40   2%  47% 1.00       39 rqcred_lock         svc_getreq+0x260
   37   2%  49% 1.00     2372 0x300068e83d0       hmestart+0x1c4
   36   2%  50% 1.00       77 0x30000021058       timeout_common+0x4
   36   2%  52% 1.00      354 0x300066fa120       background+0xb0
   32   1%  53% 1.00       97 0x30000024070       timeout_common+0x4
   31   1%  55% 1.00     2923 0x300069883d0       hmestart+0x1c4
   29   1%  56% 1.00      366 0x300066fb290       background+0xb0
   28   1%  57% 1.00      117 0x3000001e040       untimeout+0x1c
   25   1%  59% 1.00       93 0x3000001e040       timeout_common+0x4
   22   1%  60% 1.00       25 0x30005161110       sync_stream_buf+0xdc
   21   1%  60% 1.00      291 0x30006834eb8       putq+0xa4
   19   1%  61% 1.00       43 0x3000515dcb0       mdf_alloc+0xc
   18   1%  62% 1.00      456 0x30006834eb8       qenable+0x8
   18   1%  63% 1.00       61 service_queue       queuerun+0x168
   17   1%  64% 1.00      268 0x30005418ee8       vmem_free+0x3c
[...]

R/W reader blocked by writer: 76 events in 5.055 seconds (15 events/sec)

Count indv cuml rcnt     nsec Lock                 Caller
------------------------------------------------------------------------
   23  30%  30% 1.00 22590137 0x300098ba358       ufs_dirlook+0xd0
   17  22%  53% 1.00  5820995 0x3000ad815e8       find_bp+0x10
   13  17%  70% 1.00  2639918 0x300098ba360       ufs_iget+0x198
    4   5%  75% 1.00  3193015 0x300098ba360       ufs_getattr+0x54
```

EXAMPLE 1 Measuring Kernel Lock Contention *(Continued)*

```
    3   4%  79% 1.00  7953418 0x3000ad817c0       find_bp+0x10
    3   4%  83% 1.00   935211 0x3000ad815e8       find_read_lof+0x14
    2   3%  86% 1.00 16357310 0x300073a4720       find_bp+0x10
    2   3%  88% 1.00  2072433 0x300073a4720       find_read_lof+0x14
    2   3%  91% 1.00  1606153 0x300073a4370       find_bp+0x10
    1   1%  92% 1.00  2656909 0x300107e7400       ufs_iget+0x198
[...]
```

EXAMPLE 2 Measuring Hold Times

```
example# lockstat -H -D 10 sleep 1
Adaptive mutex spin: 513 events

Count indv cuml rcnt     nsec Lock                Caller
-------------------------------------------------------------------------------
  480   5%   5% 1.00     1136 0x300007718e8       putnext+0x40
  286   3%   9% 1.00      666 0x3000077b430       getf+0xd8
  271   3%  12% 1.00      537 0x3000077b430       msgio32+0x2fc
  270   3%  15% 1.00     3670 0x300007718e8       strgetmsg+0x3d4
  270   3%  18% 1.00     1016 0x300007c38b0       getq_noenab+0x200
  264   3%  20% 1.00     1649 0x300007718e8       strgetmsg+0xa70
  216   2%  23% 1.00     6251 tcp_mi_lock         tcp_snmp_get+0xfc
  206   2%  25% 1.00      602 thread_free_lock    clock+0x250
  138   2%  27% 1.00      485 0x300007c3998       putnext+0xb8
  138   2%  28% 1.00     3706 0x300007718e8       strrput+0x5b8
-------------------------------------------------------------------------------
[...]
```

EXAMPLE 3 Measuring Hold Times for Stack Traces Containing a Specific Function

```
example# lockstat -H -f tcp_rput_data -s 50 -D 10 sleep 1
Adaptive mutex spin: 11 events in 1.023 seconds (11
events/sec)

-------------------------------------------------------------------------------
Count indv cuml rcnt     nsec Lock                   Caller
    9  82%  82% 1.00     2540 0x30000031380          tcp_rput_data+0x2b90

      nsec ------ Time Distribution ------ count     Stack
       256 |@@@@@@@@@@@@@@@@@               5         tcp_rput_data+0x2b90
       512 |@@@@@@                          2         putnext+0x78
      1024 |@@@                             1         ip_rput+0xec4
      2048 |                                0         _c_putnext+0x148
      4096 |                                0         hmeread+0x31c
      8192 |                                0         hmeintr+0x36c
     16384 |@@@                             1
sbus_intr_wrapper+0x30
[...]
```

EXAMPLE 3 Measuring Hold Times for Stack Traces Containing a Specific Function *(Continued)*

```
Count indv cuml rcnt     nsec Lock                   Caller
    1   9%  91% 1.00     1036 0x30000055380          freemsg+0x44

      nsec ------ Time Distribution ------ count     Stack
      1024 |@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ 1        freemsg+0x44
                                                     tcp_rput_data+0x2fd0
                                                     putnext+0x78
                                                     ip_rput+0xec4
                                                     _c_putnext+0x148
                                                     hmeread+0x31c
                                                     hmeintr+0x36c

sbus_intr_wrapper+0x30
-------------------------------------------------------------------------------
[...]
```

EXAMPLE 4 Basic Kernel Profiling

For basic profiling, we don't care whether the profiling interrupt sampled `foo()+0x4c` or `foo()+0x78`; we care only that it sampled somewhere in `foo()`, so we use `-k`. The CPU and PIL aren't relevant to basic profiling because we are measuring the system as a whole, not a particular CPU or interrupt level, so we use `-W`.

```
example# lockstat -kIW -D 20 ./polltest
Profiling interrupt: 82 events in 0.424 seconds (194
events/sec)

Count indv cuml rcnt     nsec Hottest CPU+PIL     Caller
-------------------------------------------------------------------------
    8  10%  10% 1.00      698 cpu[1]              utl0
    6   7%  17% 1.00      299 cpu[0]              read
    5   6%  23% 1.00      124 cpu[1]              getf
    4   5%  28% 1.00      327 cpu[0]              fifo_read
    4   5%  33% 1.00      112 cpu[1]              poll
    4   5%  38% 1.00      212 cpu[1]              uiomove
    4   5%  43% 1.00      361 cpu[1]              mutex_tryenter
    3   4%  46% 1.00      682 cpu[0]              write
    3   4%  50% 1.00       89 cpu[0]              pcache_poll
    3   4%  54% 1.00      118 cpu[1]              set_active_fd
    3   4%  57% 1.00      105 cpu[0]              syscall_trap32
    3   4%  61% 1.00      640 cpu[1]              (usermode)
    2   2%  63% 1.00      127 cpu[1]              fifo_poll
    2   2%  66% 1.00      300 cpu[1]              fifo_write
    2   2%  68% 1.00      669 cpu[0]              releasef
    2   2%  71% 1.00      112 cpu[1]              bt_getlowbit
    2   2%  73% 1.00      247 cpu[1]              splx
```

EXAMPLE 4 Basic Kernel Profiling *(Continued)*

```
    2   2%  76% 1.00      503 cpu[0]              mutex_enter
    2   2%  78% 1.00      467 cpu[0]+10           disp_lock_enter
    2   2%  80% 1.00      139 cpu[1]              default_copyin
-------------------------------------------------------------------------
[...]
```

EXAMPLE 5 Generated-load Profiling

In the example above, 5% of the samples were in `poll()`. This tells us how much time was spent inside `poll()` itself, but tells us nothing about how much work was *generated* by `poll()`; that is, how much time we spent in functions called by `poll()`. To determine that, we use the `-g` option. The example below shows that although `polltest` spends only 5% of its time in `poll()` itself, `poll()`-induced work accounts for 34% of the load.

Note that the functions that generate the profiling interrupt (`lockstat_intr()`, `cyclic_fire()`, and so forth) appear in every stack trace, and therefore are considered to have generated 100% of the load. This illustrates an important point: the generated load percentages do *not* add up to 100% because they are not independent. If 72% of all stack traces contain both `foo()` and `bar()`, then both `foo()` and `bar()` are 72% load generators.

```
example# lockstat -kgIW -D 20 ./polltest
Profiling interrupt: 80 events in 0.412 seconds (194 events/sec)

Count genr cuml rcnt     nsec Hottest CPU+PIL     Caller
-------------------------------------------------------------------------
   80 100% ---- 1.00      310 cpu[1]              lockstat_intr
   80 100% ---- 1.00      310 cpu[1]              cyclic_fire
   80 100% ---- 1.00      310 cpu[1]              cbe_level14
   80 100% ---- 1.00      310 cpu[1]              current_thread
   27  34% ---- 1.00      176 cpu[1]              poll
   20  25% ---- 1.00      221 cpu[0]              write
   19  24% ---- 1.00      249 cpu[1]              read
   17  21% ---- 1.00      232 cpu[0]              write32
   17  21% ---- 1.00      207 cpu[1]              pcache_poll
   14  18% ---- 1.00      319 cpu[0]              fifo_write
   13  16% ---- 1.00      214 cpu[1]              read32
   10  12% ---- 1.00      208 cpu[1]              fifo_read
   10  12% ---- 1.00      787 cpu[1]              utl0
    9  11% ---- 1.00      178 cpu[0]              pcacheset_resolve
    9  11% ---- 1.00      262 cpu[0]              uiomove
    7   9% ---- 1.00      506 cpu[1]              (usermode)
    5   6% ---- 1.00      195 cpu[1]              fifo_poll
    5   6% ---- 1.00      136 cpu[1]              syscall_trap32
    4   5% ---- 1.00      139 cpu[0]              releasef
    3   4% ---- 1.00      277 cpu[1]              polllock
-------------------------------------------------------------------------
[...]
```

EXAMPLE 6 Gathering Lock Contention and Profiling Data for a Specific Module

In this example we use the `-f` option not to specify a single function, but rather to specify the entire text space of the `sbus` module. We gather both lock contention and profiling statistics so that contention can be correlated with overall load on the module.

```
example# modinfo | grep sbus
 24 102a8b6f   b8b4  59   1  sbus (SBus (sysio) nexus driver)

example# lockstat -kICE -f 0x102a8b6f,0xb8b4 sleep 10
Adaptive mutex spin: 39 events in 10.042 seconds (4 events/sec)

Count indv cuml rcnt     nsec Lock               Caller
-------------------------------------------------------------------------------
   15  38%  38% 1.00      206 0x30005160528      sync_stream_buf
    7  18%  56% 1.00       14 0x30005160d18      sync_stream_buf
    6  15%  72% 1.00       27 0x300060c3118      sync_stream_buf
    5  13%  85% 1.00       24 0x300060c3510      sync_stream_buf
    2   5%  90% 1.00       29 0x300060c2d20      sync_stream_buf
    2   5%  95% 1.00       24 0x30005161cf8      sync_stream_buf
    1   3%  97% 1.00       21 0x30005161110      sync_stream_buf
    1   3% 100% 1.00       23 0x30005160130      sync_stream_buf
[...]

Adaptive mutex block: 9 events in 10.042 seconds (1 events/sec)

Count indv cuml rcnt     nsec Lock               Caller
-------------------------------------------------------------------------------
    4  44%  44% 1.00   156539 0x30005160528      sync_stream_buf
    2  22%  67% 1.00   763516 0x30005160d18      sync_stream_buf
    1  11%  78% 1.00   462130 0x300060c3510      sync_stream_buf
    1  11%  89% 1.00   288749 0x30005161110      sync_stream_buf
    1  11% 100% 1.00  1015374 0x30005160130      sync_stream_buf
[...]

Profiling interrupt: 229 events in 10.042 seconds (23 events/sec)

Count indv cuml rcnt     nsec Hottest CPU+PIL    Caller

-------------------------------------------------------------------------------
   89  39%  39% 1.00      426 cpu[0]+6           sync_stream_buf
   64  28%  67% 1.00      398 cpu[0]+6           sbus_intr_wrapper
   23  10%  77% 1.00      324 cpu[0]+6           iommu_dvma_kaddr_load
   21   9%  86% 1.00      512 cpu[0]+6           iommu_tlb_flush
   14   6%  92% 1.00      342 cpu[0]+6           iommu_dvma_unload
   13   6%  98% 1.00      306 cpu[1]             iommu_dvma_sync
    5   2% 100% 1.00      389 cpu[1]             iommu_dma_bindhdl
-------------------------------------------------------------------------------
[...]
```

EXAMPLE 7 Determining the Average PIL (processor interrupt level) for a CPU

```
example# lockstat -Iw -l cpu[3] ./testprog

Profiling interrupt: 14791 events in 152.463 seconds (97 events/sec)

Count indv cuml rcnt     nsec CPU+PIL             Hottest Caller

-------------------------------------------------------------------------
13641  92%  92% 1.00      253 cpu[3]              (usermode)
  579   4%  96% 1.00      325 cpu[3]+6            ip_ocsum+0xe8
  375   3%  99% 1.00      411 cpu[3]+10           splx
  154   1% 100% 1.00      527 cpu[3]+4            fas_intr_svc+0x80
   41   0% 100% 1.00      293 cpu[3]+13           send_mondo+0x18
    1   0% 100% 1.00      266 cpu[3]+12           zsa_rxint+0x400
-------------------------------------------------------------------------
[...]
```

EXAMPLE 8 Determining which Subsystem is Causing the System to be Busy

```
example# lockstat -s 10 -I sleep 20

Profiling interrupt: 4863 events in 47.375 seconds (103 events/sec)

Count indv cuml rcnt     nsec CPU+PIL          Caller

-------------------------------------------------------------------------
1929   40%  40% 0.00     3215 cpu[0]           usec_delay+0x78
  nsec ------ Time Distribution ------ count   Stack
  4096 |@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@  1872    ata_wait+0x90
  8192 |                                 27      acersb_get_intr_status+0x34
 16384 |                                 29      ata_set_feature+0x124
 32768 |                                 1       ata_disk_start+0x15c
                                                 ata_hba_start+0xbc
                                                 ghd_waitq_process_and \
                                                 _mutex_hold+0x70
                                                 ghd_waitq_process_and \
                                                 _mutex_exit+0x4
                                                 ghd_transport+0x12c
                                                 ata_disk_tran_start+0x108
-------------------------------------------------------------------------
[...]
```

Attributes

See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/dtrace

See Also dtrace(1M), plockstat(1M), attributes(5), lockstat(7D), mutex(9F), rwlock(9F)

Solaris Dynamic Tracing Guide

Notes The profiling support provided by `lockstat -I` replaces the old (and undocumented) `/usr/bin/kgmon` and `/dev/profile`.

Tail-call elimination can affect call sites. For example, if `foo()+0x50` calls `bar()` and the last thing `bar()` does is call `mutex_exit()`, the compiler can arrange for `bar()` to branch to `mutex_exit()` with a return address of `foo()+0x58`. Thus, the `mutex_exit()` in `bar()` will appear as though it occurred at `foo()+0x58`.

The PC in the stack frame in which an interrupt occurs can be bogus because, between function calls, the compiler is free to use the return address register for local storage.

When using the `-I` and `-s` options together, the interrupted PC will usually not appear anywhere in the stack since the interrupt handler is entered asynchronously, not by a function call from that PC.

The `lockstat` technology is provided on an as-is basis. The format and content of `lockstat` output reflect the current Solaris kernel implementation and are therefore subject to change in future releases.

Name lofiadm – administer files available as block devices through lofi

Synopsis

```
/usr/sbin/lofiadm -a | -r file [device]

/usr/sbin/lofiadm -c crypto_algorithm -a file | -r [device]

/usr/sbin/lofiadm -c crypto_algorithm -k raw_key_file
     -a | -r file [device]

/usr/sbin/lofiadm -c crypto_algorithm -T token_key -a | -r file [device]

/usr/sbin/lofiadm -c crypto_algorithm -T token_key
     -k wrapped_key_file -a | -r file [device]

/usr/sbin/lofiadm -c crypto_algorithm -e -a file [device]

/usr/sbin/lofiadm -C algorithm [-s segment_size] file

/usr/sbin/lofiadm -d file | device

/usr/sbin/lofiadm -U file

/usr/sbin/lofiadm [ file | device]

/usr/sbin/lofiadm -r device]
```

Description `lofiadm` administers `lofi`, the loopback file driver. `lofi` allows a file to be associated with a block device. That file can then be accessed through the block device. This is useful when the file contains an image of some filesystem (such as a CD-ROM image), because the block device can then be used with the normal system utilities for mounting, checking or repairing file systems. See `fsck(1M)` and `mount(1M)`.

Use `lofiadm` to add a file as a loopback device, remove such an association, or display information about the current associations.

Two types of loopback devices can be created: a normal read-write loopback device and a removable loopback device. They differ in the following ways.

Firstly, a file cannot be dissociated from a normal loopback device during its lifetime. By contrast, a file *can* be dissociated from a removable loopback device, which leaves it as an empty loopback device. Following disassociation, a different file can be associated with it. Note that `eject(1)` should be used to dissociate a file from a removable device.

Secondly, there is one-to-one mapping between a normal loopback device and its associated file. By contrast, a single file can be associated with multiple removable loopback devices at the same time.

Thirdly, a normal loopback device is writable. A removable loopback device is read-only.

The number of potential `lofi` devices is limited by the `zone.max-lofi` rctl, which can be set by means of `zonecfg(1M)` in the global zone. See `resource_controls(5)` for a description of `zone.max-lofi`.

Encryption and compression options are mutually exclusive on the command line. Further, an encrypted file cannot be compressed later, nor can a compressed file be encrypted later.

In the global zone, `lofiadm` can be used on both the global zone devices and all devices owned by other non-global zones on the system.

Options The following options are supported:

`-a` | `-r` *file* [*device*]
Add *file* as a normal loopback device, when `-a` is specified or a removable loopback device, when `-r` is specified.

If *device* is not specified, a non-existing device is picked.

If *device* is specified, `lofiadm` attempts to assign it to *file*. If `-a` is specified, *device* must not exist or `lofiadm` will fail. If `-r` is specified, `lofiadm` will fail if *device* exists and is not an empty removable loopback device.

`-C` {*gzip* | *gzip-N* | *lzma*}
Compress the file with the specified compression algorithm.

The `gzip` compression algorithm uses the same compression as the open-source `gzip` command. You can specify the `gzip` level by using the value `gzip-`*N* where *N* is 6 (fast) or 9 (best compression ratio). Currently, `gzip`, without a number, is equivalent to `gzip-6` (which is also the default for the `gzip` command).

lzma stands for the LZMA (Lempel-Ziv-Markov) compression algorithm.

Note that you cannot write to a compressed file, nor can you mount a compressed file read/write.

Note that removal by file will fail if multiple devices is currently associated with it. Use removal by device in this case.

`-d` *file* | *device*
Remove an association by *file* or *device* name, if the associated block device is not busy, and deallocates the block device.

`-s` *segment_size*
The segment size to use to divide the file being compressed. *segment_size* can be an integer multiple of 512.

`-U` *file*
Uncompress a compressed file.

The following options are used when the file is encrypted:

`-c` *crypto_algorithm*
Select the encryption algorithm. The algorithm must be specified when encryption is enabled because the algorithm is not stored in the disk image.

If none of `-e`, `-k`, or `-T` is specified, `lofiadm` prompts for a passphrase, with a minimum length of eight characters, to be entered . The passphrase is used to derive a symmetric encryption key using PKCS#5 PBKD2.

`-k` *raw_key_file* | *wrapped_key_file*
Path to raw or wrapped symmetric encryption key. If a PKCS#11 object is also given with the `-T` option, then the key is wrapped by that object. If `-T` is not specified, the key is used raw.

`-T` *token_key*
The key in a PKCS#11 token to use for the encryption or for unwrapping the key file.

If `-k` is also specified, `-T` identifies the unwrapping key, which must be an RSA private key.

`-e`
Generate an ephemeral symmetric encryption key. Note that you cannot use `-e` together with `-r`.

`-r` [*device*]
Create an empty removable loopback device. A file can be associated with that device at a later time.

If *device* is not specified, a non-existing device is selected.

If *device* is specified, `lofiadm` attempts to select it. *device* must not exist or `lofiadm` will fail.

Operands The following operands are supported:

crypto_algorithm
One of: `aes-128-cbc`, `aes-192-cbc`, `aes-256-cbc`, `des3-cbc`, `blowfish-cbc`.

device
Display the file name associated with the block device *device*.

Without arguments, print a list of the current associations. Filenames must be valid absolute pathnames.

When a file is added, it is opened for reading or writing by root. Any restrictions apply (such as restricted root access over NFS). The file is held open until the association is removed. It is not actually accessed until the block device is used, so it will never be written to if the block device is only opened read-only.

Note that the filename might appear as a question mark (`?`) if it is not possible to resolve the path in the current context (for example, if it is an NFS path in a non-global zone). Or, the filename might appear as a dash (`-`) if the device is an empty removable loopback device.

file
Display all block device(s) associated with *file*.

raw_key_file
Path to a file of the appropriate length, in bits, to use as a raw symmetric encryption key.

token_key
PKCS#11 token object in the format:

token_name:*manufacturer_id*:*serial_number*:*key_label*

All but the key label are optional and can be empty. For example, to specify a token object with only its key label `MylofiKey`, use:

```
-T :::MylofiKey
```

wrapped_key_file
Path to file containing a symmetric encryption key wrapped by the RSA private key specified by `-T`.

Examples

EXAMPLE 1 Mounting an Existing CD-ROM Image

You should ensure that Solaris understands the image before creating the CD. `lofi` allows you to mount the image and see if it works.

This example mounts an existing CD-ROM image (`sparc.iso`), of the `Red Hat 6.0 CD` which was downloaded from the Internet. It was created with the `mkisofs` utility from the Internet.

Use `lofiadm` to attach a block device to it:

```
# lofiadm -a /home/mike_s/RH6.0/sparc.iso
/dev/lofi/1
```

`lofiadm` picks the device and prints the device name to the standard output. You can run `lofiadm` again by issuing the following command:

```
# lofiadm
Block Device     File                           Options
/dev/lofi/1      /home/mike_s/RH6.0/sparc.iso   -
```

Or, you can give it one name and ask for the other, by issuing the following command:

```
# lofiadm /dev/lofi/1
/home/mike_s/RH6.0/sparc.iso
```

Use the `mount` command to mount the image:

```
# mount -F hsfs -o ro /dev/lofi/1 /mnt
```

Check to ensure that Solaris understands the image:

```
# df -k /mnt
Filesystem            kbytes    used   avail capacity  Mounted on
/dev/lofi/1           512418  512418       0   100%    /mnt
# ls /mnt
./            RedHat/       doc/          ls-lR         rr_moved/
../           TRANS.TBL     dosutils/     ls-lR.gz      sbin@
.buildlog     bin@          etc@          misc/         tmp/
```

EXAMPLE 1 Mounting an Existing CD-ROM Image *(Continued)*

```
COPYING       boot/         images/       mnt/          usr@
README        boot.cat*     kernels/      modules/
RPM-PGP-KEY   dev@          lib@          proc/
```

Solaris can mount the CD-ROM image, and understand the filenames. The image was created properly, and you can now create the CD-ROM with confidence.

As a final step, unmount and detach the images:

```
# umount /mnt
# lofiadm -d /dev/lofi/1
# lofiadm
Block Device             File             Options
```

EXAMPLE 2 Making a UFS Filesystem on a File

Making a UFS filesystem on a file can be useful, particularly if a test suite requires a scratch filesystem. It can be painful (or annoying) to have to repartition a disk just for the test suite, but you do not have to. You can `newfs` a file with `lofi`

Create the file:

```
# mkfile 35m /export/home/test
```

Attach it to a block device. You also get the character device that `newfs` requires, so `newfs` that:

```
# lofiadm -a /export/home/test
/dev/lofi/1
# newfs /dev/rlofi/1
newfs: construct a new file system /dev/rlofi/1: (y/n)? y
/dev/rlofi/1:   71638 sectors in 119 cylinders of 1 tracks, 602 sectors
        35.0MB in 8 cyl groups (16 c/g, 4.70MB/g, 2240 i/g)
super-block backups (for fsck -F ufs -o b=#) at:
 32, 9664, 19296, 28928, 38560, 48192, 57824, 67456,
```

Note that `ufs` might not be able to use the entire file. Mount and use the filesystem:

```
# mount /dev/lofi/1 /mnt
# df -k /mnt
Filesystem            kbytes    used   avail capacity  Mounted on
/dev/lofi/1            33455       9   30101     1%    /mnt
# ls /mnt
./           ../          lost+found/
# umount /mnt
# lofiadm -d /dev/lofi/1
```

EXAMPLE 3 Creating a PC (FAT) File System on a Unix File

The following series of commands creates a FAT file system on a Unix file. The file is associated with a block device created by `lofiadm`.

EXAMPLE 3 Creating a PC (FAT) File System on a Unix File *(Continued)*

```
# mkfile 10M /export/test/testfs
# lofiadm -a /export/test/testfs
/dev/lofi/1
Note use of rlofi, not lofi, in following command.
# mkfs -F pcfs -o nofdisk,size=20480 /dev/rlofi/1
Construct a new FAT file system on /dev/rlofi/1: (y/n)? y
# mount -F pcfs /dev/lofi/1 /mnt
# cd /mnt
# df -k .
Filesystem            kbytes    used   avail capacity  Mounted on
/dev/lofi/1            10142       0   10142     0%    /mnt
```

EXAMPLE 4 Compressing an Existing CD-ROM Image

The following example illustrates compressing an existing CD-ROM image (`solaris.iso`), verifying that the image is compressed, and then uncompressing it.

```
# lofiadm -C gzip /export/home/solaris.iso
```

Use `lofiadm` to attach a block device to it:

```
# lofiadm -a /export/home/solaris.iso
  /dev/lofi/1
```

Check if the mapped image is compressed:

```
# lofiadm
Block Device      File                           Options
/dev/lofi/1       /export/home/solaris.iso       Compressed(gzip)
/dev/lofi/2       /export/home/regular.iso       -
```

Unmap the compressed image and uncompress it:

```
# lofiadm -d /dev/lofi/1
# lofiadm -U /export/home/solaris.iso
```

EXAMPLE 5 Creating an Encrypted UFS File System on a File

This example is similar to the example of making a UFS filesystem on a file, above.

Create the file:

```
# mkfile 35m /export/home/test
```

Attach the file to a block device and specify that the file image is encrypted. As a result of this command, you obtain the character device, which is subsequently used by `newfs`:

```
# lofiadm -c aes-256-cbc -a /export/home/secrets
Enter passphrase: My-M0th3r;l0v3s_m3+4lw4ys!          (not echoed)
Re-enter passphrase: My-M0th3r;l0v3s_m3+4lw4ys!       (not echoed)
```

EXAMPLE 5 Creating an Encrypted UFS File System on a File *(Continued)*

```
/dev/lofi/1

# newfs /dev/rlofi/1
newfs: construct a new file system /dev/rlofi/1: (y/n)? y
/dev/rlofi/1:   71638 sectors in 119 cylinders of 1 tracks, 602 sectors
       35.0MB in 8 cyl groups (16 c/g, 4.70MB/g, 2240 i/g)
super-block backups (for fsck -F ufs -o b=#) at:
32, 9664, 19296, 28928, 38560, 48192, 57824, 67456,
```

The mapped file system shows that encryption is enabled:

```
# lofiadm
Block Device    File                    Options
/dev/lofi/1     /export/home/secrets    Encrypted
```

Mount and use the filesystem:

```
# mount /dev/lofi/1 /mnt
# cp moms_secret_*_recipe /mnt
# ls /mnt
./           moms_secret_cookie_recipe    moms_secret_soup_recipe
../          moms_secret_fudge_recipe     moms_secret_stuffing_recipe
lost+found/  moms_secret_meatloaf_recipe  moms_secret_waffle_recipe
# umount /mnt
# lofiadm -d /dev/lofi/1
```

Subsequent attempts to map the filesystem with the wrong key or the wrong encryption algorithm will fail:

```
# lofiadm -c blowfish-cbc -a /export/home/secrets
Enter passphrase: mommy                                 (not echoed)
Re-enter passphrase: mommy                              (not echoed)
lofiadm: could not map file /root/lofi: Invalid argument
# lofiadm
Block Device    File                   Options
#
```

Attempts to map the filesystem without encryption will succeed, however attempts to mount and use the filesystem will fail:

```
# lofiadm -a /export/home/secrets
/dev/lofi/1
# lofiadm
Block Device    File                    Options
/dev/lofi/1     /export/home/secrets    -
# mount /dev/lofi/1 /mnt
mount: /dev/lofi/1 is not this fstype
#
```

EXAMPLE 6 Manipulating a Removable Loopback Device

The following example illustrates how to create an empty removable loopback device, associate a file with it, and then dissociate the file from the device.

Use `lofiadm` to create an empty removable loopback device:

```
# lofiadm -r
/dev/lofi/1
```

Verify that the device has been created:

```
# lofiadm
Block Device      File                      Options
/dev/lofi/1       -                         Removable,Readonly
```

Use `lofiadm` to associate a file with the device:

```
# lofiadm -r /export/home/solaris.iso /dev/lofi/1
/dev/lofi/1
```

Verify that the association has succeeded:

```
# lofiadm
Block Device      File                            Options
/dev/lofi/1       /export/home/solaris.iso        Removable,Readonly
```

Use `eject` to dissociate the file from the device:

```
# eject /dev/lofi/1
```

Verify that the dissociation succeeded:

```
# lofiadm
Block Device      File                            Options
/dev/lofi/1       -                               Removable,Readonly
```

Environment Variables

See `environ(5)` for descriptions of the following environment variables that affect the execution of `lofiadm`: `LC_CTYPE`, `LC_MESSAGES` and `NLSPATH`.

Exit Status

The following exit values are returned:

`0`
: Successful completion.

`>0`
: An error occurred.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE

Availability	system/core-os

See Also eject(1), fsck(1M), mount(1M), mount_ufs(1M), newfs(1M), zonecfg(1M), attributes(5), resource_controls(5), lofi(7D), lofs(7FS)

Notes Just as you would not directly access a disk device that has mounted file systems, you should not access a file associated with a block device except through the `lofi` file driver. It might also be appropriate to ensure that the file has appropriate permissions to prevent such access.

The abilities of `lofiadm`, and who can use them, are controlled by the permissions of `/dev/lofictl`. Read-access allows query operations, such as listing all the associations. Write-access is required to do any state-changing operations, like adding an association. As shipped, `/dev/lofictl` is owned by `root`, in group `sys`, and mode `0644`, so all users can do query operations but only root can change anything. The administrator can give users write-access, allowing them to add or delete associations, but that is very likely a security hole and should probably only be given to a trusted group.

When mounting a filesystem image, take care to use appropriate mount options. In particular, the `nosuid` mount option might be appropriate for UFS images whose origin is unknown. Also, some options might not be useful or appropriate, like `logging` or `forcedirectio` for UFS. For compatibility purposes, a raw device is also exported along with the block device. For example, newfs(1M) requires one.

The output of `lofiadm` (without arguments) might change in future releases.

Name logadm – manage endlessly growing log files

Synopsis

```
logadm

logadm [-options] logname...
```

Description `logadm` is a general log rotation tool that is suitable for running from `cron(1M)`.

Without arguments, `logadm` reads the `/etc/logadm.conf` file, and for every entry found in that file checks the corresponding log file to see if it should be rotated. Typically this check is done each morning by an entry in the root's `crontab`.

If the *logname* argument is specified, `logadm` renames the corresponding log file by adding a suffix so that the most recent log file ends with `.0` (that is, *logfile*`.0`), the next most recent ends with `.1` (that is, *logfile*`.1`), and so forth. By default, ten versions of old log files are kept (that is, *logfile*`.0` through *logfile*`.9`). At the point when what would be the eleventh file is logged, `logadm` automatically deletes the oldest version to keep the count of files at ten.

`logadm` takes a number of *options*. You can specify these options on the command line or in the `/etc/logadm.conf` file. The `logadm` command searches `/etc/logadm.conf` for lines of the form *logname options*

logname
: Identifies an entry in `/etc/logadm.conf`. This can be a name or the pathname of the log file. If you specify a log file, rather than a name, for this field, it must be a fully qualified pathname.

options
: Identifies command line options exactly as they would be entered on the command line. This allows commonly used log rotation policies to be stored in the `/etc/logadm.conf` file. See `EXAMPLES`.

  If *options* are specified both in `/etc/logadm.conf` and on the command line, those in the `/etc/logadm.conf` file are applied first. Therefore, the command line options override those in `/etc/logadm.conf`.

  Log file names specified in `/etc/logadm.conf` may contain filename substitution characters such as `*` and `?`, that are supported by `csh(1)`.

Two options control when a log file is rotated. They are: `-s` size `-p` period.

When using more than one of these options at a time, there is an implied *and* between them. This means that all conditions must be met before the log is rotated.

If neither of these two options are specified, the default conditions for rotating a log file are: `-s 1b -p 1w`, which means the log file is only rotated if the size is non-zero and if at least 1 week has passed since the last time it was rotated.

By specifying `-p never` as a rotation condition, any other rotation conditions are ignored and `logadm` moves on to the expiration of old log files. By specifying `-p now` as a rotation condition, a log rotation is forced.

Unless specified by the `-o`, `-g`, or `-m` options, `logadm` replaces the log file (after renaming it) by creating an empty file whose owner, group ID, and permissions match the original file.

Three options control when old log files are expired: `-A` age `-C` count `-S` size. These options expire the oldest log files until a particular condition or conditions are met. For example, the combination `-C 5` and the `-S 10m` options expires old log files until there are no more than 5 of the files *and* their combined disk usage is no more than 10 megabytes. If none of these options are specified, the default expiration is `-C 10` which keeps ten old log files. If no files are to be expired, use `-C 0` to prevent expiration by default.

`logadm` stores timestamps in the file `/var/logadm/timestamps`. For users of previous versions of `logadm`, the utility automatically moves timestamps from `/etc/logadm.conf`, their previous repository, to `/var/logadm/timestamps`.

Options The following options are supported:

`-a` *post_command*
Execute the *post_command* after renaming the log file. *post_command* is passed to `sh -c`.

Specify *post_command* as a valid shell command. Use quotes to protect spaces or shell metacharacters in *post_command*.

This option can be used to restart a daemon that is writing to the file. When rotating multiple logs with one `logadm` command, *post_command* is executed only once after all the logs are rotated, not once per rotated log.

`-A` *age*
Delete any versions that have not been modified for the amount of time specified by *age*.

Specify *age* as a number followed by an `h` (hours), `d` (days), `w`(weeks), `m` (months), or `y` (years).

`-b` *pre_command*
Execute *pre_command* before renaming the log file. *pre_command* is passed to `sh -c`.

Specify *pre_command* as a valid shell command. Use quotes to protect spaces or shell metacharacters in the *pre_command*.

This option can be used to stop a daemon that is writing to the file. When rotating multiple logs with one `logadm` command, *pre_command* is executed only once before all the logs are rotated, not once per rotated log.

`-c`
Rotate the log file by copying it and truncating the original logfile to zero length, rather than renaming the file.

`-C` *count*
Delete the oldest versions until there are not more than *count* files left.

If no expire options (`-A`, `-C`, or `-S`) are specified, `-C 10` is the default. To prevent the default expire rule from being added automatically, specify `-C 0`.

`-e` *mail_addr*
Send error messages by email to *mail_addr*.

As `logadm` is typically run from `cron`(1M), error messages are captured by `cron` and mailed to the owner of the `crontab`.

This option is useful if you want the mail regarding error messages to go to another address instead. If no errors are encountered, no mail message is generated.

`-E` *cmd*
Execute *cmd* to expire the file, rather than deleting the old log file to expire it.

cmd is passed it to `sh -c`. The file is considered expired after *cmd* completes. If the old log file is not removed or renamed by the *cmd*, `logadm` considers it for expiration the next time that it runs on the specified log file. If present, the keyword `$file` is expanded in the specified *cmd*to the name of the file being expired.

This option is useful for tasks such as mailing old log files to administrators, or copying old log files to long term storage.

`-f` *conf_file*
Use *conf_file* instead of `/etc/logadm.conf`.

This option allows non-root users to keep their own `logadm` configuration files.

`-F` *timestamp_file*
Use *timestamp_file* instead of `/var/logadm/timestamps` to store `logadm` timestamps.

`-g` *group*
Create a new empty file with the ID specified by *group*, instead of preserving the group ID of the log file.

Specify *group* by name or by numeric group ID, as accepted by `chgrp`(1).

This option requires the ability to change file group ownership using the `chgrp`(1) command.

`-h`
Print a help message that describes `logadm`'s options.

`-l`
Use local time rather than the Coordinated Universal Time (UTC) when naming rotated log files (see the discussion of percent sequences in the templates supplied with the `-t` option).

`-m` *mode*
Create a new empty file with the mode specified by *mode*, instead of preserving the mode of the log file.

Specify *mode* in any form that is accepted by the `chmod(1)` command.

`-M` *cmd*
Use *cmd* to rename the log file. If the keyword `$file` is specified, it is expanded to the name of the log file. Similarly, the keyword `$nfile` is expanded to the new name of the log file. The `$nfile` keyword is only available with commands provided with the `-M` option. After the command completes, the log file is replaced by the rotate file. The default *cmd* is "`/bin/mv $file$nfile`".

`-n`
Print the actions that the `logadm` command will perform without actually performing them.

This option is useful for checking arguments before making any changes to the system.

It is important to remember, however, that since log rotating actions are only printed with this option, `logadm` might not find files that need expiring, but if run without the `-n` `logadm` might create a file that needs expiring by performing the log rotating actions. Therefore, if you see no files being expired with the `-n` option, files still might be expired without it.

`-N`
Prevent an error message if the specified logfile does not exist. Normally, `logadm` produces an error message if the log file is not found. With `-N`, if the log file doesn't exist `logadm` moves on to the expire rules (if any) and then to the next log file (if any), without creating the empty replacement log file.

`-o` *owner*
Create the new empty file with *owner*, instead of preserving the owner of the log file.

Specify *owner* in any form that is accepted by the `chown(1)` command.

`-p` *period*
Rotate a log file after the specified time period (*period*).

Specify *period* as a number followed by `d` for days, `h` for hours, `w` for weeks, `m` for months (30 days) or `y` for years. There are also two special values for period: `now` and `never`. "`-p now`" forces log rotation. "`-p never`" forces no log rotation.

`-P` *timestamp*
Used by `logadm` to record the last time the log was rotated in `/var/logadm/timestamps`.

This option uses *timestamp* to determine if the log rotation period has passed. The format of *timestamp* matches the format generated by `ctime(3C)`, with quotes around it to protect embedded spaces. *timestamp* is always recorded in the Coordinated Universal Time (UTC) timezone.

`-r`
Remove any entries corresponding to the specified *logname* from the `/etc/logadm.conf`.

`-R` *cmd*
Run the *cmd* when an old log file is created by a log rotation. If the keyword `$file` is embedded in the specified command, it is expanded to the name of the old log file just created by log rotation.

This option is useful for processing log file contents after rotating the log. *cmd* is executed by passing it to `sh -c`. When rotating multiple logs with one logadm command, the command supplied with `-R` is executed once every time a log is rotated. This is useful for post-processing a log file (that is, sorting it, removing uninteresting lines, etc.). The `-a` option is a better choice for restarting daemons after log rotation.

`-s` *size*
Rotate the log file only if its size is greater than or equal to *size*.

Specify *size* as a number followed by the letter `b` for bytes, `k` for kilobytes, `m` for megabytes, or `g` for gigabytes.

`-S` *size*
Delete the oldest versions until the total disk space used by the old log files is less than the specified size.

Specify *size* as a number followed by the letter `b` for bytes, `k` for kilobytes, `m` for megabytes, or `g` for gigabytes.

`-t` *template*
Specify the template to use when renaming log files.

template can be a simple name, such as `/var/adm/oldfile`, or it can contain special keywords which are expanded by `logadm` and are in the form $*word*. Allowed sequences are:

$*basename*
The log file name, without the directory name

$*dirname*
The directory of the file to be rotated

$*domain*
Expands to the output of `domainname`

$*file*
The full path name of the file to be rotated

$*isa*
Expands to the output of `uname -p`

$*machine*
Expands to the output of `uname -m`

$*n*
The version number, `0` is most recent, `1` is next most recent, and so forth

$*N*
The same as $*n*, but starts at `1` instead of zero

$*nodename*
Expands to the output of `uname -n`

$*platform*
Expands to the output of `uname -i`

$*release*
Expands to the output of `uname -r`

$*secs*
The number of seconds since `00:00:00 UTC, January 1,1970`

$*zonename*
Expands to the output of `zonename(1)`.

To actually have the dollar sign character in the file name, use `$$`. Any percent sequences allowed by `strftime(3C)` are also allowed, for example, `%d` expands to the day of the month. To actually have a percent sign character in the file name, use `%%`. Both dollar-sign keywords and percent sequences can appear anywhere in the template. If the template results in a pathname with non-existent directories, they are created as necessary when rotating the log file.

If no `-t` option is specified, the default template is `$file.$n`. Actual *rotation* of log files, where each version is shifted up until it expires is done using the `$n` keyword. If the template does not contain the `$n` keyword, the log file is simply renamed to the new name and then the expire rules, if any, are applied.

`-T` *pattern*
Normally `logadm` looks for a list of old log files by turning the template (specified with the `-t` option) into a pattern and finding existing files whose names match that pattern. The `-T` option causes the given pattern to be used instead.

This option is useful if another program fiddles with the old log file names, like a `cron` job to compress them over time. The pattern is in the form of a pathname with special characters such as * and `?` as supported by `csh(1)` filename substitution.

`-v`
Print information about the actions being executed in verbose mode.

`-V`
Validate the configuration file.

This option validates that an entry for the specified *logname* exists in the `/etc/logadm.conf` file and is syntactically correct. If *logname* is not specified, all entries in

the configuration file are validated. If a `logname` argument is specified, the command validates the syntax of that entry. If the entry is found, it is printed and the exit value of the command is true. Otherwise the exit value is false.

`-w` *entryname*
: Write an entry into the config file (that is, `/etc/logadm.conf`) that corresponds to the current command line arguments. If an entry already existed for the specified *entryname*, it is removed first. This is the preferred method for updating `/etc/logadm.conf`, because it prevents syntax errors. The *entryname* is an argument to an invocation of `logadm`. *entryname* might be chosen as something easy to remember or it can be the pathname of the log file. If a pathname, rather than a name is used, it must be a fully qualified pathname.

  If no log file name is provided on a `logadm` command line, the entry name is assumed to be the same as the log file name. For example, the following two lines achieve the same thing, keeping two copies of rotated log files:

```
% logadm -C2 -w mylog /my/really/long/log/file/name
% logadm -C2 -w /my/really/long/log/file/name
```

`-z` *count*
: Compress old log files after all other commands have been executed. *count* of the most recent log files are left uncompressed, therefore making the *count* most recent files easier to peruse. Use *count* of zero to compress all old logs.

  The compression is done with `gzip`(1) and the resulting log file has the suffix of `.gz`.

Operands The following operands are supported:

logname
: Identifies the name of the entry in `/etc/logadm.conf`. If the log file name is specified in the *logname* field, it is assumed that *logname* is the same as the actual log file name.

Examples **EXAMPLE 1** Rotating a File and Keeping Previous Versions

The following example rotates the `/var/adm/exacct/proc` file, keeping ten previous versions in `/var/adm/exacct/proc.0` through `/var/adm/exacct/proc.9`.

Tell `logadm` to copy the file and truncate it.

```
 % logadm -c /var/adm/exacct/proc
```

EXAMPLE 2 Rotating syslog

The following example rotates `syslog` and keeps eight log files. Old log files are put in the directory `/var/oldlogs` instead of `/var/log`:

```
% logadm -C8 -t'/var/oldlogs/syslog.$n' /var/log/syslog
```

EXAMPLE 3 Rotating /var/adm/sulog and Expiring Based on Age

The following entry in the `/etc/logadm.conf` file rotates the `/var/adm/sulog` file and expires any copies older than 30 days.

```
/var/adm/sulog -A 30d
```

EXAMPLE 4 Rotating Files and Expiring Based on Disk Usage

The following entry in the `/etc/logadm.conf` file rotates the `/var/adm/sulog` file and expires old log files when more than 100 megabytes are used by the sum of all the rotated log files.

```
/var/adm/sulog -S 100m
```

EXAMPLE 5 Creating an Entry that Stores the Logfile Name

This example creates an entry storing the log file name and the fact that we want to keep 20 copies in `/etc/logadm.conf`, but the `-p never` means the entry is ignored by the normal `logadm` run from root's crontab every morning.

```
% logadm -w locallog /usr/local/logfile -C20 -p never
```

Use the following entry on the command line to override the `-p never` option:

```
 % logadm -p now locallog
```

EXAMPLE 6 Rotating the apache Error and Access Logs

The following example rotates the apache error and access logs monthly to filenames based on current year and month. It keeps the 24 most recent copies and tells apache to restart after renaming the logs.

This command is run once, and since the `-w` option is specified, an entry is made in `/etc/logadm.conf` so the apache logs are rotated from now on.

```
 % logadm -w apache -p 1m -C 24\
     -t '/var/apache/old-logs/$basename.%Y-%m'\
     -a '/usr/apache/bin/apachectl graceful'\
     '/var/apache/logs/*{access,error}_log'
```

This example also illustrates that the entry name supplied with the `-w` option doesn't have to match the log file name. In this example, the entry name is `apache` and once the line has been run, the entry in `/etc/logadm.conf` can be forced to run by executing the following command:

```
% logadm -p now apache
```

Because the expression matching the apache log file names was enclosed in quotes, the expression is stored in `/etc/logadm.conf`, rather than the list of files that it expands to. This means that each time `logadm` runs from `cron` it expands that expression and checks all the log files in the resulting list to see if they need rotating.

EXAMPLE 6 Rotating the apache Error and Access Logs *(Continued)*

The following command is an example without the quotes around the log name expression. The shell expands the last argument into a list of log files that exist at the time the command is entered, and writes an entry to `/etc/logadm.conf` that rotates the files.

```
logadm -w apache /var/apache/logs/*_log
```

Files

`/etc/logadm.conf`
configuration file for `logadm` command

`/var/logadm/timestamps`
repository for logging timestamps

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os
Interface Stability	Committed

See Also

`chgrp(1)`, `chmod(1)`, `chown(1)`, `csh(1)`, `gzip(1)`, `cron(1M)`, `ctime(3C)`, `strftime(3C)`, `logadm.conf(4)`, `attributes(5)`

Notes

When `logadm` applies expire conditions (supplied by the `-A`, `-C`, and `-S` options), it deletes files, the oldest first, until the conditions are satisfied. If the template used for naming the old logs contained `$n` or `$N`, `logadm` picks the highest value of `$n` or `$N` found in the old log file names first. If the template used is something else, `logadm` uses the modification time to determine which files to expire first. This may not be the expected behavior if an old log file has been modified since it was rotated.

Depending on log file sizes and number of log files, log file rotations can be very time-consuming.

By default, `logadm` works in UTC. Therefore, all entries written to the `/etc/logadm.conf` file (see `logadm.conf(4)`) will have a UTC timestamp. Users can use the `-l` option to set `logadm` to local time.

The `-f` and `-F` options can specify the same file, in which case `logadm` reverts to the same behavior as in prior releases. That is, timestamps are written to the configuration file.

Name logins – list user and system login information

Synopsis

```
/usr/bin/logins [-admoprstux] [-g group...]
     [-l login_name...]
```

Description This command displays information on user, role, and system logins known to the system. Contents of the output is controlled by the command options and can include the following: user, role, or system login; user id number; `passwd` account field value (user name or other information); primary group name; primary group id; multiple group names; multiple group ids; home directory; login shell; and four password-aging parameters. The default information is the following: login id, user id, primary group name, primary group id, and the account field value. Output is sorted by user id, unless the `-t` option is specified.

Options Options may be used together. If so, any login that matches any criteria are displayed.

The following options are supported:

`-a`
Add two password expiration fields to the display. The fields show how many days a password can remain unused before it automatically becomes inactive, and the date that the password expires.

`-d`
Selects logins with duplicate uids.

`-g` *group*
Selects all users belonging to `group`, sorted by login. Multiple groups can be specified as a comma-separated list. When the `-l` and `-g` options are combined, a user is only listed once, even if the user belongs to more than one of the selected groups.

`-l` *login_name...*
Selects the requested login. Multiple logins can be specified as a comma-separated list. Depending on the nameservice lookup types set in `/etc/nsswitch.conf`, the information can come from the `/etc/passwd` and `/etc/shadow` files and other nameservices. When the `-l` and `-g` options are combined, a user is only listed once, even if the user belongs to more than one of the selected groups.

`-m`
Displays multiple group membership information.

`-o`
Formats output into one line of colon-separated fields.

`-p`
Selects logins with no passwords.

`-r`
Select all role logins.

`-s`
Selects all system logins.

`-t`
Sorts output by login instead of by uid.

`-u`
Selects all user logins.

`-x`
Prints an extended set of information about each selected user. The extended information includes home directory, login shell, and password-aging information, each displayed on a separate line. The password information currently consists of password status:

`NP`
Account has no password

`LK`
Account is locked for UNIX authentication

`NL`
Account is a no login account

`UP`
This account has not yet been activated by the administrator and cannot be used.

`PS`
Account probably has a valid password

`UN`
Account password status is unknown. That is, it is not a recognizable hashed password or any of the above entries. See `crypt(3C)` for valid password hashes.

If the login is passworded, status is followed by the date the password was last changed, the number of days required between changes, and the number of days allowed before a change is required. The password-aging information shows the time interval that the user receives a password expiration warning message (when logging on) before the password expires.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also `passwd(1)`, `crypt(3C)`, `attributes(5)`

Name lshal – list HAL devices

Synopsis `/usr/sbin/lshal [`*options*`]`

Description The `lshal` command displays items in the HAL device database.

When invoked without options, `lshal` defaults to long output (`-l`). Long output includes all devices and all properties associated with each device.

The `-s` option lists only UDIs (Unique Device Identifier) for all HAL devices.

The `-t` option prints the HAL device tree, reflecting parent-child relationships with whitespace indentation.

The `-u` option limits output to the specified device.

When invoked with `-m` option, `lshal` monitors HAL device changes, such as property modification and device addition and removal. This option can be combined with the `-u` option to monitor a particular device.

Options The following options are supported:

`-h`, `--help`
Show help information.

`-l`, `--long`
Long output.

`-m`, `--monitor`
Monitor device list.

`-s`, `--short`
Short output (display only non-static part of UDI).

`-t`, `--tree`
Tree view.

`-u`, `--show` *udi*
Show only the specified device.

`-V`, `--version`
Display version number.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/hal
Interface Stability	Volatile

See Also hald(1M), hal-device(1M), attributes(5)

REFERENCE

System Administration Commands - Part 2

Name luxadm – administer Sun Fire 880 storage subsystem and FC_AL devices

Synopsis

```
luxadm [options]... subcommand [options]... enclosure
     [,dev] | pathname...
```

Description The `luxadm` program is an administrative command that manages the SENA, Sun Fire 880 internal storage subsystem, and individual Fiber Channel Arbitrated Loop (FC_AL) devices. `luxadm` performs a variety of control and query tasks depending on the command line arguments and options used.

The command line must contain a subcommand. The command line may also contain options, usually at least one enclosure name or pathname, and other parameters depending on the subcommand. You need specify only as many characters as are required to uniquely identify a subcommand.

Specify the device that a subcommand interacts with by entering a pathname. For the SENA subsystem, a disk device or enclosure services controller may instead be specified by entering the World Wide Name (WWN) for the device or a port to the device. The device may also be specified by entering the name of the SENA enclosure, and an optional identifier for the particular device in the enclosure. The individual FC_AL devices may be specified by entering the WWN for the device or a port to the device.

Pathname Specify the device or controller by either a complete physical pathname or a complete logical pathname.

For SENA, a typical physical pathname for a device is:

```
/devices/sbus@1f,0/SUNW,socal@1,0/sf@0,0/ssd@w2200002037000f96,
         0:a,raw
```

For all SENA IBs (Interface Boards) and Sun Fire 880 SES device controllers on the system, a logical link to the physical paths is kept in the directory `/dev/es`. An example of a logical link is `/dev/es/ses0`.

The WWN may be used in place of the pathname to select an FC_AL device, SENA subsystem IB, or Sun Fire 880 internal storage subsystem. The WWN is a unique 16 hexadecimal digit value that specifies either the port used to access the device or the device itself. A typical WWN value is:

```
2200002037000f96
```

See NOTES for more information on the WWN formats.

For a disk in a Sun Fire 880 internal storage subsystem, a typical physical pathname is:

```
/devices/pci@8,600000/SUNW,qlc@2/fp@0,0/ssd@w2100002037a6303c,0:a
```

and a typical logical pathname is:

```
/dev/rdsk/c2t8d0s2
```

For individual FC_AL devices, a typical physical pathname is:

```
/devices/sbus@3.0/SUNW,socal@d,10000/sf@0,0/ssd@w2200002037049fc3,0:a,raw
```

and a typical logical pathname is:

```
/dev/rdsk/c1t0d0s2
```

Enclosure For SENA, a device may be identified by its enclosure name and slotname:

box_name[`,f`*slot_number*]
box_name[`,r`*slot_number*]

box_name is the name of the SENA enclosure, as specified by the `enclosure_name` subcommand. When used without the optional *slot_number* parameter, the *box_name* identifies the SENA subsystem IB.

`f` or `r` specifies the front or rear slots in the SENA enclosure.

slot_number specifies the slot number of the device in the SENA enclosure, `0-6` or `0-10`.

For a Sun Fire 880 internal storage subsystem, a device may also be identified by its enclosure name and slot name. However, there is only one set of disks:

box_name`[,s`*slot_number*`]`

box_name is the name of the Sun Fire 880 enclosure, as specified by the `enclosure_name` subcommand. When used without the optional *slot_number* parameter, *box_name* identifies the Sun Fire 880 internal storage subsystem enclosure services device. Use *s* to specify the disk slot number in the Sun Fire 880 internal storage subsystem, `0 - 11`.

See `disks(1M)` and `devlinks(1M)` for additional information on logical names for disks and subsystems.

Options The following options are supported by all subcommands:

`-e` Expert mode. This option is not recommended for the novice user.

`-v` Verbose mode.

Options that are specific to particular subcommands are described with the subcommand in the `USAGE` section.

Operands The following operands are supported:

enclosure
The *box_name* of the SENA or Sun Fire 880 internal storage subsystem.

fibre_channel_HBA_port
The path to the host controller port. A typical path is:

```
/devices/pci@8,600000/pci@1/SUNW,qlc@4/fp@0,0:devctl
```

pathname
The logical or physical path of a SENA IB, Sun Fire 880 internal storage subsystem, or disk device. *pathname* can also be the WWN of a SENA IB, SENA disk, or individual FC_AL device.

Usage

Subcommands

`display` *enclosure*[,*dev*] . . . | *pathname* . . .
`display -p` *pathname* . . .
`display -r` *enclosure*[,*dev*] . . . | *pathname* . . .
`display -v` *enclosure*[,*dev*] . . . | *pathname* . . .
Displays enclosure or device specific data.

Subsystem data consists of enclosure environmental sense information and status for all subsystem devices, including disks.

Disk data consists of inquiry, capacity, and configuration information.

`-p` Displays performance information for the device or subsystem specified by *pathname*. This option only applies to subsystems that accumulate performance information.

`-r` Displays error information for the FC_AL device specified by the pathname, or, if the path is a SENA, for all devices on the loop. The `-r` option only applies to SENA subsystems and individual FC_AL devices.

`-v` Displays in verbose mode, including mode sense data.

`download` [`-s`] [`-f` *filename_path*] *enclosure*. . .
Download the prom image pointed to the SENA subsystem Interface Board unit or the Sun Fire 880 internal storage subsystem specified by the enclosure or pathname.

When the SENA's download is complete, the SENA will be reset and the downloaded code executed. If no filename is specified, the default prom image will be used. The default prom image for the SENA is in the directory `usr/lib/locale/C/LC_MESSAGES` and is named `ibfirmware`

When the Sun Fire 880 internal storage subsystem's download is complete, the subsystem resets and the downloaded code begins execution. The default firmware image for the Sun Fire 880 internal storage subsystem is in:
`/usr/platform/SUNW,Sun-Fire-880/lib/images/int_fcbpl_fw`.

`-s` Save. The `-s` option is used to save the downloaded firmware in the FEPROM. If `-s` is not specified, the downloaded firmware will not be saved across power cycles.

The `-s` option does not apply to the Sun Fire 880 internal storage subsystem as it always stores downloaded firmware in the flash memory.

When using the `-s` option, the `download` subcommand modifies the FEPROM on the subsystem and should be used with *caution*.

`enclosure_name` *new_name enclosure* | *pathname*
Change the enclosure name of the enclosure or enclosures specified by the enclosure or pathname. The new name (*new_name*) must be 16 or less characters. Only alphabetic or numeric characters are acceptable. This subcommand applies only to the SENA and the Sun Fire 880 internal storage subsystem.

`failover primary` | `secondary` *pathname*
Select which Sun Storage T3 storage array partner group controller accesses a given logical volume. If `primary` is specified, the logical volume is accessed through the primary controller. If `secondary` is specified, the logical volume is accessed through the secondary controller specified by *pathname*.

`fcal_s_download` [`-f` *fcode-file*]
Download the fcode contained in the file *fcode-file* into *all* the FC100/S Sbus Cards. This command is interactive and expects user confirmation before downloading the fcode.

Use `fcal_s_download` *only* in single-user mode. Using `fcal_s_download` to update a host adapter while there is I/O activity through that adapter *will* cause the adapter to reset. Newly updated FCode will not be executed or visible until a system reboot.

`-f` *fcode-file* When invoked without the `-f` option, the current version of the fcode in each FC100/S Sbus card is printed.

`fcode_download -p`
`fcode_download -d` *dir-name*
Locate the installed FC/S, FC100/S, FC100/P, or FC100/2P host bus adapter cards and download the FCode files in *dir-name* to the appropriate cards. The command determines the correct card for each type of file, and is interactive. User confirmation is required before downloading the FCode to each device.

Use `fcode_download` to load FCode only in single-user mode. Using `fcode_download` to update a host adapter while there is I/O activity through that adapter causes the adapter to reset. Newly updated FCode will not be executed or visible until a system reboot.

`-d` *dir-name* Download the FCode files contained in the directory *dir-name* to the appropriate adapter cards.

`-p` Prints the current version of FCode loaded on each card. No download is performed.

`inquiry` *enclosure*[,*dev*] . . . | *pathname* . . .
Display the inquiry information for the selected device specified by the enclosure or pathname.

`insert_device` [*enclosure*,*dev* . . .]
Assist the user in the hot insertion of a new device or a chain of new devices. Refer to `NOTES` for limitations on hotplug operations. This subcommand applies only to the SENA, Sun Fire 880 internal storage subsystem, and individual FC_AL drives. For the SENA, if more than one enclosure has been specified, concurrent hot insertions on multiple busses can be performed. With no arguments to the subcommand, entire enclosures or individual FC_AL drives can be inserted. For the SENA or the Sun Fire 880 internal storage subsystem, this subcommand guides the user interactively through the hot insertion steps of a new device or chain of devices. If a list of disks was entered it will ask the user to verify the list of devices to be inserted is correct, at which point the user can continue or quit. It then interactively asks the user to insert the disk(s) or enclosure(s) and then creates and displays the logical pathnames for the devices.

`led` *enclosure*,*dev* . . . | *pathname*. . .
Display the current state of the LED associated with the disk specified by the enclosure or pathname. This subcommand only applies to subsystems that support this functionality.

`led_blink` *enclosure*,*dev* . . . | *pathname* . . .
Requests the subsystem to start blinking the LED associated with the disk specified by the enclosure or pathname. This subcommand only applies to subsystems that support this functionality.

`led_off` *enclosure*,*dev* . . . | *pathname* . . .
Requests the subsystem to disable (turn off) the LED associated with the disk specified by the enclosure or pathname. On a SENA subsystem, this may or may not cause the LED to turn off or stop blinking depending on the state of the SENA subsystem. Refer to the SENA Array Installation and Service Manual (p/n 802-7573). This subcommand only applies to subsystems that support this functionality.

`led_on` *pathname* . . .
Requests the subsystem to enable (turn on) the LED associated with the disk specified by the pathname. This subcommand only applies to subsystems that support this functionality.

`power_off` [`-F`] *enclosure*[,*dev*] . . . | *pathname* . . .
When a SENA is addressed, this subcommand causes the SENA subsystem to go into the power-save mode. The SENA drives are not available when in the power-save mode. When a drive in a SENA is addressed the drive is set to the drive off/unmated state. In the drive off/unmated state, the drive is spun down (stopped) and in bypass mode. This command does not apply to the Sun Fire 880 internal storage subsystem.

`-F` The force option only applies to the SENA. Instructs `luxadm` to attempt to power off one or more devices even if those devices are being used by this host (and are, therefore, busy).

Warning: Powering off a device which has data that is currently being used will cause unpredictable results. Users should attempt to power off the device normally (without `-F`) first, only resorting to this option when sure of the consequences of

overriding normal checks.

`power_on` *enclosure*[, *dev*] . .
Causes the SENA subsystem to go out of the power-save mode, when this subcommand is addressed to a SENA.. When this subcommand is addressed to a drive the drive is set to its normal start-up state. This command does not apply to the Sun Fire 880 internal storage subsystem.

`probe` [`-p`]
Finds and displays information about all attached SENA subsystems, Sun Fire 880 internal storage subsystems, and individual FC_AL devices, including the logical pathname, the WWNs, and enclosure names. This subcommand warns the user if it finds different SENAs with the same enclosure names.

`-p` Includes the physical pathname in the display.

`qlgc_s_download` [`-f` *fcode-file*]
Download the FCode contained in the file *fcode-file* into all the FC100/P, FC100/2P PCI host adapter cards. This command is interactive and expects user confirmation before downloading the FCode to each device. Only use `qlgc_s_download` in single-user mode. Using `qlgc_s_download` to update a host adapter while there is I/O activity through that adapter will cause the adapter to reset. Newly updated FCode will not be executed or visible until a system reboot.

`-f` *fcode-file* When invoked without the `-f` option, the current version of the FCode in each FC100/P, FC100/2P PCI card is printed.

`release` *pathname*
Release a reservation held on the specified disk. The pathname should be the physical or logical pathname for the disk.

This subcommand is included for historical and diagnostic purposes only.

`remove_device` [`-F`] *enclosure*[,*dev*] . . . | *pathname* . . .
Assists the user in hot removing a device or a chain of devices. This subcommand can also be used to remove entire enclosures. This subcommand applies to the SENA, Sun Fire 880 internal storage subsystem, and individual FC_AL drives. Refer to `NOTES` for limitations on hotplug operations. For the SENA, Sun Fire 880 internal storage subsystem, and individual FC_AL devices, this subcommand guides the user through the hot removal of a device or devices. During execution it will ask the user to verify the list of devices to be removed is correct, at which point the user can continue or quit. It then prepares the disk(s) or enclosure(s) for removal and interactively asks the user to remove the disk(s) or enclosure(s).

For Multi-Hosted disk, the steps taken are:

- Issue the `luxadm remove_device` command on the first host. When prompted to continue, wait.

- Issue the `luxadm remove_device` command on the secondary hosts. When prompted to continue, wait.
- Continue with the `remove_device` command on the first host. Remove the device when prompted to do so.
- Complete the `luxadm remove_device` command on the additional hosts.

`-F` Instructs `luxadm` to attempt to hot plug one or more devices even if those devices are being used by this host (and are, therefore, *busy* or *reserved*), to *force* the hotplugging operation.

Warning: Removal of a device which has data that is currently being used will cause unpredictable results. Users should attempt to hotplug normally (without `-F`) first, only resorting to this option when sure of the consequences of overriding normal hotplugging checks.

`reserve` *pathname* — Reserve the specified disk for exclusive use by the issuing host. The pathname used should be the physical or logical pathname for the disk.

This subcommand is included for historical and diagnostic purposes only.

`set_boot_dev` [`-y`] *pathname* — Set the boot-device variable in the system PROM to the physical device name specified by *pathname*, which can be a block special device or the pathname of the directory on which the boot file system is mounted. The command normally runs interactively requesting confirmation for setting the default boot-device in the PROM. The `-y` option can be used to run it non-interactively, in which case no confirmation is requested or required.

`start` *pathname* — Spin up the specified disk(s) in a SENA.

`stop` *pathname*... — Spin down the specified disks in a SENA.

SENA, Sun Fire 880 Internal Storage Subsystem, and Individual FC_AL Drive Expert Mode Subcommands

The following subcommands are for expert use only, and are applicable only to the SENA, Sun Fire 880 internal storage subsystem, and fiber channel loops. They should only be used by users that are knowledgeable about the SENA subsystem and fiber channel loops.

If you specify a disk to an expert subcommand that operates on a bus, the subcommand operates on the bus to which the specified disk is attached.

`-e bypass [-ab]` *enclosure*,*dev*
`-e bypass -f` *enclosure* — Request the enclosure services controller to set the LRC (Loop Redundancy Circuit) to the bypassed state for the port and device specified.

This subcommand supports the following options:

`-a` Bypass port `a` of the device specified.

`-b` Bypass port `b` of the device specified.

`-e dump_map` *fibre_channel_HBA_port*

Display WWN data for a target device or host bus adapter on the specified fibre channel port. If there are no target devices on the specified port, an error is returned.

`-e enable [-ab]` *enclosure,dev*
`-e enable -f` *enclosure*

Request the enclosure services controller to set the LRC (Loop Redundancy Circuit) to the enabled state for the port and device specified.

This subcommand supports the following options:

`-a` Enable port `a` of the device specified.

`-b` Enable port `b` of the device specified.

`-e forcelip` *enclosure*`[,`*dev*`] ... |` *pathname* `...`

Force the link to reinitialize, using the Loop Initialization Primitive (LIP) sequence. The enclosure or pathname can specify any device on the loop. Use the pathname to specify a specific path for multiple loop configurations.

This is an expert only command and should be used with caution. It will reset all ports on the loop.

`-e rdls` *enclosure*`[,`*dev*`] ... |` *pathname* `...`

Read and display the link error status information for all available devices on the loop that contains the device specified by the enclosure or pathname.

Other Expert Mode Subcommands

See `NOTES` for limitations of these subcommands. They should only be used by users that are knowledgeable about the systems they are managing.

These commands do not apply to the Sun Fire 880 internal storage subsystem.

`-e bus_getstate` *pathname*	Get and display the state of the specified bus.
`-e bus_quiesce` *pathname*	Quiesce the specified bus.
`-e bus_reset` *pathname*	Reset the specified bus only.
`-e bus_resetall` *pathname*	Reset the specified bus and all devices.
`-e bus_unquiesce` *pathname*	Unquiesce the specified bus. the specified device.
`-e dev_getstate` *pathname*	Get and display the state of the specified device.
`-e dev_reset` *pathname*	Reset the specified device.
`-e offline` *pathname*	Take the specified device offline.
`-e online` *pathname*	Put the specified device online.

Examples

EXAMPLE 1 Displaying the SENAs and Individual FC_AL Devices on a System

The following example finds and displays all of the SENAs and individual FC_AL devices on a system:

```
example% luxadm probe
```

EXAMPLE 2 Displaying a SENA or Sun Fire 880 Internal Storage Subsystem

The following example displays a SENA or Sun Fire 880 internal storage subsystem:

```
example% luxadm display /dev/es/ses0
```

EXAMPLE 3 Displaying Two Subsystems

The following example displays two subsystems using the enclosure names:

```
example% luxadm display BOB system1
```

EXAMPLE 4 Displaying Information about the First Disk

The following example displays information about the first disk in the front of the enclosure named `BOB`. Use `f` to specify the front disks. Use `r` to specify the rear disks.

```
example% luxadm display BOB,f0
```

EXAMPLE 5 Displaying Information on a Sun Fire 880 Internal Storage Subsystem

The Sun Fire 880 internal storage subsystem has only one set of disks. In this case, use *s* to specify the slot:

```
example% luxadm display BOB,s0
```

EXAMPLE 6 Displaying Information about a SENA disk, an Enclosure, or an Individual FC_AL Drive

The following example displays information about a SENA disk, an enclosure, or an individual FC_AL drive with the port WWN of 2200002037001246:

```
example% luxadm display 2200002037001246
```

EXAMPLE 7 Using Unique Characters to Issue a Subcommand

The following example uses only as many characters as are required to uniquely identify a subcommand:

```
example% luxadm disp BOB
```

EXAMPLE 8 Displaying Error Information

The following example displays error information about the loop that the enclosure BOB is on:

```
example% luxadm display -r BOB
```

EXAMPLE 9 Downloading New Firmware into the Interface Board

The following example downloads new firmware into the Interface Board in the enclosure named BOB (using the default path for the file to download):

```
example% luxadm download -s BOB
```

EXAMPLE 10 Displaying Information from the SCSI Inquiry Command

The following example displays information from the SCSI inquiry command from all individual disks on the system, using only as many characters as necessary to uniquely identify the inquiry subcommand:

```
example% luxadm inq /dev/rdsk/c?t?d?s2
```

EXAMPLE 11 Hotplugging

The following example hotplugs a new drive into the first slot in the front of the enclosure named BOB:

```
example% luxadm insert_device BOB,f0
```

The following example hotplugs a new drive into the first slot in the Sun Fire 880 internal storage subsystem named SF880-1:

```
example% luxadm insert_device SF880-1,s0
```

EXAMPLE 12 Running an Expert Subcommand

The following example runs an expert subcommand. The subcommand forces a loop initialization on the loop that the enclosure BOB is on:

```
example% luxadm -e forcelip BOB
```

EXAMPLE 13 Using the Expert Mode Hot Plugging Subcommands

An example of using the expert mode hot plugging subcommands to hot remove a disk follows. See NOTES for hot plugging limitations.

The first step reserves the SCSI device so that it can't be accessed by way of its second SCSI bus:

```
example# luxadm reserve /dev/rdsk/c1t8d0s2
```

EXAMPLE 14 Taking the Disk to be Removed Offline

The next two steps take the disk to be removed offline then quiesce the bus:

```
example# luxadm -e offline /dev/rdsk/c1t8d0s2
example# luxadm -e bus_quiesce /dev/rdsk/c1t8d0s2
```

EXAMPLE 15 Unquiescing the Bus

The user then removes the disk and continues by unquiescing the bus, putting the disk back online, then unreserving it:

```
example# luxadm -e bus_unquiesce /dev/rdsk/c1t8d0s2
example# luxadm -e online /dev/rdsk/c1t8d0s2
example# luxadm release /dev/rdsk/c1t8d0s2
```

Environment Variables

See environ(5) for a description of the LANG environment variable that affects the execution of luxadm.

Exit Status

The following exit values are returned:

0 Successful completion.

−1 An error occurred.

Files

- usr/lib/firmware/fc_s/fc_s_fcode
- usr/lib/locale/C/LC_MESSAGES/ibfirmware

Attributes

See attributes(5) for descriptions of the following attributes:

usr/sbin

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/storage/fc-utilities

See Also

devlinks(1M), disks(1M), attributes(5), environ(5), ses(7D)

Oracle Solaris 11.1 Administration: SAN Configuration and Multipathing

Oracle Solaris Cluster 3.3 with Fibre Channel JBOD Storage Device Manual

Notes Refer to *Tutorial for SCSI use of IEEE Company_ID*, R. Snively, for additional information regarding the IEEE extended WWN. Currently, only some device drivers support hot plugging. If hot plugging is attempted on a disk or bus where it is not supported, an error message of the form:

```
luxadm: can't acquire "PATHNAME": No such file or directory
```

...will be displayed.

You must be careful not to quiesce a bus that contains the root or the `/usr` filesystems or any swap data. If you do quiesce such a bus a deadlock can result, requiring a system reboot.

Name mail.local – store mail in a mailbox

Synopsis `/usr/lib/mail.local [-f` *sender*`] [-d]` *recipient*

Description `mail.local` reads the standard input up to an end-of-file and appends it to each user's mail file (mailbox). This program is intended to be used by `sendmail(1M)` as a mail delivery agent for local mail. It is not a user interface agent.

Messages are appended to the user's mail file in the `/var/mail` directory. The user must be a valid user name.

Each delivered mail message in the mailbox is preceded by a "Unix From line" with the following format:

`From` *sender_address time_stamp*

The *sender_address* is extracted from the SMTP envelope address (the envelope address is specified with the `-f` option).

A trailing blank line is also added to the end of each message.

The mail files are locked with a `.lock` file while mail is appended.

The mail files are created with mode `660`, owner is set to *recipient*, and group is set to `mail`. If the "biff" service is returned by `getservbyname(3SOCKET)`, the biff server is notified of delivered mail. This program also computes the `Content-Length:` header which will be used by the mailbox reader to mark the message boundary.

Options The following options are supported:

`-f` *sender*	Specifies the "envelope from address" of the message. This flag is technically optional, but should be used.
`-d`	Specifies the recipient of the message. This flag is also optional and is supported here for backward compatibility. That is, `mail.local` *recipient* is the same as `mail.local -d` *recipient*.
`-l`	Turn on LMTP mode.
`-r` *from*	Specify the sender's name (for backward compatibility).
`-7`	Do not advertise 8BITMIME support in LMTP mode.
`-b`	Return a permanent error instead of a temporary error if a mailbox exceeds quota.

Operands The following operand is supported:

recipient	The recipient of the mail message.

Environment Variables

`TZ` Used to set the appropriate time zone on the timestamp.

Exit Status

The following exit values are returned:

`0` Successful operation.

`>0` An error occurred.

Files

`/tmp/local.`*XXXXXX* temporary files

`/tmp/lochd.`*XXXXXX* temporary files

`/var/mail/`*user_name* user's mail file

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	service/network/smtp/sendmail

See Also

`mail(1)`, `comsat(1M)`, `sendmail(1M)`, `getservbyname(3SOCKET)`, `attributes(5)`

Name makedbm – make a dbm file, or get a text file from a dbm file

Synopsis

```
makedbm [-b] [-l] [-s] [-E] [-i yp_input_file]
     [-o yp_output_name] [-d yp_domain_name]
     [-m yp_master_name] [-S delimiter]
     [-D number_of_delimiters] infile outfile

makedbm [-u dbmfilename]
```

Description The `makedbm` utility takes the *infile* and converts it to a pair of files in `ndbm` format (see `ndbm(3C)`), namely *outfile*`.pag` and *outfile*`.dir`. Each line of the input file is converted to a single `dbm` record. All characters up to the first TAB or SPACE form the key, and the rest of the line is the data. If a line ends with '\' (backslash), the data for that record is continued on to the next line. `makedbm` does not treat '#' (pound-sign) as a special character.

Because `makedbm` is mainly used in generating `dbm` files for the NIS name service, it generates a special entry with the key *yp_last_modified*, which is the date of *infile* (or the current time, if *infile* is '–'). The entries that have keys with the prefix *yp_* are interpreted by NIS server utilities.

Options The following options are supported:

`-b`	Insert the `YP_INTERDOMAIN` into the output. This key causes `ypserv(1M)` to use DNS for host name and address lookups for hosts not found in the maps.
`-d` *yp_domain_name*	Create a special entry with the key *yp_domain_name*.
`-D` *number_of delimiters*	Specify *number_of_delimiters* to skip before forming the key.
`-E`	Delimiters are escaped.
`-i` *yp_input_file*	Create a special entry with the key *yp_input_file*.
`-l`	Lower case. Convert the keys of the given map to lower case, so that, for example, host name matches succeed independent of upper or lower case distinctions.
`-m` *yp_master_name*	Create a special entry with the key *yp_master_name*. If no master host name is specified, *yp_master_name* is set to the local host name.
`-o` *yp_output_name*	Create a special entry with the key *yp_output_name*.
`-s`	Secure map. Accept connections from secure NIS networks only.
`-S` *delimiter*	Specify the *delimiter* to use instead of the default delimiter for forming the key.
`-u` *dbmfilename*	Undo a `dbm` file. Prints out the file in text format, one entry per line, with a single space separating keys from values.

Operands The following operands are supported:

infile Input file for `makedbm`. If *infile* is '–' (dash), the standard input is read.

outfile One of two output files in `ndbm` format: *outfile*`.pag` and *outfile*`.dir`.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also `ypserv(1M)`, `ndbm(3C)`, `attributes(5)`

Name makemap – create database maps for sendmail

Synopsis

```
makemap [-N] [-d] [-f] [-o] [-r] [-s] [-v] [-C file]
     [-c cachesize] [-D commentchar] [-e] [-l] [-t delim]
     [-u] mantype mapname
```

Description `makemap` creates the database maps used by the keyed map lookups in `sendmail(1M)`. `makemap` reads from the standard input and outputs to the specified *mapname*.

In all cases, `makemap` reads lines from the standard input consisting of two words separated by whitespace. The first is the database key, the second is the value. The value may contain `%`*n* strings to indicated parameter substitution. Literal percents should be doubled (`%%`). Blank lines and lines beginning with `#` are ignored.

`makemap` handles three different database formats. Database format is selected using the *maptype* parameter. See `OPERANDS`.

Options The following options are supported:

`-c` *cachesize*	Use the specified hash and B-Tree cache size (*cachesize*).
`-C` *file*	Use the specified sendmail configuration file (*file*) for looking up the TrustedUser option.
`-d`	Allow duplicate keys in the map. This is only allowed on B-Tree format maps. If two identical keys are read, both be inserted into the map.
`-D` *commentchar*	Use the specified character to indicate a comment (which is ignored) instead of the default of '`#`'.
`-e`	Allow empty value (right hand side).
`-f`	Normally, all upper case letters in the key are folded to lower case. This flag disables that behavior. This is intended to mesh with the `-f` flag in the `K` line in `sendmail.cf`. The value is never case folded.
`-l`	List supported map types.
`-N`	Include the null byte that terminates strings in the map. This must match the `-N` flag in the `K` line in `sendmail.cf`
`-o`	Append to an old file. This allows you to augment an existing file.
`-r`	Allow replacement of existing keys. Normally `makemap` complains if you repeat a key, and does not do the insert.
`-s`	Ignore safety checks on maps being created. This includes checking for hard or symbolic links in world writable directories.
`-t` *delim*	Use the specified delimiter (*delim*) instead of whitespace.

`-u` Dump (unmap) the content of the database to standard output. Note that, if the `-t` option is also provided, the specified delimiter is used when the content is dumped instead of whitespace.

`-v` Verbosely print what it is doing.

Operands The following operands are supported:

mapname File name of the database map being created.

maptype Specifies the database format. The following *maptype* parameters are available:

`dbm` Specifies DBM format maps.

`btree` Specifies B-Tree format maps.

`hash` Specifies hash format maps.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	service/network/smtp/sendmail

See Also `editmap(1M)`, `sendmail(1M)`, `attributes(5)`

Name masfcnv – SNMP configuration migration script

Synopsis

```
/usr/lib/net-snmp/masfcnv [-cimnrs] [-l agentmaster]
     [-p enabledisableerror] [-t noneadd]
     [-u agentmastererror] [-y agentmastererror]

masfcnv [-V]

masfcnv [-?]
```

Description The `masfcnv` script is used to assist the system administrator in migrating an existing set of configuration files for the Sun SNMP Management Agent for Sun Fire and Netra Systems (MASF) to the Systems Management Agent (SMA).

The script accepts as input the currently installed set of MASF and SMA configuration files and outputs a new set of SMA configuration files. Existing SMA configuration files are backed up by appending `.bak` to the filename. The administrator can choose to output the new configuration to standard output, instead of replacing the current configuration, by specifying the `-n` option.

The migration script must be run as the superuser. Failure to do so causes the script to exit with an error message. Before running the script you should ensure that both the SMA and MASF agents are not running. If the agents are running they will be shut down by the script.

The migration script installs a new startup script for the MASF agent in `/etc/init.d`, as well as a backup of the old script. During migration, MASF will be configured as an AgentX subagent of SMA. All migration settings will be migrated to the SMA configuration file.

The migration script aborts if any unrecognized directives are found in either the MASF configuration files or the SMA configuration files. This can be overridden with the `-i` option. If this option is selected, the behavior is to retain unrecognized directives that were present in the SMA configuration, but remove those present in the MASF configuration.

The migration script then proceeds to migrate access control and trap configuration. As a side effect of running the migration script, the following directives might be expanded by the script into multiple directives with an equivalent interpretation:

- `rwcommunity`
- `rocommunity`
- `rwuser`
- `rouser`
- `trapcommunity`
- `trapsink`
- `trap2sink`
- `informsink`

Access Control Migration

Access control directives are expanded into the equivalent com2sec, group, access and view directives. Existing group names are renamed by prepending a prefix to avoid conflict with any which may already be defined in SMA.

When migrating SNMPv1 or v2c access control, a conflict can occur if both MASF and SMA configuration files have defined access permissions for the same community and source address. The default behavior is to abort with a message, unless a use of the `-y` option specifies otherwise. If `-y agent` is specified then the MASF configuration takes precedence. If `-y master` is specified then the SMA configuration is retained.

When migrating USM configuration (SNMPv3), a conflict can occur if both SMA and MASF configurations define a user with the same `securityName`. If this occurs, the behavior of the script is determined by the `-u` option. If `-u agent` is specified, the configuration of the user defined in the MASF configuration files is the one that is retained. Otherwise, if the `-u master` option is specified, the use defined in the SMA configuration files is retained.

By default, the migration script attempts to migrate USM users from MASF to SMA. The script determines whether there are any SNMPv3 users present in the SMA configuration and whether the default `engineID` has been overridden in the SMA configuration files. If neither of these conditions obtain, then the any `usmUser` statements containing localized authentication keys can be migrated to SMA, along with the MASF `engineID`. This results in the `engineID` of the SMA master agent changing.

If the script determines that there are existing SNMPv3 users or a manually configured `engineID` present in the SMA configuration, only those users defined in `createUser` statements are transferred. Those users that were defined in `usmUser` statements are transferred but will have their passwords reset to a random value. You should notify your users of their new password or reset the password yourself by editing the newly-generated configuration file.

Trap/Inform Migration

The migration script performs a check to determine whether a trap destination defined for MASF is already specified in an existing SMA `trapsink`, `trap2sink` or `informsink` directive. If this is the case, then the directive in the MASF configuration will be discarded to avoid duplicate traps/informs being received.

`trapsink`, `trap2sink` and `informsink` directives specified in the existing SMA configuration are considered valid destinations for MASF traps/informs and will receive them from the MASF subagent after migration.

If the `-t none` option was specified on the command line, the migration script carries over any remaining MASF trap/inform directives without modification.

If the `-t add` option was specified (the default), the migration script expands any `trapsink`, `trap2sink`, or `informsink` directives to use the `TARGET-MIB` and `NOTIFICATION-MIB`. The `TARGET-MIB` specifies targets using IP addresses, so it might be desirable to use the `-t none` option if, for example, the network allocates IP addresses to hostnames dynamically by means of DHCP.

The expanded directives defines filters specific to the MASF agent so that traps from other subagents will not be received by migrated trap destinations. Existing filters present in the SMA configuration are, by default, not modified and might or might not receive MASF traps, depending upon the filters that were originally defined for them.

If the `-l` option is specified, any filters already defined in the `TARGET-MIB` and the `NOTIFICATION-MIB` for SMA are extended to include traps from MASF. In the event that a trap destination is already configured in the `TARGET-MIB` with the same target address and community as an existing MASF trap/inform sink, a conflict will arise.

If `-l agent` was specified and a conflict arises, the migration script uses the target SNMP parameters (that is, the SNMP version and choice of trap/inform) defined by the MASF `trap/informsink` directive to send traps to this destination. Otherwise, if the `-l master` option was specified, the conflict will be resolved using the target SNMP parameters specified in the SMA configuration.

Miscellaneous

If the migration script encounters in the MASF configuration file any of the directives listed below and the directives are either not present or differ from the SMA configuration, the script will log a warning message.

- `syslocation`
- `syscontact`
- `sysname`
- `sysservices`
- `agentgroup`
- `agentuser`
- `authtrapenable`

Options

The following options are supported:

`-?`
`--help`
: Displays usage information.

`-c`
`--no-community`
: Do not transfer v1/v2c communities.

`-i`
`--ignore-unrecognized-directives`
: Continue processing if unrecognized directives are present.

`-l agent | master`
`--master-trap-target=agent | master`
: If `agent` is specified, the existing SMA trap targets will be configured to receive traps that were previously sent to destinations for the Sun Fire SNMP agent. If `master` is specified, the targets will be configured to receive Sun Fire SNMP traps, but existing SNMP target parameters will be used.

`-m`
`--no-usmuser`
: Do not transfer usm (v3) users.

`-n`
`--dry-run`
: Run the migration without modifying any files. If an error arises, continue processing. This can be used to determine the likely migration issues.

`-p enable | disable | error`
`--use-agent-port=enable | disable | error`
: Indicates whether the port originally used by the Sun Fire SNMP agent should be used by the SMA agent after migration (if the two agents are using different ports). If `enable` is specified, then the port used by the Sun Fire SNMP agent will also be used by the SMA agent after migration. If `disable` is specified, the ports used by SMA will not be updated by the migration tool. If the `error` option is specified and the SMA agent is not already using the same ports as those used by the original Sun Fire SNMP agent, an error is reported and the migration process is terminated. If no option is specified the default behavior is equivalent to the `error` flag.

`-r`
`--no-trap`
: Do not transfer trap destinations.

`-s`
`--skip-user`
: If a user is found in the MASF configuration file that cannot be created in the new configuration because of a change in the engine ID, then output a message indicating that the user could not be migrated (needs to be manually recreated) and continue processing. If this option is not present, the migration tool will consider such a situation as an error and abort.

`-t none | add`
`--trap-filter=none | add`
: If `none` is specified then the script will copy trap directives directly. The administrator might need to manually update the

configuration file to ensure traps are only delivered to their intended destinations. If `add` is specifed, trap filters will be constructed so that traps originating from the original Sun Fire SNMP agent are delivered only to the destinations that originally received them. The default behavior is `add`.

`-u` *agent*|*master*|*error*
`--select-user=`*agent*|*master*|*error*

Specifies that if a user with the same name is found in both configuration files that the conflict is to be resolved using the specified configuration file as input. Selecting a user from a particular file will also cause the group declaration for that user to be taken from the same file. If `agent` is specified then the user will be taken from the configuration file for the Sun Fire SNMP Agent. If `master` is specified, the user will be taken from the SMA configuration. Otherwise, if `error` is given, the script will terminate. If this option is not present, the default behavior is equivalent to the `error` flag.

`-V`
`--version`

Display the version of this script.

`-y` *agent*|*master*|*error*
`--select-community=`*agent*|*master*|*error*

Specifies that if a community with the same name is found in both configuration files that the conflict is to be resolved using the specified configuration file as input. If `agent` is specified then the community will be taken from the configuration file for the Sun Fire SNMP Agent. If `master` is specified, the community will be taken from the SMA configuration. Otherwise, if `error` is given, the script will terminate. If this option is not present, the default behavior is equivalent to the `error` flag.

Examples

EXAMPLE 1 Simplest Case

The command shown below is appropriate for a simple migration. The migration fails if there are any potential conflicts.

```
# masfcnv
```

EXAMPLE 2 Migrating Such That MASF Settings Override

To migrate the MASF configuration such that it will always succeed, that MASF settings will override in the event of a conflict with SMA, and that access will still be provided on the original MASF port, enter:

```
# masfcnv -is -l agent -p enable -u agent -y agent
```

EXAMPLE 3 Dry Run, Retaining SMA Settings

To attempt a dry run and migrate the configuration such that any conflicts will be resolved by retaining existing SMA settings, enter:

```
masfcnv -l master -u master -y master
```

Exit Status

`0` Success.

`non-zero` A problem occurred during migration.

Files

`/etc/sma/snmp/snmpd.conf` `/var/sma_snmp/snmpd.conf`	SMA configuration files
`/etc/opt/SUNWmasf/conf/snmpd.conf` `/var/opt/SUNWmasf/snmpd.dat`	MASF configuration files
`/tmp/sma_migration.log`	`masfcnv` log file

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsmcmd
Interface Stability	Committed

See Also

`attributes(5)`

Notes

The former path to this utility, `/usr/sfw/lib`, is now a link to `/usr/lib`.

Name mdlogd – Solaris Volume Manager daemon

Synopsis

```
mdlogd
```

Description `mdlogd` implements a simple daemon that watches the system console looking for messages written by the Solaris Volume Manger. When a Solaris Volume Manager message is detected, `mdlogd` sends a generic SNMP trap.

To enable traps, you must configure `mdlogd` into the SNMP framework. See *Solaris Volume Manager Administration Guide*.

Usage `mdlogd` implements the following SNMP MIB:

```
SOLARIS-VOLUME-MGR-MIB DEFINITIONS ::= BEGIN
        IMPORTS
                enterprises FROM RFC1155-SMI
                DisplayString FROM SNMPv2-TC;

        -- Sun Private MIB for Solaris Volume Manager


        sun       OBJECT IDENTIFIER ::= { enterprises 42 }
        sunSVM       OBJECT IDENTIFIER ::= { sun 104 }

        -- this is actually just the string from /dev/log that
        -- matches the md: regular expressions.
        -- This is an interim SNMP trap generator to provide
        -- information until a more complete version is available.

        -- this definition is a formalization of the old
        -- Solaris DiskSuite mdlogd trap mib.

        svmOldTrapString OBJECT-TYPE
                        SYNTAX DisplayString (SIZE (0..255))
                        ACCESS read-only
                        STATUS mandatory
                        DESCRIPTION
                        "This is the matched string that
                         was obtained from /dev/log."
        ::= { sunSVM 1 }

        -- SVM Compatibility ( error trap )

        svmNotice        TrapTRAP-TYPE
                        ENTERPRISE sunSVM
                        VARIABLES { svmOldTrapString }
                        DESCRIPTION
                                "SVM error log trap for NOTICE.
                                 This matches 'NOTICE: md:'"
```

```
        ::= 1

        svmWarningTrap  TRAP-TYPE
                        ENTERPRISE sunSVM
                        VARIABLES { svmOldTrapString }
                        DESCRIPTION
                                "SVM error log trap for WARNING..
                                 This matches 'WARNING: md:'"
        ::= 2

        svmPanicTrap    TRAP-TYPE
                        ENTERPRISE sunSVM
                        VARIABLES { svmOldTrapString }
                        DESCRIPTION
                                "SVM error log traps for PANIC..
                                This matches 'PANIC: md:'"
        ::= 3
END
```

Attributes

See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWlvma, SUNWlvmr
Interface Stability	Obsolete

See Also

snmpdx(1M), attributes(5)

Solaris Volume Manager Administration Guide

Name mdmonitord – daemon to monitor metadevices

Synopsis

```
/usr/sbin/mdmonitord [-t time_interval]
```

Description The `mdmonitord` utility is part of Solaris Volume Manager. It monitors and checks RAID1 (mirrors), RAID5 and hot spares.

There are two methods for checking:

- At fixed time intervals.
- When a RAID-1 (mirror), RAID-5, or hot spare fails. A failure generates an error event which triggers a check of these metadevices.

Options The following options are supported:

`-t` Time interval in seconds. The default value is 0, which causes probes to occur only upon an error. If you want to run `mdmonitord` at a regular interval, a value of 1800 (seconds, every half hour) is recommended as a starting point.

Exit Status The following exit values are returned:

`0` Successful completion.

>0 An error occurred.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	storage/svm

See Also `svcs(1)`, `metaclear(1M)`, `metadb(1M)`, `metadetach(1M)`, `metahs(1M)`, `metainit(1M)`, `metaoffline(1M)`, `metaonline(1M)`, `metaparam(1M)`, `metarecover(1M)`, `metarename(1M)`, `metareplace(1M)`, `metaset(1M)`, `metassist(1M)`, `metastat(1M)`, `metasync(1M)`, `metattach(1M)`, `svcadm(1M)`, `md.tab(4)`, `md.cf(4)`, `mddb.cf(4)`, `md.tab(4)`, `attributes(5)`, `smf(5)`, `md(7D)`

Solaris Volume Manager Administration Guide

Notes Since frequent probes can affect performance, it is recommended that the intervals between probes be limited.

The `mdmonitord` service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/system/mdmonitor
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. The service's status can be queried using the `svcs(1)` command.

Name mdnsd – Multicast DNS daemon

Synopsis

```
mdnsd [-debug]
```

Description `mdnsd` is the daemon program for Multicast DNS and DNS Service Discovery.

The `mdnsd` daemon listens on UDP port 5353 for Multicast DNS Query packets. When it receives a query for which it knows an answer, `mdnsd` issues the appropriate Multicast DNS Reply packet.

The `mdnsd` daemon also performs Multicast DNS Queries on behalf of client processes, and maintains a cache of the replies.

The `mdnsd` daemon has no user-specifiable command-line argument, and users should not run `mdnsd` manually. The `mdnsd` daemon should be managed by the Solaris Management Facility (SMF) and should be administered by the `svcadm(1M)` command using the following fault management resource identifier (FMRI):

```
svc:/network/dns/multicast:default
```

To examine `mdnsd`'s internal state for debugging and diagnostic purposes, send it a SIGUSR1 signal, and it will then log a snapshot summary of its internal state using the `syslog(3C)` facility. `mdnsd` uses the syslog facility code daemon and info priority level.

Options The `mdnsd` daemon recognizes the following option:

`-debug` Debug mode. The `mdnsd` daemon sends output to the standard error, and does not run in the background. This option is only intended for debugging the daemon.

CONFIGURATION Multicast DNS can be used to look up host names and host addresses by specifying `mdns` as a source for hosts and ipnodes in the name service switch configuration file `nsswitch.conf(4)`. The configuration options for host name and host address queries using Multicast DNS are stored in the SMF repository. This configuration can be modified by the `svccfg(1M)` command using the following fault management resouce identfier (FMRI):

```
svc:/network/dns/multicast:default
```

The configuration options for host name and host address queries using Multicast DNS are stored in a property group named "`nss_mdns_config`". The properties that make up the configuration options are as follows:

search A list of search domains for host name look up. By default, no search domains are included in the configuration. The search list is currently limited to six domains.

valid A list of valid domains checked before Multicast DNS is used to look up the host name or host address for a domain. Domains specified in the search list are always included in the valid list. The valid domain list is currently limited to ten domains.

Please note the above configuration options are Volatile and may change in a future release.

Files /usr/lib/inet/mdnsd

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	service/network/dns/mdns
Interface Stability	Volatile

See Also `dns-sd(1M)`, `svcadm(1M)`, `svccfg(1M)`, `syslog(3C)`,`nsswitch.conf(4)`, `attributes(5)`

For information on Multicast DNS, see http://www.multicastdns.org/

For information on DNS Service Discovery, see `http://www.dns-sd.org/`.

Name medstat – check the status of mediator hosts for a given diskset

Synopsis

```
/usr/sbin/medstat [-q] -s setname
```

Description If a specified diskset has been configured for mediators, `medstat` attempts to contact these hosts to see if they are accessible and returns the results of the communication.

Options

`-q` This optional argument disables the printing of informative text. When used with `-q`, `medstat` still prints error messages and returns a result code.

`-s` *setname* Specifies the name of a diskset on which `medstat` will work.

Examples **EXAMPLE 1** Checking diskset

This example checks the mediator hosts for the selected diskset.

```
# medstat -s relo-red
```

The name of the diskset is `relo-red`. The `medstat` command prints the status for each mediator host. Additionally, if the mediator quorum is met, either through a "golden" mediator host or because half+1 of the mediator hosts respond, the exit code is 0. If the quorum is not met, then the exit code is 1. If no mediator hosts have been configured for the named diskset, the exit code is 2. The status field will contain one of the following values: Unreachable, Bad, Fatal, or Ok, where Unreachable indicates an RPC/communication problem, Bad indicates an error in the mediator data, Fatal indicates any other error condition, and Ok indicates no error conditions.

Files `/etc/lvm/meddb` Contains the mediator data for a host that has been selected as a mediator host.

Exit Status The following exit values are returned:

`0` Successful completion.

>0 An error occurred.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	storage/svm
Interface Stability	Committed

See Also `metaclear(1M)`, `metadb(1M)`, `metadetach(1M)`, `metahs(1M)`, `metainit(1M)`, `metaoffline(1M)`, `metaonline(1M)`, `metaparam(1M)`, `metareplace(1M)`, `metaset(1M)`, `metastat(1M)`, `metasync(1M)`, `metattach(1M)`, `md.cf(4)`, `md.tab(4)`, `mddb.cf(4)`, `meddb(4)`, `mediator(7D)`

Sun Cluster documentation, *Solaris Volume Manager Administration Guide*

Notes This command is designed for use in the high availability product.

Name metaclear – delete active metadevices and hot spare pools

Synopsis

```
/usr/sbin/metaclear -h

/usr/sbin/metaclear [-s setname] -a [-f]

/usr/sbin/metaclear component

/usr/sbin/metaclear [-s setname] [-f] metadevice... hot_spare_pool...

/usr/sbin/metaclear [-s setname] -r [-f] metadevice... hot_spare_pool...

/usr/sbin/metaclear [-s setname] -p component

/usr/sbin/metaclear [-s setname] -p metadevice
```

Description The `metaclear` command deletes the specified metadevice or *hot_spare_pool*, or purges all soft partitions from the designated component. Once a metadevice or hot spare pool is deleted, it must be re-created using `metainit` before it can be used again.

Any metadevice currently in use (open) cannot be deleted.

Options Root privileges are required for all of the following options except `-h`.

`-a` Deletes all metadevices and configured hot spare pools in the set named by `-s`, or the local set by default.

`-f` Deletes (forcibly) a metadevice that contains a subcomponent in an error state.

`-h` Displays usage message.

`-p` Deletes (purges) all soft partitions from the specified metadevice or component.

`-r` Recursively deletes specified metadevices and hot spare pools, but does not delete metadevices on which others depend.

`-s` *setname* Specifies the name of the diskset on which `metaclear` will work. Using the `-s` option causes the command to perform its administrative function within the specified diskset. Without this option, the command performs its function on local metadevices and/or hot spare pools.

Operands

metadevice ... Specifies the name(s) of the metadevice(s) to be deleted.

component Specifies the `c*d*t*s*` name(s) of the components containing soft partitions to be deleted.

hot_spare_pool ... Specifies the name(s) of the hot spare pools to be deleted. Names for hot spare pools can be any legal file name that is composed of alphanumeric characters, a dash ("-"), an underscore ("_"), or a period (".") . Names must begin with a letter. The words "all" and "none" are reserved and cannot be used.

Examples

EXAMPLE 1 Deleting Various Devices

The following example deletes a metadevice named `d10`.

```
# metaclear /dev/md/dsk/d10
```

The following example deletes all local metadevices and hot spare pools on the system.

```
# metaclear -a
```

The following example deletes a mirror, `mymirror`, with a submirror in an error state.

```
# metaclear -f mymirror
```

The following example deletes a hot spare pool, `hsp001`.

```
# metaclear hsp001
```

The following example deletes a soft partition, `d23`.

```
# metaclear d23
```

The following example purges all soft partitions on the slice `c2t3d5s2` if those partitions are not being used by other metadevices or are not open.

```
# metaclear -p c2t3d5s2
```

The following example purges soft partitions from a metadevice.

```
# metaclear -p d2
d3: Soft Partition is cleared
d4: Soft Partition is cleared
d5: Soft Partition is cleared
```

Exit Status

The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	storage/svm
Interface Stability	Committed

See Also

`mdmonitord(1M)`, `metadb(1M)`, `metadetach(1M)`, `metahs(1M)`, `metainit(1M)`, `metaoffline(1M)`, `metaonline(1M)`, `metaparam(1M)`, `metarecover(1M)`, `metarename(1M)`, `metareplace(1M)`, `metaset(1M)`, `metassist(1M)`, `metastat(1M)`, `metasync(1M)`, `metattach(1M)`, `md.tab(4)`, `md.cf(4)`, `mddb.cf(4)`, `md.tab(4)`, `attributes(5)`, `md(7D)`

Solaris Volume Manager Administration Guide

Name metadb – create and delete replicas of the metadevice state database

Synopsis

```
/usr/sbin/metadb -h

/usr/sbin/metadb [-s setname]

/usr/sbin/metadb [-s setname] -a [-f] [-k system-file] mddbnn

/usr/sbin/metadb [-s setname] -a [-f] [-k system-file]
     [-c number] [-l length] slice...

/usr/sbin/metadb [-s setname] -d [-f] [-k system-file] mddbnn

/usr/sbin/metadb [-s setname] -d [-f] [-k system-file] slice...

/usr/sbin/metadb [-s setname] -i

/usr/sbin/metadb [-s setname] -p [-k system-file]
     [mddb.cf-file]
```

Description The `metadb` command creates and deletes replicas of the metadevice state database. State database replicas can be created on dedicated slices, or on slices that will later become part of a simple metadevice (concatenation or stripe) or RAID5 metadevice. Do not place state database replicas on fabric-attached storage, SANs, or other storage that is not directly attached to the system and available at the same point in the boot process as traditional SCSI or IDE drives. See `NOTES`.

The metadevice state database contains the configuration of all metadevices and hot spare pools in the system. Additionally, the metadevice state database keeps track of the current state of metadevices and hot spare pools, and their components. Solaris Volume Manager automatically updates the metadevice state database when a configuration or state change occurs. A submirror failure is an example of a state change. Creating a new metadevice is an example of a configuration change.

The metadevice state database is actually a collection of multiple, replicated database copies. Each copy, referred to as a replica, is subject to strict consistency checking to ensure correctness.

Replicated databases have an inherent problem in determining which database has valid and correct data. To solve this problem, Volume Manager uses a *majority consensus algorithm*. This algorithm requires that a majority of the database replicas be available before any of them are declared valid. This algorithm strongly encourages the presence of at least three initial replicas, which you create. A consensus can then be reached as long as at least two of the three replicas are available. If there is only one replica and the system crashes, it is possible that all metadevice configuration data can be lost.

The majority consensus algorithm is conservative in the sense that it will fail if a majority consensus cannot be reached, even if one replica actually does contain the most up-to-date data. This approach guarantees that stale data will not be accidentally used, regardless of the failure scenario. The majority consensus algorithm accounts for the following: the system will

stay running with exactly half or more replicas; the system will panic when less than half the replicas are available; the system will not reboot without one more than half the total replicas.

When used with no options, the `metadb` command gives a short form of the status of the metadevice state database. Use `metadb -i` for an explanation of the flags field in the output.

The initial state database is created using the `metadb` command with both the `-a` and `-f` options, followed by the slice where the replica is to reside. The `-a` option specifies that a replica (in this case, the initial) state database should be created. The `-f` option forces the creation to occur, even though a state database does not exist. (The `-a` and `-f` options should be used together only when no state databases exist.)

Additional replicas beyond those initially created can be added to the system. They contain the same information as the existing replicas, and help to prevent the loss of the configuration information. Loss of the configuration makes operation of the metadevices impossible. To create additional replicas, use the `metadb -a` command, followed by the name of the new slice(s) where the replicas will reside. All replicas that are located on the same slice must be created at the same time.

To delete all replicas that are located on the same slice, the `metadb -d` command is used, followed by the slice name.

When used with the `-i` option, `metadb` displays the status of the metadevice state databases. The status can change if a hardware failure occurs or when state databases have been added or deleted.

To fix a replica in an error state, delete the replica and add it back again.

The metadevice state database (`mddb`) also contains a list of the replica locations for this set (local or shared diskset).

The local set `mddb` can also contain host and drive information for each of the shared disksets of which this node is a member. Other than the diskset host and drive information stored in the local set `mddb`, the local and shared diskset `mddb`s are functionality identical.

The `mddb`s are written to during the resync of a mirror or during a component failure or configuration change. A configuration change or failure can also occur on a single replica (removal of a `mddb` or a failed disk) and this causes the other replicas to be updated with this failure information.

Options Root privileges are required for all of the following options except `-h` and `-i`.

The following options can be used with the `metadb` command. Not all the options are compatible on the same command line. Refer to the `SYNOPSIS` to see the supported use of the options.

`-a`
Attach a new database device. The `/kernel/drv/md.conf` file is automatically updated with the new information and the `/etc/lvm/mddb.cf` file is updated as well. An alternate way to create replicas is by defining them in the `/etc/lvm/md.tab` file and specifying the assigned name at the command line in the form, `mddb`*nn*, where *nn* is a two-digit number given to the replica definitions. Refer to the `md.tab(4)` man page for instructions on setting up replicas in that file.

`-c` *number*
Specifies the number of replicas to be placed on each device. The default number of replicas is 1.

`-d`
Deletes all replicas that are located on the specified *slice*. The `/kernel/drv/md.conf` file is automatically updated with the new information and the `/etc/lvm/mddb.cf` file is updated as well.

`-f`
The `-f` option is used to create the initial state database. It is also used to force the deletion of replicas below the minimum of one. (The `-a` and `-f` options should be used together only when no state databases exist.)

`-h`
Displays a usage message.

`-i`
Inquire about the status of the replicas. The output of the `-i` option includes characters in front of the device name that represent the status of the state database. Explanations of the characters are displayed following the replica status and are as follows:

`d`
replica does not have an associated device ID.

`o`
replica active prior to last `mddb` configuration change

`u`
replica is up to date

`l`
locator for this replica was read successfully

`c`
replica's location was in `/etc/lvm/mddb.cf`

`p`
replica's location was patched in kernel

`m`
replica is master, this is replica selected as input

`r`
replica does not have device relocation information

`t`
tagged data is associated with the replica

`W`
replica has device write errors

`a`
replica is active, commits are occurring to this

`M`
replica had problem with master blocks

`D`
replica had problem with data blocks

`F`
replica had format problems

`S`
replica is too small to hold current database

`R`
replica had device read errors

`B`
tagged data associated with the replica is not valid

`-k` *system-file*
Specifies the name of the kernel file where the replica information should be written. The default *system-file* is `/kernel/drv/md.conf`. This option is for use with the local diskset only.

`-l` *length*
Specifies the size of each replica. The default *length* is 8192 blocks, which should be appropriate for most configurations. "Replica" sizes of less than 128 blocks are not recommended.

`-p`
Specifies updating the system file (`md.conf`) in the current working directory with entries from the `/etc/lvm/mddb.cf` file. This option is normally used to update a newly built system before it is booted for the first time. If the system has been built on a system other than the one where it will run, the location of the `mddb.cf` on the local machine can be passed as an argument. The system file to be updated can be changed using the `-k` option. This option is for use with the local diskset only.

`-s` *setname*
Specifies the name of the diskset on which the `metadb` command will work. Using the `-s` option will cause the command to perform its administrative function within the specified diskset. Without this option, the command will perform its function on local database replicas.

slice
Specifies the logical name of the physical slice (partition), such as `/dev/dsk/c0t0d0s3`.

Examples

EXAMPLE 1 Creating Initial State Database Replicas

The following example creates the initial state database replicas on a new system.

```
# metadb -a -f c0t0d0s7 c0t1d0s3 c1t0d0s7 c1t1d0s3
```

The `-a` and `-f` options force the creation of the initial database and replicas. You could then create metadevices with these same slices, making efficient use of the system.

EXAMPLE 2 Adding Two Replicas on Two New Disks

This example shows how to add two replicas on two new disks that have been connected to a system currently running Volume Manager.

```
# metadb -a c0t2d0s3 c1t1d0s3
```

EXAMPLE 3 Deleting Two Replicas

This example shows how to delete two replicas from the system. Assume that replicas have been set up on `/dev/dsk/c0t2d0s3` and `/dev/dsk/c1t1d0s3`.

```
# metadb -d c0t2d0s3 c1t1d0s3
```

Although you can delete all replicas, you should never do so while metadevices still exist. Removing all replicas causes existing metadevices to become inoperable.

Files

`/etc/lvm/mddb.cf`
Contains the location of each copy of the metadevice state database.

`/etc/lvm/md.tab`
Workspace file for metadevice database configuration.

`/kernel/drv/md.conf`
Contains database replica information for all metadevices on a system. Also contains Solaris Volume Manager configuration information.

Exit Status

The following exit values are returned:

`0`
successful completion

`>0`
an error occurred

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	storage/svm
Interface Stability	Committed

See Also mdmonitord(1M), metaclear(1M), metadetach(1M), metahs(1M), metainit(1M), metaoffline(1M), metaonline(1M), metaparam(1M), metarecover(1M), metarename(1M), metareplace(1M), metaset(1M), metassist(1M), metastat(1M), metasync(1M), metattach(1M), md.tab(4), md.cf(4), mddb.cf(4), md.tab(4), attributes(5), md(7D)

Solaris Volume Manager Administration Guide

Notes Replicas cannot be stored on fabric-attached storage, SANs, or other storage that is not directly attached to the system. Replicas must be on storage that is available at the same point in the boot process as traditional SCSI or IDE drives. A replica can be stored on a:

- Dedicated local disk partition
- Local partition that will be part of a volume
- Local partition that will be part of a UFS logging device

Name metadevadm – update metadevice information

Synopsis

```
/usr/sbin/metadevadm [-h] [-n] [ [-l]-r] [-s setname]
     [-u disk_specifier] [-v]
```

Description The `metadevadm` command facilitates the administration of device ID entries in Solaris Volume Manager. Use this command when the pathname stored in the metadevice state database no longer correctly addresses the device or when a disk drive has had its device ID changed.

This command requires root privileges.

Options The following options are supported.

`-h` Provide a help display.

`-l` Specify that `metadevadm` log to `syslog(3C)`. `metadevadm` logs to the `DAEMON` facility at the `ERR` level by default. See `syslog.conf(4)` for additional information on changing logging levels.

Use this option anytime. It is most useful in startup scripts and less useful interactively.

`-n` Emulate the effect of a command, without making any changes to the system.

`-r` Recompute the pathname and disk specifier (including slice) associated with all devices in the metadevice state database if the device supports device IDs. If a device does not support device IDs or the device is not available, then no action is taken for that device.

Use this option when the disk has been moved or readdressed.

This option is run automatically at boot time to detect device ID changes and update the state database.

`-s` *setname* Specify the name of the disk set on which `metadevadm` works. This option causes the command to perform its administrative function within the specified disk set. Without this option, the command performs its function on devices in the local disk set.

`-u` *disk_specifier* Obtain the device ID associated with the *disk_specifier* (for example, `c1t2d0`) of a device and update the metadevice state database. If the device ID has not changed this option does nothing. Use this option when a disk drive has had its device ID changed during a firmware upgrade or due to changing the controller of a storage subsystem.

`-v` Execute in verbose mode. This option has no effect when used with `-u`. Verbose is the default.

Examples

EXAMPLE 1 Updating Device ID of Disk

The following example updates the device `c2t3d0`:

```
# metadevadm -u c2t3d0
Updating SLVM device relocation information for c2t3d0.
Old device reloc information: id1928019239129312301201201001201209139 8
New device reloc information: id1938019239129312301201201001201209139 8
```

The following example is a variation of the preceding, using the full pathname.

```
# metadevadm -u /dev/dsk/c2t3d0
```

The following example uses the `-n` option, which means that the command is emulated, but does not take effect. Note that when the `-v` option is used with `-u`, `-v` has no effect (verbose is the default).

```
# metadevadm -u -v -n c2t3d0
Updating SLVM device relocation information for c2t3d0.
Old device reloc information: id1928019239129312301201201001201209139 8
New device reloc information: id1938019239129312301201201001201209139 8
```

EXAMPLE 2 Recomputing Pathnames

In the following example, all device names are valid.

```
# metadevadm -r
Disk movement detected.
Updating device names in SLVM.
```

In the following example, once again device names are valid.

```
# metadevadm -r -v
Disk movement detected.
Updating device names in SLVM.
c0t0d0s0 changed to c0t0d1s0 from device relocation information
id12098123lkmklsdjaasdkfjadfjakds
```

In the following example, `metadevadm` detects an invalid device name.

```
# metadevadm -r
Invalid device relocation information detected in SLVM.
Please check status of following disk(s):
c3t0d0
```

Return Values

The following exit values are returned:

`0` Command was successful.

`1` `metadevadm` encountered an error condition.

2 An invalid device ID was detected when using the `-r` option. This is for use in the `rc2.d` script. See `init.d(4)`.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	storage/svm
Interface Stability	Committed

See Also `mdmonitord(1M)`, `metaclear(1M)`, `metadb(1M)`, `metadetach(1M)`, `metahs(1M)`, `metainit(1M)`, `metaoffline(1M)`, `metaonline(1M)`, `metaparam(1M)`, `metarecover(1M)`, `metarename(1M)`, `metareplace(1M)`, `metaset(1M)`, `metassist(1M)`, `metastat(1M)`, `metasync(1M)`, `metattach(1M)`, `md.tab(4)`, `md.cf(4)`, `mddb.cf(4)`, `md.tab(4)`, `attributes(5)`, `md(7D)`

Solaris Volume Manager Administration Guide

Name metahs – manage hot spares and hot spare pools

Synopsis

```
/usr/sbin/metahs [-s setname] -a all component

/usr/sbin/metahs [-s setname] -a hot_spare_pool [component]

/usr/sbin/metahs [-s setname] -d hot_spare_pool [component]

/usr/sbin/metahs [-s setname] -d all component

/usr/sbin/metahs [-s setname] -e component

/usr/sbin/metahs [-s setname] -r hot_spare_pool component-old

/usr/sbin/metahs [-s setname] -r all component-old component-new

/usr/sbin/metahs [-s setname] -i [hot_spare_pool]...
```

Description The `metahs` command manages existing hot spares and hot spare pools. It is used to add, delete, enable, and replace components (slices) in hot spare pools. Like the `metainit` command, the `metahs` command can also create an initial hot spare pool. The `metahs` command does not replace a component of a metadevice. This function is performed by the `metareplace` command.

Hot spares are always in one of three states: available, in-use, or broken. Available hot spares are running and ready to accept data, but are not currently being written to or read from. In-use hot spares are currently being written to and read from. Broken hot spares are out of service and should be repaired. The status of hot spares is displayed when `metahs` is invoked with the `-i` option.

Solaris Volume Manager supports storage devices and logical volumes, including hot spares, greater than 1 terabyte (TB) when Solaris 10 is running a 64-bit kernel.

If a system with large volumes or hot spares is rebooted under a 32-bit Solaris 10 kernel, the large volumes are visible through `metastat` output, but they cannot be accessed, modified or deleted, and no new large volumes can be created. Any volumes or file systems on a large volume in this situation are also unavailable. If a system with large volumes is rebooted under a version of Solaris prior to Solaris 10, Solaris Volume Manager will not start. All large volumes must be removed before Solaris Volume Manager runs under another version of the Solaris Operating Environment.

Options Root privileges are required for any of the following options except `-i`.

The following options are supported:

`-a all` *component*
: Add *component* to all hot spare pools. `all` is not case sensitive.

`-a` *hot_spare_pool* `[`*component*`]`
: Add the *component* to the specified *hot_spare_pool*. *hot_spare_pool* is created if it does not already exist.

`-d all` *component*	Delete *component* from all the hot spare pools. The *component* cannot be deleted if it is in the in-use state.
`-d` *hot_spare_pool* `[`*component*`]`	Delete *hot_spare_pool*, if the *hot_spare_pool* is both empty and not referenced by a metadevice. If *component* is specified, it is deleted from the *hot_spare_pool*. Hot spares in the in-use state cannot be deleted.
`-e` *component*	Enable *component* to be available for use as a hot spare. The *component* can be enabled if it is in the broken state and has been repaired.
`-i [`*hot_spare_pool* `. . .]`	Display the status of the specified *hot_spare_pool* or for all hot spare pools if one is not specified.
`-r all` *component-old component-new*	Replace *component-old* with *component-new* in all hot spare pools which have the component associated. Components cannot be replaced from any hot spare pool if the old hot spare is in the in-use state.
`-r` *hot_spare_pool component-old component-new*	Replace *component-old* with *component-new* in the specified *hot_spare_pool*. Components cannot be replaced from a hot spare pool if the old hot spare is in the in-use state.
`-s` *setname*	Specify the name of the diskset on which `metahs` works. Using the `-s` option causes the command to perform its administrative function within the specified diskset. Without this option, the command performs its function on local hot spare pools.

Operands The following operands are supported:

component	The logical name for the physical slice (partition) on a disk drive, such as `/dev/dsk/c0t0d0s2`.
hot_spare_pool	Names for hot spare pools can be any legal file name that is composed of alphanumeric characters, a dash ("-"), an underscore ("_"), or a period

(“.“). Names must begin with a letter. The words “all” and “none” are reserved and cannot be used.

Examples

EXAMPLE 1 Adding a Hot Spare to a Hot Spare Pool

The following example adds a hot spare `/dev/dsk/c0t0d0s7` to a hot spare pool `mirror1_pool`:

```
# metahs -a mirror1_pool c0t0d0s7
```

When the hot spare is added to the pool, the existing order of the hot spares already in the pool is preserved. The new hot spare is added at the end of the list of hot spares in the hot spare pool specified.

EXAMPLE 2 Adding a Hot Spare to All Currently Defined Pools

This example adds a hot spare to the hot spare pools that are currently defined:

```
# metahs -a all c0t0d0s7
```

The keyword `all` in this example specifies adding the hot spare, `/dev/dsk/c0t0d0s7`, to all the hot spare pools.

EXAMPLE 3 Deleting a Hot Spare

This example deletes a hot spare, `/dev/dsk/c0t0d0s7`, from a hot spare pool, `hsp003`:

```
# metahs -d hsp003 c0t0d0s7
```

When you delete a hot spare, the position of the remaining hot spares in the pool changes to reflect the new order. For instance, if in this example `/dev/dsk/c0t0d0s7` were the second of three hot spares, after deletion the third hot spare would move to the second position.

EXAMPLE 4 Replacing a Hot Spare

This example replaces a hot spare that was previously defined:

```
# metahs -r hsp001 c0t1d0s0 c0t3d0s0
```

In this example, the hot spare `/dev/dsk/c0t1d0s0` is replaced by `/dev/dsk/c0t3d0s0`. The order of the hot spares does not change.

Exit Status

The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	storage/svm
Interface Stability	Committed

See Also mdmonitord(1M), metaclear(1M), metadb(1M), metadetach(1M), metainit(1M), metaoffline(1M), metaonline(1M), metaparam(1M), metarecover(1M), metarename(1M), metareplace(1M), metaset(1M), metassist(1M), metastat(1M), metasync(1M), metattach(1M), md.tab(4), md.cf(4), mddb.cf(4), md.tab(4), attributes(5), md(7D)

Solaris Volume Manager Administration Guide

Warnings Do not create large (>1 TB) volumes if you expect to run the Solaris Operating Environment with a 32-bit kernel or if you expect to use a version of the Solaris Operating Environment prior to Solaris 10.

Name metaimport – imports disk sets into existing Solaris Volume Manager configurations

Synopsis

```
metaimport -s setname [-n] [-f] [-v] [disks]...

metaimport -r [-v] [disks]...

metaimport -V

metaimport -?
```

Description The `metaimport` command allows the importing of disk sets, including replicated disk sets, into an existing Solaris Volume Manager configuration. Replicated disk sets are disk sets created using remote replication software.

The default Solaris Volume Manager configuration specifies a maximum number of disk sets that can be configured. The `metaimport` command fails if importing the disk set would result in exceeding the number of disk sets configured on the system. To increase the number of disk sets allowed on a system, see the *Solaris Volume Manager Administration Guide*.

Use `metaset(1M)` or `metastat(1M)` to view the configuration of the imported set.

You must run `metaimport` as root.

`metaimport` requires a functional Solaris Volume Manager configuration before it runs.

Options The following options are supported:

`-f` Force the import, even if a quorum of replicas from the imported disk set is not available. This option could result in corrupt configurations and should only be used when `metaimport` fails with the "Insufficient quorum detected; exiting" error. If only a partial disk set is available, this option might be necessary to successfully import. Some or all data could be corrupted or unavailable when importing a partial set or a set lacking a replica quorum.

`-n` Does not actually perform the operation, but shows the output or errors that would have resulted from the operation, had it been run.

`-r` Report on the non-configured disk sets found on the system. If no disk device or LUN is specified, `metaimport` reports on all non-configured disk sets attached to the system. When the name of one disk is specified, `metaimport` reports on the disk set (or virtual LUN) containing the specified disk. If two or more disks are specified, metaimport reports on the set (or sets, if they belong to different disk sets) containing the specified disks. If two or more disks are specified, `metaimport` reports on the set (or sets, if they belong to different disk sets) containing the specified disks.

This option can be used in conjunction with the `-v` option to give verbose output on each disk set reported.

`-s` *setname*	Specify the disk set name to use when importing. The imported disk set will be called *setname*, without regard to the name it may have had on a different system.
`-v`	Verbose. Provides detailed information about the `metadb` replica location and status. It also provides detailed information about the disk set configuration and status similar to the "`metastat –c`" output.
`-V`	Version information.
`-?`	Display a help message.

Examples

EXAMPLE 1 Importing a Disk Set

The following example creates a disk set called `blue` and identifies `c1t5d0` as a disk containing a state database replica from the disk set being imported.

```
# metaimport -s blue c1t5d0
```

EXAMPLE 2 Reporting Disk Sets to Import

The following example scans all disks and LUNs attached to the system and configured as part of the system. It scans for disks that could be part of a disk set to be imported. Components that are already part of the Solaris Volume Manager configuration are ignored.

This use of `metaimport` provides suggested forms of the `metaimport` command to use to actually import the disk sets that have been found. You can specify a component on the command line to reduce the scope of the scan and generate results more quickly.

```
 # metaimport -r
```

Exit Status

`0`	Successful completion.
`>0`	An error occurred.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	storage/svm
Interface Stability	Committed

See Also

`mdmonitord(1M)`, `metaclear(1M)`, `metadb(1M)`, `metadetach(1M)`, `metahs(1M)`, `metainit(1M)`, `metaoffline(1M)`, `metaonline(1M)`, `metaparam(1M)`, `metarecover(1M)`, `metareplace(1M)`, `metaset(1M)`, `metastat(1M)`, `metasync(1M)`, `metattach(1M)`, `md.tab(4)`, `md.cf(4)`, `mddb.cf(4)`, `attributes(5)`

Solaris Volume Manager Administration Guide

Name metainit – configure metadevices

Synopsis

```
/usr/sbin/metainit -h

/usr/sbin/metainit [generic options] concat/stripe numstripes width
     component... [-i interlace]

/usr/sbin/metainit [width component... [-i interlace]]
     [-h hot_spare_pool]

/usr/sbin/metainit [generic options] mirror -m submirror
     [read_options] [write_options]
     [pass_num]

/usr/sbin/metainit [generic options] RAID -r component...
     [-i interlace]
     [-h hot_spare_pool] [-k] [-o original_column_count]

/usr/sbin/metainit [generic options] hot_spare_pool
     [hotspare...]

/usr/sbin/metainit [generic options] metadevice-name

/usr/sbin/metainit [generic options] -a

/usr/sbin/metainit [generic options] softpart -p [-e] component
     [-A alignment] size

/usr/sbin/metainit -r
```

Description The `metainit` command configures metadevices and hot spares according to the information specified on the command line. Alternatively, you can run `metainit` so that it uses configuration entries you specify in the `/etc/lvm/md.tab` file (see `md.tab(4)`). All metadevices must be set up by the `metainit` command before they can be used.

Solaris Volume Manager supports storage devices and logical volumes greater than 1 terabyte (TB) when a system runs a 64-bit Solaris kernel. Support for large volumes is automatic. If a device greater than 1 TB is created, Solaris Volume Manager configures it appropriately and without user intervention.

If a system with large volumes is rebooted under a 32–bit Solaris kernel, the large volumes are visible through `metastat` output. Large volumes cannot be accessed, modified or deleted, and no new large volumes can be created. Any volumes or file systems on a large volume in this situation are unavailable. If a system with large volumes is rebooted under a version of Solaris prior to the Solaris 9 4/03 release, Solaris Volume Manager does not start. You must remove all large volumes before Solaris Volume Manager runs under an earlier version of the Solaris Operating System.

If you edit the `/etc/lvm/md.tab` file to configure metadevices, specify one complete configuration entry per line. You then run the `metainit` command with either the `-a` option, to activate all metadevices you entered in the `/etc/lvm/md.tab` file, or with the metadevice name corresponding to a specific configuration entry.

`metainit` does not maintain the state of the volumes that would have been created when `metainit` is run with both the `-a` and `-n` flags. Any volumes in `md.tab` that have dependencies on other volumes in `md.tab` are reported as errors when `metainit -a -n` is run, although the operations might succeed when `metainit -a` is run. See `md.tab(4)`.

Solaris Volume Manager never updates the `/etc/lvm/md.tab` file. Complete configuration information is stored in the metadevice state database, not `md.tab`. The only way information appears in `md.tab` is through editing it by hand.

When setting up a disk mirror, the *first* step is to use `metainit` create a one-on-one concatenation for the named slice. See `EXAMPLES`.

Options The following options are supported:

Generic Options Root privileges are required for all of the following options except `-h`.

The following generic options are supported:

`-f`
Forces the `metainit` command to continue even if one of the slices contains a mounted file system or is being used as `swap`, or if the stripe being created is smaller in size than the underlying soft partition. This option is required when configuring mirrors on `swap`.

`-h`
Displays usage message.

`-n`
Checks the syntax of your command line or `md.tab` entry without actually setting up the metadevice. If used with `-a`, all devices are checked but not initialized.

`-r`
Only used in a shell script at boot time. Sets up all metadevices that were configured before the system crashed or was shut down. The information about previously configured metadevices is stored in the metadevice state database (see `metadb(1M)`).

`-s` *setname*
Specifies the name of the diskset on which `metainit` works. Without the `-s` option, the `metainit` command operates on your local metadevices and/or hotspares.

Concat/Stripe Options The following concat/stripe options are supported:

concat/stripe
Specifies the metadevice name of the concatenation, stripe, or concatenation of stripes being defined.

numstripes
Specifies the number of individual stripes in the metadevice. For a simple stripe, *numstripes* is always 1. For a concatenation, *numstripes* is equal to the number of slices. For a concatenation of stripes, *numstripes* varies according to the number of stripes.

width
Specifies the number of slices that make up a stripe. When *width* is greater than 1, the slices are striped.

component
The logical name for the physical slice (partition) on a disk drive, such as `/dev/dsk/c0t0d0s0`. For RAID level 5 metadevices, a minimum of three slices is necessary to enable striping of the parity information across slices.

`-i` *interlace*
Specifies the interlace size. This value tells Solaris Volume Manager how much data to place on a slice of a striped or RAID level 5 metadevice before moving on to the next slice. *interlace* is a specified value, followed by either 'k' for kilobytes, 'm' for megabytes, or 'b' for blocks. The characters can be either uppercase or lowercase. The *interlace* specified cannot be less than 16 blocks, or greater than 100 megabytes. If *interlace* is not specified, it defaults to 512 kilobytes.

`-h` *hot_spare_pool*
Specifies the *hot_spare_pool* to be associated with the metadevice. If you use the command line, the hot spare pool must have been previously created by the `metainit` command before it can be associated with a metadevice. Use `/-h hsp`*nnn* when the concat/stripe being created is to be used as a submirror.

Names for hot spare pools can be any legal file name that is composed of alphanumeric characters, a dash ("-"), an underscore ("_"), or a period (".."). Names must begin with a letter. The words "all" and "none" are reserved and cannot be used.

Mirror Options
The following mirror options are supported:

mirror `-m` *submirror*
Specifies the metadevice name of the mirror. The `-m` indicates that the configuration is a mirror. *submirror* is a metadevice (stripe or concatentation) that makes up the initial one-way mirror. Solaris Volume Manager supports a maximum of four-way mirroring. When defining mirrors, first create the mirror with the `metainit` command as a one-way mirror. Then attach subsequent submirrors using the `metattach` command. This method ensures that Solaris Volume Manager properly syncs the mirrors. (The second and any subsequent submirrors are first created using the `metainit` command.)

read_options
The following read options for mirrors are supported:

`-g`
Enables the geometric read option, which results in faster performance on sequential reads.

`-r`
Directs all reads to the first submirror. This should only be used when the devices comprising the first submirror are substantially faster than those of the second mirror. This flag cannot be used with the `-g` flag.

If neither the `-g` nor `-r` flags are specified, reads are made in a round-robin order from all submirrors in the mirror. This enables load balancing across the submirrors.

write_options
The following write options for mirrors are supported:

`-S`
Performs serial writes to mirrors. The first submirror write completes before the second is started. This can be useful if hardware is susceptible to partial sector failures. If `-S` is not specified, writes are replicated and dispatched to all mirrors simultaneously.

pass_num
A number in the range 0-9 at the end of an entry defining a mirror that determines the order in which that mirror is resynced during a reboot. The default is 1. Smaller pass numbers are resynced first. Equal pass numbers are run concurrently. If 0 is used, the resync is skipped. 0 should be used only for mirrors mounted as read-only, or as `swap`.

RAID Level 5 Options

The following RAID level 5 options are available:

RAID `-r`
Specifies the name of the RAID level 5 metadevice. The `-r` specifies that the configuration is RAID level 5.

`-k`
For RAID level 5 metadevices, informs the driver that it is not to initialize (zero the disk blocks) due to existing data. Only use this option to recreate a previously created RAID level 5 device.

Use the `-k` option with extreme caution. This option sets the disk blocks to the `OK` state. If any errors exist on disk blocks within the metadevice, Solaris Volume Manager might begin fabricating data. Instead of using the `-k` option, you might want to initialize the device and restore data from tape.

`-o` *original_column_count*
For RAID level 5 metadevices, used with the `-k` option to define the number of original slices in the event the originally defined metadevice was grown. This is necessary since the parity segments are not striped across concatenated devices.

Use the `-o` option with extreme caution. This option sets the disk blocks to the `OK` state. If any errors exist on disk blocks within the metadevice, Solaris Volume Manager might begin fabricating data. Instead of using the `-o` option, you might want to initialize the device and restore data from tape.

Soft Partition Options

The following soft partition options are supported:

softpart `-p` [`-e`] *component* [`-A` *alignment*] *size*
The *softpart* argument specifies the name of the soft partition. The `-p` specifies that the configuration is a soft partition.

The `-e` specifies that the entire disk specified by *component* as `c*t*d*` should be repartitioned and reserved for soft partitions. The specified component is repartitioned such that slice 7 reserves space for system (state database replica) usage and slice 0 contains all remaining space on the disk. Slice 7 is a minimum of 4MB, but can be larger, depending on the disk geometry. The newly created soft partition is placed on slice 0 of the device.

The *component* argument specifies the disk (`c*t*d*`), slice (`c*t*d*s*`), or meta device (`d*`) from which to create the soft partition. The *size* argument determines the space to use for the soft partition and can be specified in `K` or `k` for kilobytes, `M` or `m` for megabytes, `G` or `g` for gigabytes, `T` or `t` for terabyte (one terabyte is the maximum size), and `B` or `b` for blocks (sectors). All values represent powers of 2, and upper and lower case options are equivalent. Only integer values are permitted.

The `-A` alignment option sets the value of the soft partition extent alignment. This option used when it is important specify a starting offset for the soft partition. It preserves the data alignment between the metadevice address space and the address space of the underlying physical device. For example, a hardware device that does checksumming should not have its I/O requests divided by Solaris Volume Manager. In this case, use a value from the hardware configuration as the value for the alignment. When you use this option in conjunction with a software I/O load, the alignment value corresponds to the I/O load of the application. This prevents I/O from being divided unnecessarily and affecting performance.

The literal `all`, used instead of specifying size, specifies that the soft partition should occupy all available space on the device.

Hot Spare Pool Options

The following hot spare pool options are supported:

hot_spare_pool [*hotspare...*]
: When used as arguments to the `metainit` command, *hot_spare_pool* defines the name for a hot spare pool, and *hotspare...* is the logical name for the physical slice(s) for availability in that pool. Names for hot spare pools can be any legal file name that is composed of alphanumeric characters, a dash ("-"), an underscore ("_"), or a period ("."). Names must begin with a letter. The words "all" and "none" are reserved and cannot be used.

`md.tab` File Options

The following `md.tab` file options are supported:

metadevice-name
: When the `metainit` command is run with a *metadevice-name* as its only argument, it searches the `/etc/lvm/md.tab` file to find that name and its corresponding entry. The order in which entries appear in the `md.tab` file is unimportant. For example, consider the following `md.tab` entry:

```
d0 2 1 c1t0d0s0 1 c2t1d0s0
```

When you run the command `metainit d0`, it configures metadevice `d0` based on the configuration information found in the `md.tab` file.

`-a`
Activates all metadevices defined in the `md.tab` file.

`metainit` does not maintain the state of the volumes that would have been created when `metainit` is run with both the `-a` and `-n` flags. If a device `d0` is created in the first line of the `md.tab` file, and a later line in `md.tab` assumes the existence of `d0`, the later line fails when `metainit -an` runs (even if it would succeed with `metainit -a`).

Examples

EXAMPLE 1 Creating a One-on-One Concatenation

The following command creates a one-on-one concatenation for the root slice. This is the first step you take when setting up a mirror for any other slice that cannot be unmounted. The `-f` option is required to create a volume with an existing file system.

```
# metainit -f d1 1 1 c0t0d0s0
```

The preceding command makes `d1` a one-on-one concatenation, using the named slice. You can then enter:

```
# metainit d0 -m d1
```

...to make a one-way mirror of the named slice.

EXAMPLE 2 Concatenation

All drives in the following examples have the same size of 525 Mbytes.

This example shows a metadevice, `/dev/md/dsk/d7`, consisting of a concatenation of four slices.

```
# metainit d7 4 1 c0t1d0s0 1 c0t2d0s0 1 c0t3d0s0 1 /dev/dsk/c0t4d0s0
```

The number 4 indicates there are four individual stripes in the concatenation. Each stripe is made of one slice, hence the number 1 appears in front of each slice. The first disk sector in all of these devices contains a disk label. To preserve the labels on devices `/dev/dsk/c0t2d0s0`, `/dev/dsk/c0t3d0s0`, and `/dev/dsk/c0t4d0s0`, the metadisk driver must skip at least the first sector of those disks when mapping accesses across the concatenation boundaries. Because skipping only the first sector would create an irregular disk geometry, the entire first cylinder of these disks is skipped. This allows higher level file system software to optimize block allocations correctly.

EXAMPLE 3 Stripe

This example shows a metadevice, `/dev/md/dsk/d15`, consisting of two slices.

```
# metainit d15 1 2 c0t1d0s0 c0t2d0s0 -i 32k
```

The number 1 indicates that one stripe is being created. Because the stripe is made of two slices, the number 2 follows next. The optional `-i` followed by `32k` specifies the interlace size as 32 Kbytes. If the interlace size were not specified, the stripe would use the default value of 16

EXAMPLE 3 Stripe *(Continued)*

Kbytes.

EXAMPLE 4 Concatentation of Stripes

This example shows a metadevice, `/dev/md/dsk/d75`, consisting of a concatenation of two stripes of three disks.

```
# metainit d75 2 3 c0t1d0s0 c0t2d0s0 \
      c0t3d0s0 -i 16k \
      3 c1t1d0s0 c1t2d0s0 c1t3d0s0 -i 32k
```

On the first line, the `-i` followed by 16k specifies that the stripe interlace size is 16 Kbytes. The second set specifies the stripe interlace size as 32 Kbytes. If the second set did not specify 32 Kbytes, the set would use the default interlace value of 16 Kbytes. The blocks of each set of three disks are interlaced across three disks.

EXAMPLE 5 Mirroring

This example shows a two-way mirror, `/dev/md/dsk/d50`, consisting of two submirrors. This mirror does not contain any existing data.

```
# metainit d51 1 1 c0t1d0s0
# metainit d52 1 1 c0t2d0s0
# metainit d50 -m d51
# metattach d50 d52
```

In this example, two submirrors, `d51` and `d52`, are created with the `metainit` command. These two submirrors are simple concatenations. Next, a one-way mirror, `d50`, is created using the `-m` option with `d51`. The second submirror is attached later using the `metattach` command. When creating a mirror, any combination of stripes and concatenations can be used. The default read and write options in this example are a round-robin read algorithm and parallel writes to all submirrors.

EXAMPLE 6 Creating a metadevice in a diskset

This example shows a metadevice, `/dev/md/dsk/d75`, consisting of a concatenation of two stripes within a diskset called `set1`.

```
# metainit -s set1 d75 2 3 c2t1d0s0 c2t2d0s0 \
      c2t3d0s0 -i 32k
# metainit -s set1 d51 1 1 c2t1d0s0
# metainit -s set1 d52 1 1 c3t1d0s0
# metainit -s set1 d50 -m d51
# metattach -s set1 d50 d52
```

In this example, a diskset is created using the `metaset` command. Metadevices are then created within the diskset using the `metainit` command. The two submirrors, `d51` and `d52`, are simple concatenations. Next, a one-way mirror, `d50`, is created using the `-m` option with

EXAMPLE 6 Creating a metadevice in a diskset *(Continued)*

`d51`. The second submirror is attached later using the `metattach` command. When creating a mirror, any combination of stripes and concatenations can be used. The default read and write options in this example are a round-robin read algorithm and parallel writes to all submirrors.

EXAMPLE 7 RAID Level 5

This example shows a RAID level 5 device, `d80`, consisting of three slices:

```
# metainit d80 -r c1t0d0s0 c1t1d0s0 c1t3d0s0 -i 20k
```

In this example, a RAID level 5 metadevice is defined using the `-r` option with an interlace size of 20 Kbytes. The data and parity segments are striped across the slices, `c1t0d0s0`, `c1t2d0s0`, and `c1t3d0s0`.

EXAMPLE 8 Soft Partition

The following example shows a soft partition device, `d1`, built on metadevice `d100` and 100 Mbytes (indicated by `100M`) in size:

```
# metainit d1 -p d100 100M
```

The preceding command creates a 100 Mbyte soft partition on the `d100` metadevice. This metadevice could be a RAID level 5, stripe, concatenation, or mirror.

EXAMPLE 9 Soft Partition on Full Disk

The following example shows a soft partition device, `d1`, built on disk `c3t4d0`:

```
# metainit d1 -p -e c3t4d0 9G
```

In this example, the disk is repartitioned and a soft partition is defined to occupy all 9 Gbytes of disk `c3t4d0s0`.

EXAMPLE 10 Soft Partition Taking All Available Space

The following example shows a soft partition device, `d1`, built on disk `c3t4d0`:

```
# metainit d1 -p -e c3t4d0 all
```

In this example, the disk is repartitioned and a soft partition is defined to occupy all available disk space on slice `c3t4d0s0`.

EXAMPLE 11 Hot Spare

This example shows a two-way mirror, `/dev/md/dsk/d10`, and a hot spare pool with three hot spare components. The mirror does not contain any existing data.

```
# metainit hsp001 c2t2d0s0 c3t2d0s0 c1t2d0s0
# metainit d41 1 1 c1t0d0s0 -h hsp001
# metainit d42 1 1 c3t0d0s0 -h hsp001
```

EXAMPLE 11 Hot Spare *(Continued)*

```
# metainit d40 -m d41
# metattach d40 d42
```

In this example, a hot spare pool, `hsp001`, is created with three slices from three different disks used as hot spares. Next, two submirrors are created, `d41` and `d42`. These are simple concatenations. The `metainit` command uses the `-h` option to associate the hot spare pool `hsp001` with each submirror. A one-way mirror is then defined using the `-m` option. The second submirror is attached using the `metattach` command.

EXAMPLE 12 Setting the Value of the Soft Partition Extent Alignment

This example shows how to set the alignment of the soft partition to 1 megabyte.

```
# metainit -s red d13 -p c1t3d0s4 -A 1m 4m
```

In this example the soft partition, `d13`, is created with an extent alignment of 1 megabyte. The `metainit` command uses the `-A` option with an alignment of `1m` to define the soft partition extent alignment.

Files

`/etc/lvm/md.tab`
: Contains list of metadevice and hot spare configurations for batch-like creation.

Warnings

This section contains information on different types of warnings.

Devices and Volumes Greater Than 1 TB

Do not create large (>1 TB) volumes if you expect to run the Solaris Operating Environment with a 32–bit kernel or if you expect to use a version of the Solaris Operating Environment prior to Solaris 10.

Multi-Way Mirror

Do not use the `metainit` command to create a multi-way mirror. Rather, create a one-way mirror with `metainit` then attach additional submirrors with `metattach`. When the `metattach` command is not used, no resync operations occur and data could become corrupted.

If you use `metainit` to create a mirror with multiple submirrors, the following message is displayed:

```
WARNING: This form of metainit is not recommended.
The submirrors may not have the same data.
Please see ERRORS in metainit(1M) for additional information.
```

Truncation of Soft Partitions

When creating stripes on top of soft partitions it is possible for the size of the new stripe to be less than the size of the underlying soft partition. If this occurs, `metainit` fails with an error indicating the actions required to overcome the failure.

If you use the `-f` option to override this behavior, the following message is displayed:

```
WARNING: This form of metainit is not recommended.
The stripe is truncating the size of the underlying device.
Please see ERRORS in metainit(1M) for additional information.
```

Write-On-Write Problem

When mirroring data in Solaris Volume Manager, transfers from memory to the disks do not all occur at exactly the same time for all sides of the mirror. If the contents of buffers are changed while the data is in-flight to the disk (called write-on-write), then different data can end up being stored on each side of a mirror.

This problem can be addressed by making a private copy of the data for mirror writes, however, doing this copy is expensive. Another approach is to detect when memory has been modified across a write by looking at the dirty-bit associated with the memory page. Solaris Volume Manager uses this dirty-bit technique when it can. Unfortunately, this technique does not work for raw I/O or direct I/O. By default, Solaris Volume Manager is tuned for performance with the liability that mirrored data might be out of sync if an application does a "write-on-write" to buffers associated with raw I/O or direct I/O. Without mirroring, you were not guaranteed what data would actually end up on media, but multiple reads would return the same data. With mirroring, multiple reads can return different data. The following line can be added to `/etc/system` to cause a stable copy of the buffers to be used for all raw I/O and direct I/O write operations.

```
set md_mirror:md_mirror_wow_flg=0x20
```

Setting this flag degrades performance.

Exit Status

The following exit values are returned:

`0`
: Successful completion.

`>0`
: An error occurred.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	storage/svm
Interface Stability	Committed

See Also

`mdmonitord(1M)`, `metaclear(1M)`, `metadb(1M)`, `metadetach(1M)`, `metahs(1M)`, `metaoffline(1M)`, `metaonline(1M)`, `metaparam(1M)`, `metarecover(1M)`, `metarename(1M)`, `metareplace(1M)`, `metaset(1M)`, `metassist(1M)`, `metastat(1M)`, `metasync(1M)`, `metattach(1M)`, `md.tab(4)`, `md.cf(4)`, `mddb.cf(4)`, `md.tab(4)`, `attributes(5)`, `md(7D)`

Limitations

Recursive mirroring is not allowed; that is, a mirror cannot appear in the definition of another mirror.

Recursive logging is not allowed; that is, a trans metadevice cannot appear in the definition of another metadevice.

Stripes, concatenations, and RAID level 5 metadevices must consist of slices only.

Mirroring of RAID level 5 metadevices is not allowed.

Soft partitions can be built on raw devices, or on stripes, RAID level 5, or mirrors.

RAID level 5 or stripe metadevices can be built directly on soft partitions.

Notes Trans metadevices have been replaced by UFS logging. Existing trans devices are *not* logging--they pass data directly through to the underlying device. See `mount_ufs(1M)` for more information about UFS logging.

Name metaoffline, metaonline – place submirrors offline and online

Synopsis

```
/usr/sbin/metaoffline -h

/usr/sbin/metaoffline [-s setname] [-f] mirror submirror

/usr/sbin/metaonline -h

/usr/sbin/metaonline [-s setname] mirror submirror
```

Description The `metaoffline` command prevents Solaris Volume Manager from reading and writing to the submirror that has been taken offline. While the submirror is offline, all writes to the mirror will be kept track of (by region) and will be written when the submirror is brought back online. The `metaoffline` command can also be used to perform online backups: one submirror is taken offline and backed up while the mirror remains accessible. (However, if this is a two-way mirror, data redundancy is lost while one submirror is offline.) The `metaoffline` command differs from the `metadetach` command because it does not sever the logical association between the submirror and the mirror. To completely remove a submirror from a mirror, use the `metadetach` command.

A submirror that has been taken offline will only remain offline until the `metaonline` command is invoked or the system is rebooted.

When the `metaonline` command is used, reading from and writing to the submirror resumes. A resync is automatically invoked to resync the regions written while the submirror was offline. Writes are directed to the submirror during resync. Reads, however, will come from a different submirror. Once the resync operation completes, reads and writes are performed on that submirror. The `metaonline` command is only effective on a submirror of a mirror that has been taken offline.

The `metaoffline` and `metaonline` commands can not be used on RAID 1 volumes in application-based recovery (ABR) mode.

A submirror that has been taken offline with the `metaoffline` command can only be mounted as read-only.

Options Root privileges are required for all of the following options except `-h`.

`-f`	Forces offlining of submirrors that have slices requiring maintenance.
`-h`	Displays usage message.
`-s` *setname*	Specifies the name of the diskset on which `metaoffline` and `metaonline` will work. Using the `-s` option will cause the command to perform its administrative function within the specified diskset. Without this option, the command will perform its function on local metadevices.
mirror	Specifies the metadevice name of the mirror from which the submirror will be either taken offline or put online.

submirror Specifies the metadevice name of the submirror to be either taken offline or put online.

Examples

EXAMPLE 1 Taking a Submirror Offline

This example takes one submirror, `mymirror_sub1`, offline from mirror `mymirror`.

```
# metaoffline mymirror mymirror_sub1
```

Exit Status

The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	storage/svm
Interface Stability	Committed

See Also

`mdmonitord(1M)`, `metaclear(1M)`, `metadb(1M)`, `metadetach(1M)`, `metahs(1M)`, `metainit(1M)`, `metaparam(1M)`, `metarecover(1M)`, `metarename(1M)`, `metareplace(1M)`, `metaset(1M)`, `metassist(1M)`, `metastat(1M)`, `metasync(1M)`, `metattach(1M)`, `md.tab(4)`, `md.cf(4)`, `mddb.cf(4)`, `md.tab(4)`, `attributes(5)`, `md(7D)`

Solaris Volume Manager Administration Guide

Notes

The `metaonline` and `metaoffline` commands are not applicable to mirrors in application-based recovery (ABR) mode.

Name metaparam – modify parameters of metadevices

Synopsis

```
/usr/sbin/metaparam -h

/usr/sbin/metaparam [-s setname]
     [concat/stripe or RAID5 options] concat/stripe RAID

/usr/sbin/metaparam [-s setname] [mirror options] mirror
```

Description The `metaparam` command is used to display or modify current parameters of metadevices.

If just the metadevice is specified as an argument to the `metaparam` command, the current settings are displayed.

The `metaparam` command enables most metadevice (volume) parameters to be changed. Only the interlace value cannot be changed by `metaparam`, because it is established when the metadevice is created and cannot be changed thereafter.

Options Root privileges are required for all of the options.

The following options are supported:

`-h`
Displays usage message.

`-s` *setname*
Specify the name of the diskset on which `metaparam` works. Using the `-s` option causes the command to perform its administrative function within the specified diskset. Without this option, the command performs its function on local metadevices.

Concat/STRIPE Or Raid5 Options

–h*hot_spare_pool* | `none`
Specifies the hot spare pool to be used by a metadevice. If `none` is specified, the metadevice is disassociated with the hot spare pool assigned to it. If the metadevice is currently using a hot spare, then `metaparam` cannot replace the hot spare pool.

concat/stripe | *RAID*
Specifies the metadevice name of the concatenation, stripe, or concatenation of stripes, or of the RAID5 metadevice.

Mirror Options

–`r roundrobin | geometric | first`
Modifies the read option for a mirror. The `-r` option must be followed by either `roundrobin`, `geometric`, or `first`. `roundrobin`, which is the default action under the `metainit` command, specifies reading the disks in a round-robin (load balancing) method. `geometric` allows for faster performance on sequential reads. `first` specifies reading only from the first submirror.

–`w parallel | serial`
Modifies the write option for a mirror. The `-w` option must be followed by either `parallel` or `serial`. `parallel`, the default action under the `metainit` command, specifies that all writes are parallel. `serial` specifies that all writes are serial.

`-p` *pass_number*
A number from 0-to-9 that specifies the order in which a mirror is resynced during reboot. The default is 1. Smaller pass numbers are resynced first. Equal pass numbers are run concurrently. If 0 is used, the mirror resync is skipped. 0 should only be used for mirrors mounted as read-only, or as `swap`.

mirror
Specifies the metadevice name of the mirror.

Examples

EXAMPLE 1 Associating Hot Spare Pool with RAID5 Metadevice

This example associates a hot spare pool, `user_pool`, with a RAID5 metadevice, `user_raid`.

```
# metaparam -h user_pool user_raid
```

EXAMPLE 2 Changing Read Option to Geometric

This example changes the read option on a mirror `d50` from the default of roundrobin to geometric.

```
# metaparam -r geometric d50
```

Exit Status

The following exit values are returned:

`0`
Successful completion.

`>0`
An error occurred.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	storage/svm
Interface Stability	Committed

See Also

`mdmonitord(1M)`, `metaclear(1M)`, `metadb(1M)`, `metadetach(1M)`, `metahs(1M)`, `metainit(1M)`, `metaoffline(1M)`, `metaonline(1M)`, `metarecover(1M)`, `metarename(1M)`, `metareplace(1M)`, `metaset(1M)`, `metassist(1M)`, `metastat(1M)`, `metasync(1M)`, `metattach(1M)`, `md.tab(4)`, `md.cf(4)`, `mddb.cf(4)`, `md.tab(4)`, `attributes(5)`, `md(7D)`

Solaris Volume Manager Administration Guide

Name metarecover – recover soft partition information

Synopsis

```
/usr/sbin/metarecover [-n] [-v] [-s setname] component -p

/usr/sbin/metarecover [-n] [-v] [-s setname] component -p {-d}

/usr/sbin/metarecover [-n] [-v] [-s setname] component -p {-m}
```

Description The `metarecover` command scans a specified component to look for soft partition configuration information and to regenerate the configuration.

Options The following options are supported:

`-d` Recover soft partitions in the metadevice state database from the extent headers on the device. Options `-d` and `-m` are mutually exclusive.

`-m` Regenerate the extent headers and reapplies them to the underlying device based on the soft partitions listed in the metadevice state database. Options `-d` and `-m` are mutually exclusive.

`-n` Do not actually perform the operation. Show the output or errors that would have resulted from the operation, had it been run.

`-p` Regenerate soft partitions based on the metadevice state database or extent headers on the underlying device. If neither `-d` nor `-m` are specified, this option compares the soft partition information in the metadevice state database to the extent headers.

`-s` *setname* Specify the name of the diskset on which `metarecover` works. Using the `s` option causes the command to perform its function within the specified diskset. Without the `-s` option, the `metarecover` command operates on the metadevices and/or hot spare pools in the local diskset.

This option is required to recover former `sps` from a diskset component or raw-device. *setname* must be identical to the former *setname* in which the `sps` were created. The set numbers, however, seem irrelevant.

`-v` Verbose mode, displaying the changes being made.

Operands The following operand is supported:

component Specifies the `c*t*d*s*` number of the disk or slice containing the partitions, or the device name (for example, `d10`) of the metadevice containing the partitions.

component can be a slice name, component name, `/dev/dsk path`, or `/dev/rdsk path`.

Examples

EXAMPLE 1 Updating Metadevice State Database Based on Disk Extent Headers

A disk containing soft partitions is moved from one system to another. The system administrator would like to use the existing soft partitions. `metarecover` updates the metadevice state database based on the extent headers on the disk.

```
# metarecover -v c0t3d0s2 -p -d
```

EXAMPLE 2 Updating Metadevice State Database Based on Incomplete Soft Partition Creation

A system crashes in the middle of creating a new soft partition. The soft partition is in the creating state and the driver does not let that device be opened. `metarecover` rewrites the extent headers for the partially created soft partition and mark it as `Okay`.

```
# metarecover -v c0t3d0s2 -p -m
```

EXAMPLE 3 Updating Extent Headers Based on Metadevice State Database

Someone accidentally overwrote a portion of a disk leaving extent headers destroyed. `metarecover` rewrites the extent headers to ensure a valid soft partition configuration, though user data is not recovered.

```
# metarecover -v d5 -m
```

The following example implements the same command using a descriptive name.

```
# metarecover -v myvolume -m
```

EXAMPLE 4 Validating Soft Partition Configuration

To validate the existing soft partition configuration, use `metarecover` with only the `-p` flag.

```
# metarecover c0t3d0s2 -p
```

Exit Status

The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	storage/svm
Interface Stability	Committed

See Also

`mdmonitord(1M)`, `metaclear(1M)`, `metadb(1M)`, `metadetach(1M)`, `metahs(1M)`, `metainit(1M)`, `metaoffline(1M)`, `metaonline(1M)`, `metaparam(1M)`, `metarename(1M)`, `metareplace(1M)`, `metaset(1M)`, `metassist(1M)`, `metastat(1M)`, `metasync(1M)`, `metattach(1M)`, `md.tab(4)`, `md.cf(4)`, `mddb.cf(4)`, `md.tab(4)`, `attributes(5)`, `md(7D)`

Solaris Volume Manager Administration Guide

Name metarename – rename metadevice or switch layered metadevice names

Synopsis

```
/usr/sbin/metarename [-s setname] metadevice1 metadevice2

/usr/sbin/metarename [-s setname] [-f] -x metadevice1 metadevice2

/usr/sbin/metarename -h
```

Description There are two ways to use `metarename`, one with and one without the `-x` option. The first method (without `-x`) renames an existing metadevice to a new name. This makes managing the metadevice namespace easier. The metadevice being renamed cannot be mounted or open, nor can the new name already exist. For example, to rename a metadevice that contains a mounted file system, you would first need to unmount the file system.

With the second way to use `metarename`, using the `-x` option, `metarename` switches (exchanges) the names of an existing layered metadevice and one of its subdevices. In Solaris Volume Manager terms, a layered metadevice can be either a mirror or a trans metadevice. The `-x` option enables you to switch the metadevice names of a mirror and one of its submirrors, or a trans metadevice and its master device.

`metarename -x` makes it easier to mirror or unmirror an existing stripe or concatenation, and to remove a trans device.

When used to mirror an existing stripe or concatenatation, you must stop access to the device. For example, if the device contains a mounted file system, you must first unmount the file system before doing the rename.

You can also use the `metarename -x` command to untrans a trans metadevice from an existing device. This applies only to the master device. You cannot remove a logging device with `metarename`. Before you can rename a trans device, you must detach the logging device. Then you must stop access to the trans metadevice itself.

You cannot rename or switch metadevices that are in an error state or that have subcomponents in an error state, or metadevices actively using a hot spare replacement.

You can only switch metadevices that have a direct child/parent relationship. You could not, for example, directly exchange a stripe in a mirror that is a master device with the trans metadevice.

You must use the `-f` flag when switching members of a trans metadevice.

Only metadevices can be switched, not slices.

Options The following options are supported:

`-f` Force the switching of trans metadevice members.

`-h` Display a help message.

`-s` *setname*	Specifies the name of the diskset on which `metarename` will work. Using the `-s` option will cause the command to perform its administrative function within the specified diskset. Without this option, the command will perform its function on the local metadevices.
`-x`	Exchange the metadevice names *metadevice1* and *metadevice2*.
metadevice1	Specifies the metadevice to be renamed or switched.
metadevice2	Specifies the target metadevice name for the rename or switch operation.

Examples

EXAMPLE 1 Renaming a Metadevice

This example renames a metadevice named `d10` to `account_records`. Note that `account_records` must not exist for the rename to succeed.

```
# metarename d10 account_records
```

EXAMPLE 2 Creating a Two-Way Mirror

This example creates a two-way mirror from an existing stripe named `d1` with a mounted file system, `/home2`.

```
# metainit d2 1 1 c13d0s1
# metainit -f d20 -m d1
# umount /home2
# metarename -x d20 d1
# metattach d1 d2
# mount /home2
```

First, a second concatenation `d2`, is created. (`d1` already exists.) The `metainit` command creates a one-way mirror, `d20`, from `d1`. Next, you umount the file system and switch `d1` for `d20`, making `d1` the top-level device (mirror). You attach the second submirror, `d2`, to create a two-way mirror. Lastly, you remount the file system.

EXAMPLE 3 Mounting a Mirrored File System on Stripe

This example takes an existing mirror named `d1` with a mounted file system, and ends up with the file system mounted on a stripe `d1`.

```
# umount /fs2
# metarename -x d1 d20
# metadetach d20 d1
# metaclear -r d20
# mount /fs2
```

First, you unmount the file system, then switch the mirror `d1` and its submirror `d20`. This makes the mirror into `d20`. Next, you detach `d1` from `d20`, then delete the mirror `d20` and its other submirror. You then remount the file system.

EXAMPLE 4 Deleting a Trans Metadevice

This example deletes a trans metadevice named `d10` while its mount point is `/myhome`. The master device, which is a stripe, is named `d2`. The logging device, also a stripe, is named `d5`.

```
# umount /myhome
# metadetach d10
# metarename -f -x d10 d2
# metaclear d2
# metaclear d5
# fsck /dev/md/dsk/d10
# mount /myhome
```

You umount the file system first, then detach the trans metadevice's logging device. The trans metadevice is switched with the master device, making the trans metadevice `d2` and the underlying stripe `d10`. You clear the trans metadevice `d2` and the logging device `d5`. `d10` must be `fsck`'d, and then the file system is remounted.

Exit Status

The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	storage/svm
Interface Stability	Committed

See Also

`mdmonitord(1M)`, `metaclear(1M)`, `metadb(1M)`, `metadetach(1M)`, `metahs(1M)`, `metainit(1M)`, `metaoffline(1M)`, `metaonline(1M)`, `metaparam(1M)`, `metarecover(1M)`, `metareplace(1M)`, `metaset(1M)`, `metassist(1M)`, `metastat(1M)`, `metasync(1M)`, `metattach(1M)`, `md.tab(4)`, `md.cf(4)`, `mddb.cf(4)`, `md.tab(4)`, `attributes(5)`, `md(7D)`

Solaris Volume Manager Administration Guide

Limitations

Renaming and exchanging metadevice names can only be used for metadevices. A physical slice cannot be renamed to a metadevice, nor can a metadevice be exchanged with a physical slice name.

Metadevice names are strings of the pattern *d<xyz>* where *xyz* is a value between `0` and `8192`. You cannot use logical names for metadevices.

Notes

Trans metadevices have been replaced by UFS logging. Existing trans devices are *not* logging--they pass data directly through to the underlying device. See `mount_ufs(1M)` for more information about UFS logging.

Name metareplace – enable or replace components of submirrors or RAID5 metadevices

Synopsis

```
/usr/sbin/metareplace -h

/usr/sbin/metareplace [-s setname] -e mirror component

/usr/sbin/metareplace [-s setname] mirror component-old component-new

/usr/sbin/metareplace [-s setname] -e RAID component

/usr/sbin/metareplace [-s setname] [-f] RAID component-old component-new
```

Description The `metareplace` command is used to enable or replace components (slices) within a submirror or a RAID5 metadevice.

When you replace a component, the `metareplace` command automatically starts resyncing the new component with the rest of the metadevice. When the resync completes, the replaced component becomes readable and writable. If the failed component has been hot spare replaced, the hot spare is placed in the available state and made available for other hot spare replacements.

Note that the new component must be large enough to replace the old component.

A component may be in one of several states. The `Last Erred` and the `Maintenance` states require action. Always replace components in the `Maintenance` state first, followed by a resync and validation of data. After components requiring maintenance are fixed, validated, and resynced, components in the `Last Erred` state should be replaced. To avoid data loss, it is always best to back up all data before replacing `Last Erred` devices.

Options Root privileges are required for all of the following options except `-h`.

`-e` Transitions the state of *component* to the available state and resyncs the failed component. If the failed component has been hot spare replaced, the hot spare is placed in the available state and made available for other hot spare replacements. This command is useful when a component fails due to human error (for example, accidentally turning off a disk), or because the component was physically replaced. In this case, the replacement component must be partitioned to match the disk being replaced before running the `metareplace` command.

`-f` Forces the replacement of an errored component of a metadevice in which multiple components are in error. The component determined by the `metastat` display to be in the “Maintenance" state must be replaced first. This option may cause data to be fabricated since multiple components are in error.

`-h` Display help message.

`-s` *setname* Specifies the name of the diskset on which `metareplace` will work. Using the `-s` option will cause the command to perform its administrative

function within the specified diskset. Without this option, the command will perform its function on local metadevices.

mirror The metadevice name of the mirror.

component The logical name for the physical slice (partition) on a disk drive, such as `/dev/dsk/c0t0d0s2`.

component-old The physical slice that is being replaced.

component-new The physical slice that is replacing *component-old*.

RAID The metadevice name of the RAID5 device.

Examples

EXAMPLE 1 Recovering from Error Condition in RAID5 Metadevice

This example shows how to recover when a single component in a RAID5 metadevice is errored.

```
# metareplace d10 c3t0d0s2 c5t0d0s2
```

In this example, a RAID5 metadevice `d10` has an errored component, `c3t0d0s2`, replaced by a new component, `c5t0d0s2`.

EXAMPLE 2 Use of `-e` After Physical Disk Replacement

This example shows the use of the `-e` option after a physical disk in a submirror (a submirror of mirror `mymirror1`, in this case) has been replaced.

```
# metareplace -e mymirror1 c1t4d0s2
```

Note: The replacement disk must be partitioned to match the disk it is replacing before running the `metareplace` command.

Exit Status

The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	storage/svm
Interface Stability	Committed

See Also

`mdmonitord(1M)`, `metaclear(1M)`, `metadb(1M)`, `metadetach(1M)`, `metahs(1M)`, `metainit(1M)`, `metaoffline(1M)`, `metaonline(1M)`, `metaparam(1M)`, `metarecover(1M)`,

metarename(1M), metaset(1M), metassist(1M), metastat(1M), metasync(1M), metattach(1M), md.tab(4), md.cf(4), mddb.cf(4), md.tab(4), attributes(5), md(7D)

Solaris Volume Manager Administration Guide

Name metaset – configure disk sets

Synopsis

```
/usr/sbin/metaset -s setname [-M-a -h hostname]

/usr/sbin/metaset -s setname -A {enable | disable}

/usr/sbin/metaset -s setname [-A {enable | disable}] -a -h hostname...

/usr/sbin/metaset -s setname -a [-l length] [-L] drivename...

/usr/sbin/metaset -s setname -C {take | release | purge}

/usr/sbin/metaset -s setname -d [-f] -h hostname...

/usr/sbin/metaset -s setname -d [-f] drivename...

/usr/sbin/metaset -s setname -j

/usr/sbin/metaset -s setname -r

/usr/sbin/metaset -s setname -w

/usr/sbin/metaset -s setname -t [-f] [-u tagnumber] [y]

/usr/sbin/metaset -s setname -b

/usr/sbin/metaset -s setname -P

/usr/sbin/metaset -s setname -q

/usr/sbin/metaset -s setname -o [-h hostname]

/usr/sbin/metaset [-s setname]

/usr/sbin/metaset [-s setname] -a | -d
     [ [m] mediator_host_list]
```

Description The `metaset` command administers sets of disks in named disk sets. Named disk sets include any disk set that is not in the local set. While disk sets enable a high-availability configuration, Solaris Volume Manager itself does not actually provide a high-availability environment.

A single-owner disk set configuration manages storage on a SAN or fabric-attached storage, or provides namespace control and state database replica management for a specified set of disks.

In a shared disk set configuration, multiple hosts are physically connected to the same set of disks. When one host fails, another host has exclusive access to the disks. Each host can control a shared disk set, but only one host can control it at a time.

When you add a new disk to any disk set, Solaris Volume Manager checks the disk format. If necessary, it repartitions the disk to ensure that the disk has an appropriately configured reserved slice 7 (or slice 6 on an EFI labelled device) with adequate space for a state database replica. The precise size of slice 7 (or slice 6 on an EFI labelled device) depends on the disk geometry. For traditional disk sets, the slice is no less than 4 Mbytes, and probably closer to 6 Mbytes, depending on where the cylinder boundaries lie. For multi-owner disk sets, the slice is

a minimum of 256 Mbytes. The minimal size for slice 7 might change in the future. This change is based on a variety of factors, including the size of the state database replica and information to be stored in the state database replica.

For use in disk sets, disks must have a dedicated slice (six or seven) that meets specific criteria:

- The slice must start at sector 0
- The slice must include enough space for disk label
- The state database replicas cannot be mounted
- The slice does not overlap with any other slices, including slice 2

If the existing partition table does not meet these criteria, or if the -L flag is specified, Solaris Volume Manager repartitions the disk. A small portion of each drive is reserved in slice 7 (or slice 6 on an EFI labelled device) for use by Solaris Volume Manager. The remainder of the space on each drive is placed into slice 0. Any existing data on the disks is lost by repartitioning.

After you add a drive to a disk set, it can be repartitioned as necessary, with the exception that slice 7 (or slice 6 on an EFI labelled device) is not altered in any way.

After a disk set is created and metadevices are set up within the set, the metadevice name is in the following form:

`/dev/md/`*setname*`/{dsk,rdsk}/d`*number*

where *setname* is the name of the disk set, and *number* is the number of the metadevice (`0-127`).

If you have disk sets that you upgraded from Solstice DiskSuite software, the default state database replica size on those sets is 1034 blocks, not the 8192 block size from Solaris Volume Manager. Also, slice 7 on the disks that were added under Solstice DiskSuite are correspondingly smaller than slice 7 on disks that were added under Solaris Volume Manager.

If disks you add to a disk set have acceptable slice 7s (that start at cylinder 0 and that have sufficient space for the state database replica), they are not reformatted.

Hot spare pools within local disk sets use standard Solaris Volume Manager naming conventions. Hot spare pools with shared disk sets use the following convention:

setname/*hot_spare_pool*

where *setname* is the name of the disk set, and *hot_spare_pool* is the name of the hot spare pool associated with the disk set.

Multi–node Environment

To create and work with a disk set in a multi—node environment, `root` must be a member of Group 14 on all hosts, or the `/.rhosts` file must contain an entry for all other host names. This is not required in a SunCluster 3.x enviroment.

Tagged data

Tagged data occurs when there are different versions of a disk set's replicas. This tagged data consists of the set owner's nodename, the hardware serial number of the owner, and the time it was written out to the available replicas. The system administer can use this information to determine which replica contains the correct data.

When a disk set is configured with an even number of storage enclosures and has replicas balanced across them evenly, it is possible that up to half of the replicas can be lost (for example, through a power failure of half of the storage enclosures). After the enclosure that went down is rebooted, half of the replicas are not recognized by SVM. When the set is retaken, the `metaset` command returns an error of "stale databases", and all of the metadevices are in a read-only state.

Some of the replicas that are not recognized need to be deleted. The action of deleting the replicas also causes updates to the replicas that are not being deleted. In a dual hosted disk set environment, the second node can access the deleted replicas instead of the existing replicas when it takes the set. This leads to the possibility of getting the wrong replica record on a disk set take. An error message is displayed, and user intervention is required.

Use the `-q` to query the disk set and the `-t`, `-u`, and `-y`, options to select the tag and take the disk set. See `OPTIONS`.

Mediator Configuration

SVM provides support for a low-end HA solution consisting of two hosts that share only two strings of drives. The hosts in this type of configuration, referred to as *mediators* or mediator hosts, run a special daemon, `rpc.metamedd(1M)`. The mediator hosts take on additional responsibilities to ensure that data is available in the case of host or drive failures.

A mediator configuration can survive the failure of a single host or a single string of drives, without administrative intervention. If both a host and a string of drives fail (multiple failures), the integrity of the data cannot be guaranteed. At this point, administrative intervention is required to make the data accessible. See `mediator(7D)` for further details.

Use the `-m` option to add or delete a mediator host. See `OPTIONS`.

Options

The following options are supported:

`-a` *drivename*

Add drives or hosts to the named set. For a drive to be accepted into a set, the drive must not be in use within another metadevice or disk set, mounted on, or swapped on. When the drive is accepted into the set, it is repartitioned and the metadevice state database replica (for the set) can be placed on it. However, if a slice 7 (or slice 6 on an EFI labelled device), starts at cylinder 0, is large enough to hold a state database replica, and the slice does not overlap with other slices (including slice 2), then the disk is not repartitioned. Also, a drive is not accepted if it cannot be found on all hosts specified as part of the set. This means that if a host within the specified set is unreachable due to network problems, or is administratively down, the add fails.

Specify a drive name in the form c*num*t*num*d*num*. Do not specify a slice number (s*num*). For drives in a Sun Cluster, you must specify a complete pathname for each drive. Such a name has the form:

`/dev/did/[r]dsk/d`*num*

`-a` | `-d` | `-m` *mediator_host_list*
Add (`-a`) or delete (`-d`) mediator hosts to the specified disk set. A *mediator_host_list* is the nodename (established by `svc:/system/identity:node` `smf(5)` service) of the mediator host to be added and (for adding) up to two other aliases for the mediator host. The nodename and aliases for each mediator host are separated only by commas. Up to three mediator hosts can be specified for the named disk set. Specify only the nodename of that host as the argument to `-m` to delete a mediator host.

In a single `metaset` command you can add or delete up to three mediator hosts. See `EXAMPLES`.

`-A {enable | disable}`
Specify auto-take status for a disk set. If auto-take is enabled for a set, the disk set is automatically taken at boot, and file systems on volumes within the disk set can be mounted through `/etc/vfstab` entries. Only a single host can be associated with an auto-take set, so attempts to add a second host to an auto-take set or attempts to configure a disk set with multiple hosts as auto-take fails with an error message. Disabling auto-take status for a specific disk set causes the disk set to revert to normal behavior. That is, the disk set is potentially shared (non-concurrently) among hosts, and unavailable for mounting through `/etc/vfstab`.

`-b`
Insure that the replicas are distributed according to the replica layout algorithm. This can be invoked at any time, and does nothing if the replicas are correctly distributed. In cases where the user has used the `metadb` command to manually remove or add replicas, this command can be used to insure that the distribution of replicas matches the replica layout algorithm.

`-C {take | release | purge}`
Do not interact with the Cluster Framework when used in a Sun Cluster 3 environment. In effect, this means do not modify the Cluster Configuration Repository. These options should only be used to fix a broken disk set configuration.

take
Take ownership of the disk set but do not inform the Cluster Framework that the disk set is available. This option is not for use with a multi-owner disk set.

release
Release ownership of the disk set without informing the Cluster Framework. This option should only be used if the disk set ownership was taken with the corresponding `-C take` option. This option is not for use with a multi-owner disk set.

purge
Remove the disk set without informing the Cluster Framework that the disk set has been purged. This option should only be used when the disk set is not accessible and requires rebuilding.

`-d` *drivename*
Delete drives or hosts from the named disk set. For a drive to be deleted, it must not be in use within the set. The last host cannot be deleted unless all of the drives within the set are deleted. Deleting the last host in a disk set destroys the disk set.

Specify a drive name in the form c*num*t*num*d*num*. Do not specify a slice number (s*num*). For drives in a Sun Cluster, you must specify a complete pathname for each drive. Such a name has the form:

```
/dev/did/[r]dsk/dnum
```

This option fails on a multi-owner disk set if attempting to withdraw the master node while other nodes are in the set.

`-f`
Force one of three actions to occur: takes ownership of a disk set when used with `-t`; deletes the last disk drive from the disk set; or deletes the last host from the disk set. Deleting the last drive or host from a disk set requires the `-d` option.

When used to forcibly take ownership of the disk set, this causes the disk set to be grabbed whether or not another host owns the set. All of the disks within the set are taken over (reserved) and fail fast is enabled, causing the other host to panic if it had disk set ownership. The metadevice state database is read in by the host performing the take, and the shared metadevices contained in the set are accessible.

You can use this option to delete the last drive in the disk set, because this drive would implicitly contain the last state database replica.

You can use `-f` option to delete hosts from a set. When specified with a partial list of hosts, it can be used for one-host administration. One-host administration could be useful when a host is known to be non-functional, thus avoiding timeouts and failed commands. When specified with a complete list of hosts, the set is completely deleted. It is generally specified with a complete list of hosts to clean up after one-host administration has been performed.

`-h` *hostname...*
Specify one or more host names to be added to or deleted from a disk set. Adding the first host creates the set. The last host cannot be deleted unless all of the drives within the set have been deleted. The host name is not accepted if all of the drives within the set cannot be found on the specified host. The host name is the nodename established by the `svc:/system/identity:node` `smf(5)` service.

`-j`
Join a host to the owner list for a multi-owner disk set. The concepts of take and release, used with traditional disk sets, do not apply to multi-owner sets, because multiple owners are allowed.

As a host boots and is brought online, it must go through three configuration levels to be able to use a multi-owner disk set:

1. It must be included in the cluster nodelist, which happens automatically in a cluster or single-node sitatuion.
2. It must be added to the multi-owner disk set with the `-a -h` options documented elsewhere in this man page
3. It must join the set. When the host is first added to the set, it is automatically joined.

On manual restarts, the administrator must manually issue

`metaset -s` *multinodesetname* `-j`

to join the host to the owner list. After the cluster reconfiguration, when the host reenters the cluster, the node is automatically joined to the set. The `metaset -j` command joins the host to all multi-owner sets that the host has been added to. In a single node situation, joining the node to the disk set starts any necessary resynchronizations.

`-L`
When adding a disk to a disk set, force the disk to be repartitioned using the standard Solaris Volume Manager algorithm. See `DESCRIPTION`.

`-l` *length*
Set the size (in blocks) for the metadevice state database replica. The length can only be set when adding a new drive; it cannot be changed on an existing drive. The default (and maximum) size is 8192 blocks, which should be appropriate for most configurations. Replica sizes of less than 128 blocks are not recommended.

`-M`
Specify that the disk set to be created or modified is a multi-owner disk set that supports multiple concurrent owners.

This option is required when creating a multi-owner disk set. Its use is optional on all other operations on a multi-owner disk set and has no effect. Existing disk sets cannot be converted to multi-owner sets.

`-o`
Return an exit status of `0` if the local host or the host specified with the `-h` option is the owner of the disk set.

`-P`
Purge the named disk set from the node on which the `metaset` command is run. The disk set must not be owned by the node that runs this command. If the node does own the disk set, the command fails.

If you need to delete a disk set but cannot take ownership of the set, use the `-P` option.

`-q`
Displays an enumerated list of tags pertaining to "tagged data" that can be encountered during a take of the ownership of a disk set.

This option is not for use with a multi-owner disk set.

`-r`
Release ownership of a disk set. All of the disks within the set are released. The metadevices set up within the set are no longer accessible.

This option is not for use with a multi-owner disk set.

`-s` *setname*
Specify the name of a disk set on which `metaset` works. If no *setname* is specified, all disk sets are returned.

`-t`
Take ownership of a disk set safely. If `metaset` finds that another host owns the set, this host is not be allowed to take ownership of the set. If the set is not owned by any other host, all the disks within the set are owned by the host on which `metaset` was executed. The metadevice state database is read in, and the shared metadevices contained in the set become accessible. The `-t` option takes a disk set that has stale databases. When the databases are stale, `metaset` exits with code 66, and prints a message. At that point, the only operations permitted are the addition and deletion of replicas. Once the addition or deletion of the replicas has been completed, the disk set should be released and retaken to gain full access to the data.

This option is not for use with a multi-owner disk set.

`-u` *tagnumber*
Once a tag has been selected, a subsequent take with `-u` *tagnumber* can be executed to select the data associated with the given *tagnumber*.

`w`
Withdraws a host from the owner list for a multi-owner disk set. The concepts of take and release, used with traditional disk sets, do not apply to multi-owner sets, because multiple owners are allowed.

Instead of releasing a set, a host can issue

```
metaset -s multinodesetname -w
```

to withdraw from the owner list. A host automatically withdraws on a reboot, but can be manually withdrawn if it should not be able to use the set, but should be able to rejoin at a later time. A host that withdrew due to a reboot can still appear joined from other hosts in the set until a reconfiguration cycle occurs.

`metaset -w` withdraws from ownership of all multi-owner sets of which the host is a member. This option fails if you attempt to withdraw the master node while other nodes are in the disk set owner list. This option cancels all resyncs running on the node. A cluster reconfiguration process that is removing a node from the cluster membership list effectively withdraws the host from the ownership list.

`-y`
Execute a subsequent take. If the take operation encounters "tagged data," the take operation exits with code 2. You can then run the `metaset` command with the `-q` option to see an enumerated list of tags.

Examples

EXAMPLE 1 Defining a Disk Set

This example defines a disk set.

```
# metaset -s relo-red -a -h red blue
```

The name of the disk set is `relo-red`. The names of the first and second hosts added to the set are `red` and `blue`, respectively. (The hostname is the nodename established by `svc:/system/identity:node` `smf(5)` service.) Adding the first host creates the disk set. A disk set can be created with just one host, with the second added later. The last host cannot be deleted until all of the drives within the set have been deleted.

EXAMPLE 2 Adding Drives to a Disk Set

This example adds drives to a disk set.

```
# metaset -s relo-red -a c2t0d0 c2t1d0 c2t2d0 c2t3d0 c2t4d0 c2t5d0
```

The name of the previously created disk set is `relo-red`. The names of the drives are `c2t0d0`, `c2t1d0`, `c2t2d0`, `c2t3d0`, `c2t4d0`, and `c2t5d0`. There is no slice identifier ("s*x*") at the end of the drive names.

EXAMPLE 3 Adding Multiple Mediator Hosts

The following command adds three mediator hosts to the specified disk set.

```
# metaset -s mydiskset -a -m myhost1,alias1 myhost2,alias2 myhost3,alias3
```

EXAMPLE 4 Purging a Disk Set from the Node

The following command purges the disk set `relo-red` from the node:

```
# metaset -s relo-red -P
```

EXAMPLE 5 Querying a Disk Set for Tagged Data

The following command queries the disk set `relo-red` for a list of the tagged data:

```
# metaset -s relo-red -q
```

This command produces the following results:

```
The following tag(s) were found:
 1 - vha-1000c - Fri Sep 20 17:20:08 2002
 2 - vha-1000c - Mon Sep 23 11:01:27 2002
```

EXAMPLE 6 Selecting a tag and taking a Disk set

The following command selects a tag and takes the disk set `relo-red`:

```
# metaset -s relo-red -t -u 2
```

EXAMPLE 7 Defining a Multi-Owner Disk Set

The following command defines a multi-owner disk set:

```
# metaset -s blue -M -a -h hahost1 hahost2
```

The name of the disk set is `blue`. The names of the first and second hosts added to the set are `hahost1` and `hahost2`, respectively. The hostname is the nodename established by `svc:/system/identity:node` `smf(5)` service. Adding the first host creates the multi-owner disk set. A disk set can be created with just one host, with additional hosts added later. The last host cannot be deleted until all of the drives within the set have been deleted.

Files

`/etc/lvm/md.tab`
: Contains list of metadevice configurations.

Exit Status

The following exit values are returned:

`0`
: Successful completion.

`>0`
: An error occurred.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	storage/svm
Interface Stability	Committed

See Also

`mdmonitord(1M)`, `metaclear(1M)`, `metadb(1M)`, `metadetach(1M)`, `metahs(1M)`, `metainit(1M)`, `metaoffline(1M)`, `metaonline(1M)`, `metaparam(1M)`, `metarecover(1M)`, `metarename(1M)`, `metareplace(1M)`, `metassist(1M)`, `metastat(1M)`, `metasync(1M)`, `metattach(1M)`, `md.tab(4)`, `md.cf(4)`, `mddb.cf(4)`, `md.tab(4)`, `attributes(5)`, `smf(5)`, `md(7D)`

Solaris Volume Manager Administration Guide

Notes Disk set administration, including the addition and deletion of hosts and drives, requires all hosts in the set to be accessible from the network.

Name metassist – automated volume creation utility to support Solaris Volume Manager

Synopsis

```
metassist -V

metassist -?

metassist create [-v n] [-c] -F config_file

metassist create [-v n] [-c | -d] -F request_file

metassist create [-v n] [-c | -d] [-f] [-n name]
     [-p datapaths] [-r redundancy]
     [-a available [,available,...]]
     [-u unavailable [,unavailable,...]] -s setname -S size

metassist create -?
```

Description The `metassist` command provides assistance, through automation, with common Solaris Volume Manager tasks.

SUBCOMMANDS The following subcommands are supported:

`create` The `create` subcommand creates one or more Solaris Volume Manager volumes. You can specify this request on the command line or in a file specified on the command line.

If you create a volume using the command line, you can specify the characteristics of the volume in terms of the desired quality of service it will provide - its size, the number of redundant copies of the data it contains, the number of data paths by which it is accessible, and whether faulty components are replaced automatically. The diskset in which the volume will reside and the volume's size must be specified on the command line in this form of the command.

If you create a volume using a request in a file, you can specify the characteristics of the volume in terms of the quality of service they provide, as on the command line. Alternatively, the file can specify the types and component parts of the volume, (for example, mirrors, stripes, concatenations, and their component slices). The file may also specify volumes partly in terms of their types and partly in terms of their component parts, and may specify the characteristics of more than one volume. All volumes specified in a file must reside in the same diskset, whose name must be specified in the file.

If you specify the `-c` or `-d` option on the command line, the command runs without creating an actual volume or volumes. Instead, it outputs either a a Bourne shell command script (`-c` option) or a volume configuration (`-d` option). The command script, when run, creates the specified volume or volumes. The volume configuration specifies the volume or volumes in complete detail, naming all their components.

The input file given on the command line can take one of the following forms:

- a volume request, which specifies a request for a volume with explicit attributes and components, or matching a given quality of service
- a volume configuration, produced by a previous execution of the command

Options The following option is mandatory if you specify a volume request or volume configuration in a file:

`-F` *config_file* | *request_file* — Specify the volume request or volume configuration file to process. If *config_file* or *request_file* is `-`, it is read from standard input.

The `-d` option cannot be specified when *inputfile* is a volume configuration file.

The following options are mandatory if you specify a volume request on the command line:

`-s` *set* — Specify the disk set to use when creating volumes. All the volumes and hot spare pools are created in this disk set. If necessary, disks are moved into the diskset for use in the volumes and hot spare pools. If the diskset doesn't exist the command creates it. This option is required. `metassist` works entirely within a named disk set. Use of the local, or unnamed disk set, is not allowed.

`-S` *size* — Specify the size of the volume to be created. The size argument consists of a numeric value (a decimal can be specified) followed by KB, MB, GB, or TB, indicating kilobytes, megabytes, gigabytes, or terabytes, respectively. Case is ignored when interpreting this option. This option is required.

The following options are optional command line parameters:

`-a` *device1*`,`*device2*`,...` — Explicitly specify the devices that can be used in the creation of this volume. Named devices may be controllers or disks. Only used when specifying a volume on the command line.

`-c` — Output the command script that would implement the specified or generated volume configuration. The command script is not run, and processing stops at this stage.

`-d` — Output the volume configuration that satisfies the specified or generated volume request. No command script is generated or executed, and processing stops at this stage.

`-f` — Specify whether the volume should support automatic component replacement after a fault. If this option is specified, a mirror is created and its submirrors are associated with a hot spare.

`-n` *name* — Specify the name of the new volume. See `metainit(1M)` for naming guidelines.

`-p` *n*	Specify the number of required paths to the storage volume. The value of *n* cannot be greater than the number of different physical paths and logical paths to attached storage. Only used when specifying a volume on the command line.
`-r` *n*	Specify the redundancy level (`0`-`4`) of the data. The default is `0`. Only used when specifying a volume on the command line. If redundancy is `0`, a stripe is created. If redundancy is `1` or greater, a mirror with this number of submirrors is created. In this case, the volume can suffer a disk failure on *n*`-1` copies without data loss. With the use of hot spares (see the `-f` option), a volume can suffer a disk failure on *n*+`hsps-1` volumes without data loss, assuming non-concurrent failures.
`-u` *device1*`,`*device2*`,...`	Explicitly specify devices to exclude in the creation of this volume. Named devices can be controllers or disks. You can use this option alone, or to exclude some of the devices listed as available with the `-a` option, Only used when specifying a volume on the command line.
`-v` *value*	Specify the level of verbosity. Values from `0` to `2` are available, with higher numbers specifying more verbose output when the command is run. `-v 0` indicates silent output, except for errors or other critical messages. The default level is `1`.
`-V`	Display program version information.
`-?`	Display help information. This option can follow a subcommand for subcommand-specific help.

Examples

EXAMPLE 1 Creating a Mirror

The following example creates a two-way, 36Gb mirror on available devices from controller 1 and controller 2. It places the volume in diskset `mirrorset`.

```
# metassist create -r 2 -a c1,c2 -s mirrorset -S 36GB
```

EXAMPLE 2 Creating a Mirror with Additional Fault Tolerance

The following example creates a two-way, 36Gb mirror on available devices from controller 1 and controller 2. It provides additional fault tolerance in the form of a hot spare. It places the volume in diskset `mirrorset`.

```
# metassist create -f -r 2 -a c1,c2 -s mirrorset -S 36GB
```

EXAMPLE 3 Creating a Three-way Mirror and Excluding Devices

The following example creates a three-way, 180Gb mirror from storage devices on controller 1 or controller 2. It excludes the disks `c1t2d0` and `c2t2d1` from the volume. It places the volume in diskset `mirrorset`.

```
metassist create -r 3 -a c1,c2 -u c1t2d0, c2t2d1 \
    -s mirrorset -S 180GB
```

EXAMPLE 4 Determining and Implementing a Configuration

The following example determines and implements a configuration satisfying the request specified in a request file:

```
# metassist create -F request.xml
```

EXAMPLE 5 Determining a Configuration and Saving It in a volume-config File

The following example determines a configuration which satisfies the given request. It saves the configuration in a volume-config file without implementing it:

```
# metassist create -d -F request.xml > volume-config
```

EXAMPLE 6 Determining a Configuration and Saving It in a Shell Script

The following example determines a configuration which satisfies the given request. It saves the configuration in a shell script without implementing it:

```
# metassist create -c -F request.xml > setupvols.sh
```

EXAMPLE 7 Implementing the Given volume-config

The following example implements the given volume-config:

```
# metassist create -F config.xml
```

EXAMPLE 8 Converting the Given volume-config to a Shell Script

The following example converts the given volume-config to a shell script that you can run later:

```
# metassist create -c -F config.xml > setupvols.sh
```

Exit Status

The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

Files

- `/usr/share/lib/xml/dtd/volume-request.dtd`
- `/usr/share/lib/xml/dtd/volume-defaults.dtd`
- `/usr/share/lib/xml/dtd/volume-config.dtd`

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	storage/svm
Interface Stability	Committed

See Also mdmonitord(1M), metaclear(1M), metadb(1M), metadetach(1M), metahs(1M), metainit(1M), metaoffline(1M), metaonline(1M), metaparam(1M), metarecover(1M), metarename(1M), metareplace(1M), metaset(1M), metastat(1M), metasync(1M), metattach(1M), md.tab(4), md.cf(4), mddb.cf(4), md.tab(4), volume-config(4), volume-request(4), attributes(5), md(7D)

Notes The quality of service arguments are mutually exclusive with the `-F` *inputfile* argument.

When specifying a request file or quality of service arguments on the command line, the `/etc/default/metassist.xml` file is read for global and per-disk set defaults.

Characteristics of this file are specified in the DTD, in `/usr/share/lib/xml/dtd/volume-defaults.dtd`.

Characteristics of the XML request file are specified in the DTD, in `/usr/share/lib/xml/dtd/volume-request.dtd`.

Characteristics of the XML configuration file are specified in the DTD, in `/usr/share/lib/xml/dtd/volume-config.dtd`.

This command must be run as root.

This command requires a functional Solaris Volume Manager configuration before it runs.

Name metastat – display status for metadevice or hot spare pool

Synopsis

```
/usr/sbin/metastat -h

/usr/sbin/metastat [-a] [-B] [-D] [-c] [-i] [-p] [-q]
     [-s setname] [-t] [metadevice...] [hot_spare_pool...]

/usr/sbin/metastat [-a] [-B] [-D] [-c] [-i] [-p] [-q]
     [-s setname] component...
```

Description The `metastat` command displays the current status for each metadevice (including stripes, concatenations, concatenations of stripes, mirrors, RAID5, soft partitions, and trans devices) or hot spare pool, or of specified metadevices, components, or hot spare pools.

It is helpful to run the `metastat` command after using the `metattach` command to view the status of the metadevice.

`metastat` displays the state of each Solaris Volume Manager RAID-1 volume on the system. The possible states include:

`Okay` — The device reports no errors.

`Needs maintenance` — A problem has been detected. This requires that the system administrator replace the failed physical device. Volumes displaying `Needs maintenance` have incurred no data loss, although additional failures could risk data loss. Take action as quickly as possible.

`Last erred` — A problem has been detected. Data loss is a possibility. This might occur if a component of a submirror fails and is not replaced by a hot spare, therefore going into `Needs maintenance` state. If the corresponding component also fails, it would go into `Last erred` state and, as there is no remaining valid data source, data loss could be a possibility.

`Unavailable` — A device cannot be accessed, but has not incurred errors. This might occur if a physical device has been removed with Solaris Dynamic Reconfiguration (DR) features, thus leaving the Solaris Volume Manager volume unavailable. It could also occur if an array or disk is powered off at system initialization, or if a >1TB volume is present when the system is booted in 32-bit mode.

After the storage has been made available, run the `metastat` command with the `-i` option to update the status of the metadevices. This clears the unavailable state for accessible devices.

See the *Solaris Volume Manager Administration Guide* for instructions on replacing disks and handling volumes in Needs maintenance or Last erred states.

Options The following options are supported:

`-a` Display all disk sets. Only metadevices in disk sets that are owned by the current host are displayed.

`-B` Display the current status of all of the 64-bit metadevices and hot spares.

`-c` Display concise output.

There is one line of output for each metadevice. The output shows the basic structure and the error status, if any, for each metadevice.

The `-c` output format is distinct from the `-p` output format. The `-p` option does not display metadevice status and is not intended as human-readable output.

`-D` Display the current status of all of the descriptive name metadevices and hotspares.

`-h` Display usage message.

`-i` Check the status of RAID-1 (mirror) volumes, RAID-5 volumes, and hot spares. The inquiry checks each metadevice for accessibility, starting at the top level metadevice. When problems are discovered, the metadevice state databases are updated as if an error had occurred.

`-p` Display the list of active metadevices and hot spare pools in the same format as `md.tab`. See `md.tab(4)`.

The `-p` output is designed for snapshotting the configuration for later recovery or setup.

`-q` Display the status for metadevices without the device relocation information.

`-s` *setname* Specify the name of the disk set on which `metastat` works. Using the `-s` option causes the command to perform its administrative function within the specified disk set. Without this option, the command performs its function on metadevices and hot spare pools in the local disk set.

`-t` Display the current status and timestamp for the specified metadevices and hot spare pools. The timestamp provides the date and time of the last state change.

Operands The following operands are supported:

component Display the status of the component hosting a soft partition, including extents, starting blocks, and block count.

hot_spare_pool Display the status of the specified hot spare pool(s).

metadevice Display the status of the specified metadevice(s). If a trans metadevice is specified, the status of the master and log devices is also displayed. Trans metadevices have been replaced by UFS logging. See NOTES.

Examples

EXAMPLE 1 Output Showing Mirror with Two Submirrors

The following example shows the partial output of the `metastat` command after creating a mirror, `opt_mirror`, consisting of two submirrors, `opt_sub1` and `opt_sub2`.

```
# metastat opt_mirror
opt_mirror: Mirror
    Submirror 0: opt_sub1
      State: Okay
    Submirror 1: opt_sub2
      State: Resyncing
    Resync in progress: 15 % done
    Pass: 1
    Read option: roundrobin (default)
    Write option: parallel (default)
    Size: 2006130 blocks
    .
    .
    .
```

EXAMPLE 2 Soft Partition on Mirror with Submirror

The following example shows the partial output of the `metastat` command after creating a soft partition, `d3`, on concat `d2`, which is built on a soft partition.

```
# metastat
d2: Concat/Stripe
    Size: 204800 blocks
    Stripe 0:
        Device              Start Block  Dbase State        Hot Spare
        d0                         0     No    Okay

d0: Soft Partition
    Component: c0t3d0s0
    Status: Okay
    Size: 204800 blocks
        Extent              Start Block  Block count
             0                     129       204800

d3: Soft Partition
    Component: d2
    Status: Okay
    Size: 202752 blocks
         Extent              Start Block  Block count
              0                     129       202752
```

EXAMPLE 3 Trans Metadevice

The following example shows the output of the `metastat` command after creating a trans metadevice.

```
# metastat
d2: Concat/Stripe
    Size: 204800 blocks
    Stripe 0:
        Device              Start Block  Dbase State        Hot Spare
        d0                         0     No    Okay

d0: Soft Partition
    Component: c0t3d0s0
    Status: Okay
    Size: 204800 blocks
        Extent              Start Block  Block count
             0                     129       204800

d3: Soft Partition
    Component: d2
    Status: Okay
    Size: 202752 blocks
         Extent              Start Block  Block count
              0                     129       202752
```

EXAMPLE 4 Multi-owner disk set

The following example shows the output of the `metastat` command with a multi-owner disk set and application-based mirror resynchronization option. Application-based resynchronization is set automatically if needed.

```
# metastat -s oban
oban/d100: Mirror
    Submirror 0: oban/d10
      State: Okay
    Submirror 1: oban/d11
      State: Okay
    Pass: 1
    Read option: roundrobin (default)
    Write option: parallel (default)
    Resync option: application based
    Owner: None
    Size: 1027216 blocks (501 MB)

oban/d10: Submirror of oban/d100
    State: Okay
    Size: 1027216 blocks (501 MB)
    Stripe 0:
```

EXAMPLE 4 Multi-owner disk set *(Continued)*

```
        Device     Start Block  Dbase     State Reloc Hot Spare
        c1t3d0s0          0     No         Okay

oban/d11: Submirror of oban/d100
    State: Okay
    Size: 1027216 blocks (501 MB)
    Stripe 0:
        Device     Start Block  Dbase     State Reloc Hot Spare
        c1t4d0s0          0     No         Okay
```

Warnings `metastat` displays states as of the time the command is entered. It is unwise to use the output of the `metastat -p` command to create a `md.tab(4)` file for a number of reasons:

- The output of `metastat -p` might show hot spares being used.
- It might show mirrors with multiple submirrors. See `metainit(1M)` for instructions for creating multi-way mirrors using `metainit` and `metattach`.
- A slice may go into an error state after `metastat -p` is issued.

Exit Status The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	storage/svm
Interface Stability	Committed

See Also `mdmonitord(1M)`, `metaclear(1M)`, `metadb(1M)`, `metadetach(1M)`, `metahs(1M)`, `metainit(1M)`, `metaoffline(1M)`, `metaonline(1M)`, `metaparam(1M)`, `metarecover(1M)`, `metarename(1M)`, `metareplace(1M)`, `metaset(1M)`, `metassist(1M)`, `metasync(1M)`, `metattach(1M)`, `md.tab(4)`, `md.cf(4)`, `mddb.cf(4)`, `md.tab(4)`, `attributes(5)`, `md(7D)`

Solaris Volume Manager Administration Guide

Notes Trans metadevices have been replaced by UFS logging. Existing trans devices are *not* logging--they pass data directly through to the underlying device. See `mount_ufs(1M)` for more information about UFS logging.

Name metasync – handle metadevice resync during reboot

Synopsis

```
/usr/sbin/metasync -h

/usr/sbin/metasync [-s setname] [buffer_size] metadevice

/usr/sbin/metasync [-s setname] -r [buffer_size]

/usr/sbin/metasync -p metadevice

/usr/sbin/metasync -c metadevice
```

Description The `metasync` command starts a resync operation on the specified *metadevice*. All components that need to be resynced are resynced. If the system crashes during a RAID5 initialization, or during a RAID5 resync, either an initialization or resync restarts when the system reboots.

Applications are free to access a metadevice at the same time that it is being resynced by `metasync`. Also, `metasync` performs the copy operations from inside the kernel, which makes the utility more efficient.

Use the `-r` option in boot scripts to resync all possible submirrors.

Options The following options are supported:

`-c` *metadevice* — Cancels the resync that is in progress on the specified metadevice. The resync will be stopped at its current point and can be resumed by running the "`metasync metadevice`" command. This option only applies to RAID1 volumes.

`-h` — Displays usage message.

`-p` *metadevice* — Regenerates parity information for RAID5 metadevices.

`-s` *setname* — Specifies the name of the diskset on which `metasync` will work. Using the `-s` option will cause the command to perform its administrative function within the specified diskset. Without this option, the command will perform its function on local metadevices.

`-r` — Specifies that the `metasync` command handle special resync requirements during a system reboot. `metasync -r` should only be invoked from the `svc:/system/mdmonitor` service. The `metasync` command only resyncs those metadevices that need to be resynced. `metasync` schedules all the mirror resyncs according to their pass numbers.

To override the default buffer_size value used by the `svc:/system/mdmonitor` service, you can edit `/etc/system` to specify:

```
set md_mirror:md_resync_bufsz = 2048
```

so that resyncs occur as quickly as possible.

Operands

buffer_size Specifies the size (number of 512-byte disk blocks) of the internal copy buffer for the mirror resync. The size defaults to 1024 512-byte disk blocks. It can be no more than 2048 blocks. For best performance (quickest completion of the resync), 2048 blocks is the recommended size.

Exit Status

The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	storage/svm

See Also

`mdmonitord(1M)`, `metaclear(1M)`, `metadb(1M)`, `metadetach(1M)`, `metahs(1M)`, `metainit(1M)`, `metaoffline(1M)`, `metaonline(1M)`, `metaparam(1M)`, `metarecover(1M)`, `metarename(1M)`, `metareplace(1M)`, `metaset(1M)`, `metassist(1M)`, `metastat(1M)`, `metattach(1M)`, `md.tab(4)`, `md.cf(4)`, `mddb.cf(4)`, `md.tab(4)`, `attributes(5)`, `md(7D)`

Solaris Volume Manager Administration Guide

Notes

The `metasync` service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/system/mdmonitor
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. The service's status can be queried using the `svcs(1)` command.

Name metattach, metadetach – attach or detach a metadevice

Synopsis

```
/usr/sbin/metattach [-h]

/usr/sbin/metattach [-s setname] mirror [metadevice]

/usr/sbin/metattach [-s setname] [-i interlace] concat/stripe component...

/usr/sbin/metattach [-s setname] RAID component...

/usr/sbin/metattach [-s setname] [-A alignment] softpart size | all

/usr/sbin/metadetach [-s setname] [-f] mirror submirror

/usr/sbin/metadetach [-s setname] [-f] trans
```

Description `metattach` adds submirrors to a mirror, grows metadevices, or grows soft partitions. Growing metadevices can be done without interrupting service. To grow the size of a mirror or trans, the slices must be added to the submirrors or to the master devices.

Solaris Volume Manager supports storage devices and logical volumes greater than 1 terabyte (TB) when a system runs a 64-bit Solaris kernel. Support for large volumes is automatic. If a device greater than 1 TB is created, Solaris Volume Manager configures it appropriately and without user intervention.

If a system with large volumes is rebooted under a 32–bit Solaris kernel, the large volumes are visible through `metastat` output. Large volumes cannot be accessed, modified or deleted, and no new large volumes can be created. Any volumes or file systems on a large volume in this situation are also unavailable. If a system with large volumes is rebooted under a version of Solaris prior to the Solaris 9 4/03 release, Solaris Volume Manager does not start. You must remove all large volumes before Solaris Volume Manager runs under an earlier version of the Solaris Operating System.

Solaris Volume Manager supports one-to-four-way mirrors. You can only attach a metadevice to a mirror if there are three or fewer submirrors beneath the mirror. Once a new metadevice is attached to a mirror, `metattach` automatically starts a resync operation to the new submirror.

`metadetach` detaches submirrors from mirrors and logging devices from trans metadevices.

When a submirror is detached from a mirror, it is no longer part of the mirror, thus reads and writes to and from that metadevice by way of the mirror are no longer performed through the mirror. Detaching the only existing submirror is not allowed. Detaching a submirror that has slices reported as needing maintenance (by `metastat`) is not allowed unless the `-f` (force) flag is used.

`metadetach` also detaches the logging device from a trans. This step is necessary before you can clear the trans volume. Trans metadevices have been replaced by UFS logging. Existing trans devices are not logging. They pass data directly through to the underlying device. See `mount_ufs(1M)` for more information about UFS logging.

Detaching the logging device from a busy trans device is not allowed unless the `-f` (force) flag is used. Even so, the logging device is not actually detached until the trans is idle. The trans is in the *Detaching* state (`metastat`) until the logging device is detached.

Options Root privileges are required for all of the following options except `-h`.

The following options are supported:

`-A` *alignment* Set the value of the soft partition extent alignment. Use this option when it is important specify a starting offset for the soft partition. It preserves the data alignment between the metadevice address space and the address space of the underlying physical device.

For example, a hardware device that does checksumming should not have its I/O requests divided by Solaris Volume Manager. In this case, use a value from the hardware configuration as the value for the alignment. When using this option in conjunction with a software I/O load, the alignment value corresponds to the I/O load of the application. This prevents I/O from being divided unnecessarily and affecting performance.

`-f` Force the detaching of metadevices that have components that need maintenance or are busy. You can use this option only when a mirror is in a maintenance state that can be fixed with `metareplace(1M)`. If the mirror is in a maintenance state that can only be fixed with `metasync(1M)` (as shown by the output of `metastat(1M)`), `metadetach -f` has no effect, because the mirrors must be resynchronized before one of them can be detached.

`-h` Display a usage message.

`-i` *interlace* Specify the interlace value for stripes, where `size` is a specified value followed by either `k` for kilobytes, `m` for megabytes, or `b` for blocks. The units can be either uppercase or lowercase. If *size* is not specified, the size defaults to the interlace size of the last stripe of the metadevice. When an interlace size change is made on a stripe, it is carried forward on all stripes that follow.

`-s` *setname* Specify the name of the diskset on which the `metattach` command or the `metadetach` command works.. Using the `-s` option causes the command to perform its administrative function within the specified diskset. Without this option, the command performs its function on local metadevices.

Operands The following operands are supported:

component The logical name for the physical slice (partition) on a disk drive, such as `/dev/dsk/c0t0d0s2`, being added to the concatenation, stripe, concatenation of stripes, or RAID5 metadevice.

concat/stripe The metadevice name of the concatenation, stripe, or concatenation of stripes.

log	The metadevice name of the logging device to be attached to the trans metadevice.
metadevice	The metadevice name to be attached to the mirror as a submirror. This metadevice must have been previously created by the `metainit` command.
mirror	The name of the mirror.
RAID	The metadevice name of the RAID5 metadevice.
size | *all*	The amount of space to add to the soft partition in `K` or `k` for kilobytes, `M` or `m` for megabytes, `G` or `g` for gigabytes, `T` or `t` for terabytes, and `B` or `b` for blocks (sectors). All values represent powers of 2, and upper and lower case options are equivalent. Only integer values are permitted. The literal `all` specifies that the soft partition should grow to occupy all available space on the underlying volume.
softpart	The metadevice name of the existing soft partition.
submirror	The metadevice name of the submirror to be detached from the mirror.
trans	The metadevice name of the trans metadevice (not the master or logging device).

Examples

EXAMPLE 1 Concatenating a New Slice to a Metadevice

This example concatenates a single new slice to an existing metadevice, `Volume.1`. Afterwards, you would use the `growfs(1M)` command to expand the file system.

```
# metattach Volume.1 /dev/dsk/c0t1d0s2
```

EXAMPLE 2 Detaching Logging Device from Trans Metadevice

This example detaches the logging device from a trans metadevice `d9`. Notice that you do not have to specify the logging device itself, as there can only be one.

```
# metadetach d9
```

EXAMPLE 3 Expanding a RAID5 Metadevice

This example expands a RAID5 metadevice, `d45`, by attaching another slice.

```
# metattach d45 /dev/dsk/c3t0d0s2
```

When you add additional slices to a RAID5 metadevice, the additional space is devoted to data. No new parity blocks are allocated. The data on the added slices is, however, included in the overall parity calculations, so it is protected against single-device failure.

EXAMPLE 4 Expanding a Soft Partition

The following example expands a soft partition, `d42`, attaching all space available on the underlying device.

EXAMPLE 4 Expanding a Soft Partition *(Continued)*

```
# metattach d42 all
```

When you add additional space to a soft partition, the additional space is taken from any available space on the slice and might not be contiguous with the existing soft partition.

EXAMPLE 5 Adding Space to Two-Way Mirror

This example adds space to a two-way mirror by adding a slice to each submirror. Afterwards, you would use the `growfs(1M)` command to expand the file system.

```
# metattach d9 /dev/dsk/c0t2d0s5
# metattach d10 /dev/dsk/c0t3d0s5
```

This example tells the mirror to grow to the size of the underlying devices

```
# metattach d11
```

This example increases the size of the UFS on the device so the space can be used.

```
# growfs -M /export /dev/md/rdsk/d11
```

EXAMPLE 6 Detaching a Submirror from a Mirror

This example detaches a submirror, `d2`, from a mirror, `d4`.

```
# metadetach d4 d2
```

EXAMPLE 7 Adding Four Slices to Metadevice

This example adds four slices to an existing metadevice, `d9`. Afterwards, you would use the `growfs(1M)` command to expand the file system.

```
# metattach d9 /dev/dsk/c0t1d0s2 /dev/dsk/c0t2d0s2 \
     /dev/dsk/c0t3d0s2 /dev/dsk/c0t4d0s2
```

EXAMPLE 8 Setting the Value of the Soft Partition Extent Alignment

This example shows how to set the alignment of the soft partition to 1mb when the soft partition is expanded.

```
# metattach -s red -A 2m d13 1m
```

Exit Status

The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	storage/svm
Interface Stability	Committed

See Also mdmonitord(1M), metaclear(1M), metadb(1M), metahs(1M), metainit(1M), metaoffline(1M), metaonline(1M), metaparam(1M), metarecover(1M), metarename(1M), metareplace(1M), metaset(1M), metassist(1M), metastat(1M), metasync(1M), md.tab(4), md.cf(4), mddb.cf(4), md.tab(4), attributes(5), md(7D)

Solaris Volume Manager Administration Guide

Warnings This section provides information regarding warnings for devices greater than 1 TB and for multi-way mirrors.

Devices and Volumes Greater Than 1 TB

Do not create large (>1 TB) volumes if you expect to run the Solaris Operating System with a 32-bit kernel or if you expect to use a version of the Solaris Operating System prior to Solaris 9 4/03.

Multi-Way Mirrors

When a submirror is detached from its mirror, the data on the metadevice might not be the same as the data that existed on the mirror prior to running `metadetach`. In particular, if the `-f` option was needed, the metadevice and mirror probably do not contain the same data.

Notes Trans metadevices have been replaced by UFS logging. Existing trans devices are *not* logging. They pass data directly through to the underlying device. See mount_ufs(1M) for more information about UFS logging.

Name mib2mof – generate MOF file(s) from input SNMP MIB file(s)

Synopsis
```
/usr/sadm/bin/mib2mof [-n] [-d directory] [-q] [-c] [-a]
     [-h] files
```

Description The `mib2mof` utility reads input Management Information Base (MIB) files and produces one or more Managed Object Format (MOF) files. MOF files contain a Common Information Model (CIM) class declaration that represents the MIB for the Solaris Simple Network Management Protocol (SNMP) provider. The SNMP provider allows Web-Based Enterprise Management (WBEM) applications to access SNMP device information.

SNMP scalar variables map to properties in the CIM class. Qualifiers on each property convey the following MIB information for each scalar variable:

- syntax
- read/write access
- OID (Object IDentifier)
- description (optional)
- index (if the variable is within a group [sequence] that defines a row)

The syntax of an SNMP scalar variable is represented in a CIM class by the property's CIM datatype. All properties are marked with write access (true or false).

The following table shows how a Solaris SNMP datatype in a MIB maps to a Web-Based Enterprise Management (WBEM) CIM datatype and then to an SNMP datatype used by the WBEM SNMP API:

SNMP SMI Datatype	SNMP Ver.	CIM Datatype	SNMP API Object type
INTEGER	v1	sint32	SnmpInt
OCTET STRING	v1	string	SnmpString
OBJECT IDENTIFIER	v1	string	SnmpOid
IpAddress	v1	string	SnmpIpAddress
Counter	v1	uint32	SnmpCounter
Gauge	v1	uint32	SnmpGauge
TimeTicks	v1	uint32	SnmpTimeticks
Opaque	v1	sint8[]	SnmpOpaque
DisplayString - see OCTET STRING	v1		
NetworkAddress - see IpAddress	v1		
Counter32 - see Counter	v2		
Counter64	v2	uint64	SnmpCounter64
Integer32	v2	sint32	SnmpInt
Gauge32 - see Gauge	v2		
Unsigned32	v2	uint32	SnmpGauge
TruthValue	v2	sint32	SnmpInt
BITS - see OCTET STRING	v2		

The `mib2mof` utility includes its required `Solaris_SNMPmib_core.txt` file (containing core `MIB` definitions), installed in `/usr/sadm/mof`. The `mib2mof` utility looks first for `mib` core file in local directory. If this file is not found in the local directory, `mib2mof` looks in `/usr/sadm/mof`.

A MOF file is generated for each SNMP group and table row sequence (that is, the columns in one row) found in the supplied MIBs. (This does not include the core MIB definitions contained in the `Solaris_SNMPmib_core.txt file`.)

There is no MOF file or property for an SNMP table - all table access is through the rows and columns of the table, and the SNMP variable for the table is marked as inaccessible in the MIB.

The MOF file created contains a CIM class that represents an SNMP group or row and a CIM class to represent a CIM association. The output file name (and CIM class) is of the format <SNMP_><MIB name><Group name>.mof.

Options The following options are supported:

`-a` Generate MOF files for all of the input MIB files. If `-a` is not given, a MOF file is generated only for the last file of the input list.

`-c` Do not use the default `Solaris_SNMPmib_core.txt` definitions file shipped with the Solaris SNMP Provider for WBEM. If this option is specified, you must specify another `MIB_CORE` definitions file as one of the input files.

`-d` *directory* Generate output MOF files in the specified directory.

`-h` Show how to invoke `mib2mof` and list its arguments.

`-n` Parse the input MIB files without generating any output.

`-q` Include the DESCRIPTION clause of `SNMP OBJECT-TYPE` as a qualifier in the generated MOF file.

Operands The following operands are supported:

files List of SNMP MIB files to be converted.

Exit Status The `mib2mof` utility terminates with exit status `0`.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWwbco

See Also `init.wbem(1M)`, `mofcomp(1M)`, `wbemadmin(1M)`, `attributes(5)`

Name mibiisa – Sun SNMP Agent

Synopsis

```
mibiisa [-ar] [-c config-dir] [-d debug-level] [-p port]
     [-t cache-timer]
```

Description The `mibiisa` utility is an RFC 1157-compliant SNMP agent. It supports MIB-II as defined in *RFC 1213*, with Sun extensions under Sun's enterprise number. The MIB (Management Information Base) is both readable and writable. The `mibiisa` utility supports all SNMP protocol operations including GET-REQUEST, GETNEXT-REQUEST, SET-REQUEST, GET-REPLY, and TRAP.

The SMA (Systems Management Agent) is the default SNMP agent in Solaris. MIB-II subagent `mibiisa` does not run by default. To enable `mibiisa`, rename the configuration file from `/etc/snmp/conf/mibiisa.rsrc-` to `/etc/snmp/conf/mibiisa.rsrc`. SMA has the capability to handle any MIB-II requests.

The `mibiisa` utility supports the coldStart, linkUp, linkDown, and authentication traps. The authentication trap may be disabled by a command-line switch, which itself may be overridden by a management station writing to a MIB variable in the standard SNMP MIB group.

The `mibiisa` utility supports four distinct views of the MIB. The view used for any request is determined by the community string contained in that request.

To enhance security, `mibiisa` supports an option to block all writes to the MIB. You can also limit the set of management stations from which the agent will accept requests in the configuration file used when starting the `mibiisa`. See the Security section for more information.

Unless overridden, `mibiisa` uses UDP port 161, the standard SNMP port. The `mibiisa` utility issues traps through the same port on which it receives SNMP requests.

The `mibiisa` utility must run with super-user privileges and is typically started at system startup via `/etc/rc3.d`. `mibiisa` may not be started using `inetd(1M)`. When started, `mibiisa` detaches itself from the keyboard, disables all signals except `SIGKILL`, `SIGILL`, `SIGUSR1`, and `SIGUSR2`, and places itself in the background.

Options The following options are supported by `mibiisa`:

`-a` Disable the generation of authentication traps. However, an SNMP manager may write a value into `snmpEnableAuthenTraps` to enable or disable authentication traps.

`-c` *config-dir* Specify a directory where it expects `snmpd.conf` file, on startup. The default directory is `/etc/snmp/conf`.

`-d` *debug-level* Debug. A value of 0 disables all debug and is the default. Levels 1 through 3 represent increasing levels of debug output. When `mibiisa` receives the

signal `SIGUSR1`, it resets the debug-level to 0. When `mibiisa` receives the signal `SIGUSR2`, it increments the debug-level by one.

Debug output is sent to the standard output in effect at the time `mibiisa` is started. No matter what debug level is in effect, certain significant events are logged in the system log.

`-p` *port* — Define an alternative UDP port on which `mibiisa` listens for incoming requests. The default is UDP port 161.

`-r` — Place the MIB into read-only mode.

`-t` *cache-timer* — By default, information fetched from the kernel is considered to be valid for 45 seconds from the time it is retrieved. This cache lifetime may be altered with this parameter. You cannot set *cache-timer* to any value less than 1.

Configuration File

The `snmpd.conf` file is used for configuration information. Each entry in the file consists of a keyword followed by a parameter string. The keyword must begin in the first position. Parameters are separated from the keyword and from one another by white space. Case in keywords is ignored. Each entry must be contained on a single line. All text following (and including) a pound sign (#) is ignored. Keywords currently supported are:

`sysdescr` — The value to be used to answer queries for sysDescr.

`syscontact` — The value to be used to answer queries for sysContact.

`syslocation` — The value to be used to answer queries for sysLocation.

`trap` — The parameter names one or more hosts to receive traps. Only five hosts may be listed.

`system-group-read-community` — The community name to get read access to the system group and Sun's extended system group.

`system-group-write-community` — The community name to get write access to the system group and Sun's extended system group.

`read-community` — The community name to get read access to the entire MIB.

`write-community` — The community name to get write access to the entire MIB (implies read access).

`trap-community` — The community name to be used in traps.

`kernel-file` — The name of the file to use for kernel symbols.

`managers` — The names of hosts that may send SNMP queries. Only five hosts may be listed on any one line. This keyword may be repeated for a total of 32 hosts.

`newdevice` — The additional devices which are not built in SNMPD. The format is as follows: *newdevice* `type` *speed name* where *newdevice* is the keyword, `type` is an integer which has to match your schema file, *speed* is the new device's speed, and *name* is this new device's name.

An example `snmpd.conf` file is shown below:

```
sysdescr        Sun SNMP Agent, Sun Fire 4800, Company
                   Property Number 123456
syscontact      Cliff Claven
sysLocation     Room 1515, building 1
#
system-group-read-community     public
system-group-write-community    private
#
read-community  all_public
write-community all_private
#
trap            localhost
trap-community  SNMP-trap
#
#kernel-file    /vmunix
#
managers        lvs golden
managers        swap
```

Installation The `mibiisa` utility and its configuration file, `snmpd.conf`, may be placed in any directory. However for Solaris 2.4 and subseqent releases, use `/usr/lib/snmp` for `mibiisa` itself and `/etc/snmp/conf` for the configuration file. You can modify the configuration file as appropriate. If you make any changes to `snmpd.conf` file keyword values, you must kill and restart `mibiisa` for the changes to take effect.

Your `/etc/services` file (or NIS equivalent) should contain the following entries:

```
snmp                161/udp                             # Simple Network Mgmt
                                                        Protocol

snmp-trap           162/udp             snmptrap        # SNMP trap (event)
                                                        messages
```

The following is an example for Solaris 2.*x* and releases compatible with Solaris 2.*x*, such as Solaris 9:

```
#
# Start the SNMP agent
#
```

```
if [ -f /etc/snmp/conf/snmpd.conf -a -x
   /usr/lib/snmp/mibiisa ];
then
/opt/SUNWconn/snm/agents/snmpd
echo 'Starting SNMP-agent.'
```

Security SNMP, as presently defined, offers relatively little security. The `mibiisa` utility accepts requests from other machines, which can have the effect of disabling the network capabilities of your computer. To limit the risk, the configuration file lets you specify a list of up to 32 manager stations from which `mibiisa` will accept requests. If you do not specify any such manager stations, `mibiisa` accepts requests from anywhere.

The `mibiisa` utility also allows you to mark the MIB as "read-only" by using the `-r` option.

`mibiisa` supports four different community strings. These strings, however, are visible in the configuration file and within the SNMP packets as they flow on the network.

The configuration file should be owned by, and readable only by super-user. In other words the mode should be:

```
-rw-------  1 root         2090 Oct 17 15:04 /etc/snmp/conf/snmpd.conf
```

Managers can be restricted based on the community strings. This can be configured by creating an optional secondary configuration file `/etc/snmp/conf/mibiisa.acl`. To enable such a restriction, add the security line in the `/etc/snmp/conf/mibiisa.rsrc` file.

An example `mibiisa.acl` file is as follows:

```
acl = {
        {
                communities = public
                access = read-only
                managers = xyz
        }
        {
                communities = private
                access = read-write
                managers = abc,pqrs
        }
}
```

An example `mibiisa.rsrc` file is as follows:

```
resource =
{
        {
                registration_file = "/etc/snmp/conf/mibiisa.reg"
                security = "/etc/snmp/conf/mibiisa.acl"
                policy = "spawn"
                type = "legacy"
```

```
                command = "/usr/lib/snmp/mibiisa -r -p $PORT"
        }
}
```

Mib This section discusses some of the differences between the `mibiisa` MIB and the standard MIB-II (as defined in RFC 1213).

The following variables are read-only in the `mibiisa` MIB:

```
sysName
atIfIndex
ipDefaultTTL
```

These variables are read-write in the standard MIB-II.

The `mibiisa` MIB Address Translation tables support limited write access: only atPhysAddress may be written, either to change the physical address of an existing entry or to delete an entire ARP table entry.

The `mibiisa` MIB IP Net to Media table supports limited write access: only ipNetToMediaPhysAddress and ipNetToMediaType may be written, either to change the physical address of an existing entry or to delete an entire ARP table entry.

The following variables are read-write in the `mibiisa` MIB; however, these variables have fixed values. Any new values "set" to them are accepted, but have no effect:

```
ipRoutIfIndex
ipRouteMetric1
ipRouteMetric2
ipRouteMetric3
ipRouteMetric4
ipRouteType
ipRouteAge
ipRouteMask
ipRouteMetric5
```

The following `mibiisa` MIB variable reflects the actual state of the related table entry. "Sets" are accepted but have no effect:

```
tcpConnState
```

The following `mibiisa` MIB variables are readable, but return a fixed value:

`icmpInDestUnreachs`	Returns 1
`icmpInTimeExcds`	Returns 1
`icmpInParmProbs`	Returns 1
`icmpInSrcQuenchs`	Returns 1

icmpInRedirects	Returns 1
icmpInEchos	Returns 1
icmpInEchoReps	Returns 1
icmpInTimestamps	Returns 1
icmpInTimestampReps	Returns 1
icmpInAddrMasks	Returns 1
icmpInAddrMaskReps	Returns 1
icmpOutDestUnreachs	Returns 1
icmpOutTimeExcds	Returns 1
icmpOutParmProbs	Returns 1
icmpOutSrcQuenchs	Returns 1
icmpOutRedirects	Returns 1
icmpOutEchos	Returns 1
icmpOutEchoReps	Returns 1
icmpOutTimestamps	Returns 1
icmpOutTimestampReps	Returns 1
icmpOutAddrMasks	Returns 1
icmpOutAddrMaskReps	Returns 1
ifInUnknownProtos	Returns 0
ipAdEntBcastAddr	Returns 1
ipAdEntReasmMaxSiz	Returns 65535
ipRouteMetric1	Returns −1
ipRouteMetric2	Returns −1
ipRouteMetric3	Returns −1
ipRouteMetric4	Returns −1
ipRouteAge	Returns 0
ipRouteMetric5	Returns −1
ipNetToMediaType	Returns (3) dynamic
ipRoutingDiscards	Returns 0

The following variables return a fixed value of `0` for drivers not conforming to the GLD framework (see `gld(7D)`), including the old LAN drivers on SPARC machines:

`ifInOctets`	Returns 0
`ifInNUcastPkts`	Returns 0
`ifInDiscards`	Returns 0
`ifOutOctets`	Returns 0
`ifOutNUcastPkts`	Returns 0
`ifOutDiscards`	Returns 0

Schema Attributes

The following describes the attributes in the group and table definitions in the `/var/snmp/mib/sun.mib` file.

system

The `system` group reports statistics about a particular system (for example, a workstation or a printer).

`sysDescr` – A textual description of the entity. This value should include the full name and version identification of the system's hardware type, software operating-system, and networking software. This value must only contain printable ASCII characters. (string[255])

`sysObjectID` – The vendor's authoritative identification of the network management subsystem contained in the entity. This value is allocated within the SMI enterprises subtree (1.3.6.1.4.1) and provides an easy and unambiguous means for determining what type of equipment is being managed. For example, if vendor “Flintstones, Inc.” was assigned the subtree 1.3.6.1.4.1.4242, it could assign the identifier 1.3.6.1.4.1.4242.1.1 to its “Fred Router.” (objectid)

`sysUpTime` – Time (in hundredths of a second) since the network management portion of the system was last reinitialized. (timeticks)

`sysContact` – The textual identification of the contact person for this managed node, together with information on how to contact this person. (string[255])

`sysName` – An administratively-assigned name for this managed node. By convention, this is the node's fully-qualified domain name. (string[255])

`sysLocation` – The physical location of this node (for example, “telephone closet, 3rd floor” (string[255]))

`sysServices` – A value indicating the set of services that this entity primarily offers. (int) The value is a sum. This sum initially takes the value zero. Then, for each layer L in the range 1 through 7 for which this node performs transactions, 2 raised to (L - 1) is added to the sum. For example, a node that performs primarily routing functions would have a value of 4

(2**(3-1)). In contrast, a node that is a host offering application services would have a value of 72 (2**(4-1) + 2**(7-1)). Note that in the context of the Internet suite of protocols, values should be calculated accordingly:

Layer	Functionality
1	physical (such as repeaters)
2	datalink/subnetwork (such as bridges)
3	internet (such as IP gateways)
4	end-to-end (such as IP hosts)
7	applications (such as mail relays)

For systems including OSI protocols, Layers 5 and 6 may also be counted.

interfaces The `interfaces` group reports the number of interfaces handled by the agent.

`ifNumber` – The number of network interfaces, regardless of their current state, present on this system. (int)

ifTable The `ifTable` is a table of interface entries. The number of entries is given by the value of ifNumber.

`ifIndex` – A unique value for each interface. Its value ranges between 1 and the value of ifNumber. The value for each interface must remain constant at least from one reinitialization of the entity's network management system to the next reinitialization. (int)

`ifDescr` – A textual string containing information about the interface. This string should include the name of the manufacturer, the product name, and the version of the hardware interface. (string[255])

`ifType` – The type of interface, distinguished according to the physical/link protocol(s) immediately below the network layer in the protocol stack. (enum)

`ifMtu` – The size of the largest datagram that can be sent/received on the interface, specified in octets. For interfaces used for transmitting network datagrams, this is the size of the largest network datagram that can be sent on the interface. (int)

`ifSpeed` – An estimate of the interface's current bandwidth in bits-per-second. For interfaces that do not vary in bandwidth, or for those where no accurate estimation can be made, this object should contain the nominal bandwidth. (gauge)

`if1hysAddress` – The interface's address at the protocol layer immediately below the network layer in the protocol stack. For interfaces without such an address (for example, a serial line), this object should contain an octet string of zero length. (octet[128])

`ifAdminStatus` – The desired state of the interface. The testing(3) state indicates that no operational packets can be passed. (enum)

`if OperStatus` – The current operational state of the interface. The testing(3) state indicates that no operational packets can be passed. (enum)

`ifLastChange` – The value of sysUpTime at the time the interface entered its current operational state. If the current state was entered prior to the last reinitialization of the local network management subsystem, then this object contains a zero value. (timeticks)

`ifInOctets` – The total number of octets received on the interface, including framing characters. (counter) Returns a fixed value of 0.

`ifInUcastPkts` – The number of subnetwork-unicast packets delivered to a higher-layer protocol. (counter)

`ifInNUcastPkts` – The number of non-unicast (that is, subnetwork- broadcast or subnetwork-multicast) packets delivered to a higher-layer protocol. (counter) Returns a fixed value of 0.

`ifInDiscards` – The number of inbound packets chosen to be discarded, even though no errors had been detected to prevent their being deliverable to a higher-layer protocol. One possible reason for discarding such a packet could be to free up buffer space. (counter) Returns a fixed value of 0.

`ifInErrors` – The number of inbound packets that contained errors preventing them from being deliverable to a higher-layer protocol. (counter)

`ifInUnknownProtos` – The number of packets received via the interface that were discarded because of an unknown or unsupported protocol. (counter) Returns a fixed value of 0.

`ifOutOctets` – The total number of octets transmitted out of the interface, including framing characters. (counter) Returns a fixed value of 0.

`ifOutUcastPkts` – The total number of packets that higher-level protocols requested be transmitted to a subnetwork-unicast address, including those that were discarded or not sent. (counter)

`ifOutNUcastPkts` – The total number of packets that higher-level protocols requested be transmitted to a non- unicast (that is, a subnetwork-broadcast or subnetwork-multicast) address, including those that were discarded or not sent. (counter) Returns a fixed value of 0.

`ifOutDiscards` – The number of outbound packets that were chosen to be discarded even though no errors had been detected to prevent their being transmitted. One possible reason for discarding such a packet could be to free up buffer space. (counter) Returns a fixed value of 0.

`ifOutErrors` – The number of outbound packets that could not be transmitted because of errors. (counter)

`ifOutQLen` – The length of the output packet queue (in packets). (gauge)

`ifSpecific` – A reference to MIB definitions specific to the particular media being used to realize the interface. For example, if the interface is realized by an Ethernet, then the value of this object refers to a document defining objects specific to Ethernet. If this information is not present, its value should be set to the OBJECT IDENTIFIER { 0 0 }, which is a syntactically valid object identifier. Any conformant implementation of ASN.1 and BER must be able to generate and recognize this value. (objectid)

atTable `atTable` Address Translation tables contain the NetworkAddress to physical address equivalences. Some interfaces do not use translation tables for determining address equivalences (for example, DDN-X.25 has an algorithmic method). If all interfaces are of this type, then the Address Translation table is empty, that is, has zero entries.

`atIfIndex` – The interface on which this entry's equivalence is effective. The interface identified by a particular value of this index is the same interface as identified by the same value of ifIndex. (int)

`atPhysAddress` – The media-dependent physical address. (octet[128]) Setting this object to a null string (one of zero length) has the effect of invaliding the corresponding entry in the atTable object. That is, it effectively dissociates the interface identified with said entry from the mapping identified with said entry. It is an implementation-specific matter as to whether the agent removes an invalidated entry from the table. Accordingly, management stations must be prepared to receive tabular information from agents that corresponds to entries not currently in use. Proper interpretation of such entries requires examination of the relevant atPhysAddress object.

`atNetAddress` – The NetworkAddress (that is, the IP address) corresponding to the media-dependent physical address. (netaddress)

ip The `ip` group reports statistics about the Internet Protocol (IP) group.

`ipForwarding` – The indication of whether this entity is acting as an IP gateway in respect to the forwarding of datagrams received by, but not addressed to, this entity. IP gateways forward datagrams. IP hosts do not— except those source-routed via the host. (enum)

Note that for some managed nodes, this object may take on only a subset of the values possible. Accordingly, it is appropriate for an agent to return a “badValue” response if a management station attempts to change this object to an inappropriate value.

`ipDefaultTTL` – The default value inserted into the Time-To-Live field of the IP header of datagrams originated at this entity, whenever a TTL value is not supplied by the transport layer protocol. (int)

`ipInReceives` – The total number of input datagrams received from interfaces, including those received in error. (counter)

`ipInHdrErrors` – The number of input datagrams discarded due to errors in their IP headers, including bad checksums, version number mismatch, other format errors, time-to-live exceeded, errors discovered in processing their IP options, and so on. (counter)

`ipInAddrErrors` – The number of input datagrams discarded because the IP address in their IP header's destination field was not a valid address to be received at this entity. This count includes invalid addresses (for example, 0.0.0.0) and addresses of unsupported Classes (for example, Class E). For entities that are not IP Gateways and therefore do not forward datagrams, this counter includes datagrams discarded because the destination address was not a local address. (counter)

`ipForwDatagrams` – The number of input datagrams for which this entity was not their final IP destination, as a result of which an attempt was made to find a route to forward them to that final destination. In entities that do not act as IP Gateways, this counter will include only those packets that were Source-Routed via this entity, and the Source- Route option processing was successful. (counter)

`ipInUnknownProtos` – The number of locally-addressed datagrams received successfully but discarded because of an unknown or unsupported protocol. (counter)

`ipInDiscards` – The number of input IP datagrams for which no problems were encountered to prevent their continued processing, but which were discarded, for example, for lack of buffer space. Note that this counter does not include any datagrams discarded while awaiting reassembly. (counter)

`ipInDelivers` – The total number of input datagrams successfully delivered to IP user-protocols (including ICMP). (counter)

`ipOutRequests` – The total number of IP datagrams that local IP user-protocols (including ICMP) supplied to IP in requests for transmission. Note that this counter does not include any datagrams counted in ipForwDatagrams. (counter)

`ipOutDiscards` – The number of output IP datagrams for which no problem was encountered to prevent their transmission to their destination, but which were discarded (for example, for lack of buffer space). Note that this counter would include datagrams counted in ipForwDatagrams if any such packets met this (discretionary) discard criterion. (counter)

`ipOutNoRoutes` – The number of IP datagrams discarded because no route could be found to transmit them to their destination. Note that this counter includes any packets counted in ipForwDatagrams which meet this “no-route” criterion. Note that this includes any datagrams that a host cannot route because all its default gateways are down. (counter)

`ipReasmTimeout` – The maximum number of seconds that received fragments are held while they are awaiting reassembly at this entity. (int)

`ipReasmReqds` – The number of IP fragments received that needed to be reassembled at this entity. (counter)

`ipReasmOKs` – The number of IP datagrams successfully reassembled. (counter)

`ipReasmFails` – The number of failures detected by the IP reassembly algorithm, for whatever reason: timed out, errors, and the like. Note that this is not necessarily a count of discarded IP fragments since some algorithms (notably the algorithm in RFC 815) can lose track of the number of fragments by combining them as they are received. (counter)

`ipFragOKs` – The number of IP datagrams that have been successfully fragmented at this entity. (counter)

`ipFragFails` – The number of IP datagrams that have been discarded because they needed to be fragmented at this entity but could not be, for example, because their "Don't Fragment" flag was set. (counter)

`ipFragCreates` – The number of IP datagram fragments that have been generated as a result of fragmentation at this entity. (counter)

`ipRoutingDiscards` – The number of routing entries that were chosen to be discarded even though they were valid. One possible reason for discarding such an entry could be to free-up buffer space for other routing entries. (counter) Returns a fixed value of 0.

ipAddrTable `ipAddrTable` is a table of addressing information relevant to this entity's IP addresses.

`ipAdEntAddr` – The IP address to which this entry's addressing information pertains. (netaddress)

`ipAdEntIfIndex` – The index value that uniquely identifies the interface to which this entry is applicable. The interface identified by a particular value of this index is the same interface as identified by the same value of ifIndex. (int)

`ipAdEntNetMask` – The subnet mask associated with the IP address of this entry. The value of the mask is an IP address with all the network bits set to 1, and all the hosts bits set to 0. (netaddress)

`ipAdEntBcastAddr` – The value of the least-significant bit in the IP broadcast address used for sending datagrams on the (logical) interface associated with the IP address of this entry. For example, when the Internet standard all-ones broadcast address is used, the value will be 1. This value applies to both the subnet and network broadcasts addresses used by the entity on this (logical) interface. (int) Returns a fixed value of 1.

`ipAdEntReasmMaxSize` – The size of the largest IP datagram that this entity can reassemble from incoming IP fragmented datagrams received on this interface. (int) Returns a fixed value of 65535.

ipRouteTable The `ipRouteTable` is this entity's IP Routing table.

`ipRouteDest` – The destination IP address of this route. An entry with a value of 0.0.0.0 is considered a default route. Multiple routes to a single destination can appear in the table, but

access to such multiple entries is dependent on the table- access mechanisms defined by the network management protocol in use. (netaddress)

`ipRouteIfIndex` – The index value that uniquely identifies the local interface through which the next hop of this route should be reached. The interface identified by a particular value of this index is the same interface as identified by the same value of ifIndex. (int)

`ipRouteMetric1` – The primary routing metric for this route. The semantics of this metric are determined by the routing-protocol specified in the route's ipRouteProto value. If this metric is not used, its value should be set to –1. (int) Returns a fixed value of –1.

`ipRouteMetric2` – An alternate routing metric for this route. The semantics of this metric are determined by the routing-protocol specified in the route's ipRouteProto value. If this metric is not used, its value should be set to –1. (int) Returns a fixed value of –1.

`ipRouteMetric3` – An alternate routing metric for this route. The semantics of this metric are determined by the routing-protocol specified in the route's ipRouteProto value. If this metric is not used, its value should be set to –1. (int) Returns a fixed value of –1.

`ipRouteMetric4` – An alternate routing metric for this route. The semantics of this metric are determined by the routing-protocol specified in the route's ipRouteProto value. If this metric is not used, its value should be set to –1. (int) Returns a fixed value of –1.

`ipRouteNextHop` – The IP address of the next hop of this route. (In the case of a route bound to an interface that is realized via a broadcast media, the value of this field is the agent's IP address on that interface.) (netaddress)

`ipRouteType` – The type of route. Note that the values direct (3) and indirect (4) refer to the notion of direct and indirect routing in the IP architecture. (enum)

Setting this object to the value invalid (2) has the effect of invalidating the corresponding entry in the ipRouteTable object. That is, it effectively dissociates the destination identified with said entry from the route identified with said entry. It is an implementation-specific matter as to whether the agent removes an invalidated entry from the table. Accordingly, management stations must be prepared to receive tabular information from agents that corresponds to entries not currently in use. Proper interpretation of such entries requires examination of the relevant ipRouteType object.

`ipRouteProto` – The routing mechanism through which this route was learned. Inclusion of values for gateway routing protocols is not intended to imply that hosts should support those protocols. (enum)

`ipRouteAge` – The number of seconds since this route was last updated or otherwise determined to be correct. Note that no semantics of "too old" can be implied except through knowledge of the routing protocol by which the route was learned. (int) Returns a fixed value of 0.

`ipRouteMask` – Indicate the mask to be logical-ANDed with the destination address before being compared to the value in the ipRouteDest field. For those systems that do not support arbitrary subnet masks, an agent constructs the value of the ipRouteMask by determining whether the value of the correspondent ipRouteDest field belongs to a class-A, B, or C network, and then using one of:

Mask	Network
255.0.0.0	class-A
255.255.0.0	class-B
255.255.255.0	class-C

If the value of the ipRouteDest is 0.0.0.0 (a default route), then the mask value is also 0.0.0.0. It should be noted that all IP routing subsystems implicitly use this mechanism. (netaddress)

`ipRouteMetric5` – An alternate routing metric for this route. The semantics of this metric are determined by the routing-protocol specified in the route's ipRouteProto value. If this metric is not used, its value should be set to –1. (int) Returns a fixed value of –1.

`ipRouteInfo` – A reference to MIB definitions specific to the particular routing protocol responsible for this route, as determined by the value specified in the route's ipRouteProto value. If this information is not present, its value should be set to the OBJECT IDENTIFIER { 0 0 }, which is a syntactically valid object identifier. Any conformant implementation of ASN.1 and BER must be able to generate and recognize this value. (objectid)

ipNetToMediaTable

The `ipNetToMediaTable` is the IP Address Translation table used for mapping from IP addresses to physical addresses.

`ipNetToMediaIfIndex` – The interface on which this entry's equivalence is effective. The interface identified by a particular value of this index is the same interface as identified by the same value of ifIndex. (int)

`ipNetToMediaPhysAddress` – The media-dependent physical address. (octet[128])

`ipNetToMediaNetAddress` – The IpAddress corresponding to the media- dependent physical address. (netaddress)

`ipNetToMediaType` – The type of mapping. (enum) Returns a fixed value of (3)dynamic. Setting this object to the value invalid(2) has the effect of invalidating the corresponding entry in the ipNetToMediaTable. That is, it effectively dissociates the interface identified with said entry from the mapping identified with said entry. It is an implementation-specific matter as to whether the agent removes an invalidated entry from the table. Accordingly, management stations must be prepared to receive tabular information from agents that corresponds to entries not currently in use. Proper interpretation of such entries requires examination of the relevant ipNetToMediaType object.

icmp The `icmp` group reports statistics about the ICMP group.

`icmpInMsgs` – The total number of ICMP messages that the entity received. Note that this counter includes all those counted by icmpInErrors. (counter)

`icmpInErrors` – The number of ICMP messages that the entity received but determined as having ICMP-specific errors (bad ICMP checksums, bad length, and the like.). (counter)

`icmpInDestUnreachs` – The number of ICMP Destination Unreachable messages received. (counter)

`icmpInTimeExcds` – The number of ICMP Time Exceeded messages received. (counter)

`icmpInParmProbs` – The number of ICMP Parameter Problem messages received. (counter)

`icmpInSrcQuenchs` – The number of ICMP Source Quench messages received. (counter)

`icmpInRedirects` – The number of ICMP Redirect messages received. (counter)

`icmpInEchos` – The number of ICMP Echo (request) messages received. (counter)

`icmpInEchoReps` – The number of ICMP Echo Reply messages received. (counter)

`icmpInTimestamps` – The number of ICMP Timestamp (request) messages received. (counter)

`icmpInTimestampReps` – The number of ICMP Timestamp Reply messages received. (counter)

`icmpInAddrMasks` – The number of ICMP Address Mask Request messages received. (counter)

`icmpInAddrMaskReps` – The number of ICMP Address Mask Reply messages received. (counter)

`icmpOutMsgs` – The total number of ICMP messages that this entity attempted to send. Note that this counter includes all those counted by icmpOutErrors. (counter)

`icmpOutErrors` – The number of ICMP messages that this entity did not send due to problems discovered within ICMP, such as a lack of buffers. This value should not include errors discovered outside the ICMP layer, such as the inability of IP to route the resultant datagram. In some implementations there may be no types of errors that contribute to this counter's value. (counter)

`icmpOutDestUnreachs` – The number of ICMP Destination Unreachable messages sent. (counter)

`icmpOutTimeExcds` – The number of ICMP Time Exceeded messages sent. (counter)

`icmpOutParmProbs` – The number of ICMP Parameter Problem messages sent. (counter)

`icmpOutSrcQuenchs` – The number of ICMP Source Quench messages sent. (counter)

`icmpOutRedirects` – The number of ICMP Redirect messages sent. For a host, this object will always be zero, since hosts do not send redirects. (counter)

`icmpOutEchos` – The number of ICMP Echo (request) messages sent. (counter)

`icmpOutEchoReps` – The number of ICMP Echo Reply messages sent. (counter)

`icmpOutTimestamps` – The number of ICMP Timestamp (request) messages sent. (counter)

`icmpOutTimestampReps` – The number of ICMP Timestamp Reply messages sent. (counter)

`icmpOutAddrMasks` – The number of ICMP Address Mask Request messages sent. (counter)

`icmpOutAddrMaskReps` – The number of ICMP Address Mask Reply messages sent. (counter)

tcp The `tcp` group reports statistics about the TCP group.

`tcpRtoAlgorithm` – The algorithm used to determine the timeout value used for retransmitting unacknowledged octets. (enum)

`tcpRtoMin` – The minimum value permitted by a TCP implementation for the retransmission timeout, measured in milliseconds. More refined semantics for objects of this type depend upon the algorithm used to determine the retransmission timeout. In particular, when the timeout algorithm is rsre(3), an object of this type has the semantics of the LBOUND quantity described in RFC 793. (int)

`tcpRtoMax` – The maximum value permitted by a TCP implementation for the retransmission timeout, measured in milliseconds. More refined semantics for objects of this type depend upon the algorithm used to determine the retransmission timeout. In particular, when the timeout algorithm is rsre(3), an object of this type has the semantics of the UBOUND quantity described in RFC 793. (int)

`tcpMaxConn` – The limit on the total number of TCP connections that the entity can support. In entities where the maximum number of connections is dynamic, this object should contain the value –1. (int)

`tcpActiveOpens` – The number of times that TCP connections have made a direct transition to the SYN-SENT state from the CLOSED state. (counter)

`tcpPassiveOpens` – The number of times that TCP connections have made a direct transition to the SYN-RCVD state from the LISTEN state. (counter)

`tcpAttemptFails` – The number of times that TCP connections have made a direct transition to the CLOSED state from either the SYN-SENT state or the SYN-RCVD state, plus the number of times TCP connections have made a direct transition to the LISTEN state from the SYN-RCVD state. (counter)

`tcpEstabResets` – The number of times TCP connections have made a direct transition to the CLOSED state from either the ESTABLISHED state or the CLOSE-WAIT state. (counter)

`tcpCurrEstab` – The number of TCP connections for which the current state is either ESTABLISHED or CLOSE-WAIT. (gauge)

`tcpInSegs` – The total number of segments received, including those received in error. This count includes segments received on currently established connections. (counter)

`tcpOutSegs` – The total number of segments sent, including those on current connections but excluding those containing only retransmitted octets. (counter)

`tcpRetransSegs` – The total number of segments retransmitted - that is, the number of TCP segments transmitted containing one or more previously transmitted octets. (counter)

`tcpInErrs` – The total number of segments received in error (for example, bad TCP checksums). (counter)

`tcpOutRsts` – The number of TCP segments sent containing the RST flag. (counter)

tcpConnTable The `tcpConnTable` is a table containing TCP connection-specific information.

`tcpConnState` – The state of this TCP connection. (enum)

The only value that may be set by a management station is deleteTCB(12). Accordingly, it is appropriate for an agent to return a "badValue" response if a management station attempts to set this object to any other value.

If a management station sets this object to the value deleteTCB(12), then this has the effect of deleting the TCB (as defined in RFC 793) of the corresponding connection on the managed node. This results in immediate termination of the connection.

As an implementation-specific option, an RST segment may be sent from the managed node to the other TCP endpoint. (Note, however, that RST segments are not sent reliably.)

`tcpConnLocalAddress` – The local IP address for this TCP connection. For a connection in the listen state that is willing to accept connections for any IP interface associated with the node, the value 0.0.0.0 is used. (netaddress)

`tcpConnLocalPort` – The local port number for this TCP connection. (int)

`tcpConnRemAddress` – The remote IP address for this TCP connection. (netaddress)

`tcpConnRemPort` – The remote port number for this TCP connection. (int)

upd The `udp` group reports statistics about the UDP group.

`udpInDatagrams` – The total number of UDP datagrams delivered to UDP users. (counter) Returns a fixed value of 0.

`udpNoPorts` – The total number of received UDP datagrams for which there was no application at the destination port. (counter) Returns a fixed value of 0.

`udpInErrors` – The number of received UDP datagrams that could not be delivered for reasons other than the lack of an application at the destination port. (counter)

`udpOutDatagrams` – The total number of UDP datagrams sent from this entity. (counter) Returns a fixed value of 0.

udpTable The `udpTable` is a table containing UDP listener information.

`udpLocalAddress` – The local IP address for this UDP listener. For a UDP listener that is willing to accept datagrams for any IP interface associated with the node, the value 0.0.0.0 is used. (netaddress)

`udpLocalPort` – The local port number for this UDP listener. (int)

snmp The `snmp` group reports statistics about the SNMP group.

`snmpInPkts` – The total number of Messages delivered to the SNMP entity from the transport service. (counter)

`snmpOutPkts` – The total number of SNMP Messages passed from the SNMP protocol entity to the transport service. (counter)

`snmpInBadVersions` – The total number of SNMP Messages delivered to the SNMP protocol entity that were for an unsupported SNMP version. (counter)

`snmpInBadCommunityNames` – The total number of SNMP Messages delivered to the SNMP protocol entity that used a SNMP community name not known to said entity. (counter)

`snmpInBadCommunityUses` – The total number of SNMP Messages delivered to the SNMP protocol entity, which represented an SNMP operation not allowed by the SNMP community named in the Message. (counter)

`snmpInASNParseErrs` – The total number of ASN.1 or BER errors encountered by the SNMP protocol entity when decoding received SNMP Messages. (counter)

`snmpInTooBigs` – The total number of SNMP PDUs delivered to the SNMP protocol entity for which the value of the error-status field is “tooBig.” (counter)

`snmpInNoSuchNames` – The total number of SNMP PDUs delivered to the SNMP protocol entity for which the value of the error-status field is “noSuchName.” (counter)

`snmpInBadValues` – The total number of SNMP PDUs delivered to the SNMP protocol entity for which the value of the error-status field is “badValue.” (counter)

`snmpInReadOnlys` – The total number valid SNMP PDUs delivered to the SNMP protocol entity for which the value of the error-status field is "readOnly." It should be noted that it is a protocol error to generate an SNMP PDU that contains the value "readOnly" in the error-status field. This object is provided as a means of detecting incorrect implementations of the SNMP. (counter)

`snmpInGenErrs` – The total number of SNMP PDUs delivered to the SNMP protocol entity for which the value of the error-status field is "genErr." (counter)

`snmpInTotalReqVars` – The total number of MIB objects successfully retrieved by the SNMP protocol entity as the result of receiving valid SNMP Get-Request and Get-Next PDUs. (counter)

`snmpInTotalSetVars` – The total number of MIB objects successfully altered by the SNMP protocol entity as the result of receiving valid SNMP Set-Request PDUs. (counter)

`snmpInGetRequests` – The total number of SNMP Get-Request PDUs accepted and processed by the SNMP protocol entity. (counter)

`snmpInGetNexts` – The total number of SNMP Get-Next PDUs accepted and processed by the SNMP protocol entity. (counter)

`snmpInSetRequests` – The total number of SNMP Set-Request PDUs accepted and processed by the SNMP protocol entity. (counter)

`snmpInGetResponses` – The total number of SNMP Get-Response PDUs accepted and processed by the SNMP protocol entity. (counter)

`snmpInTraps` – The total number of SNMP Trap PDUs accepted and processed by the SNMP protocol entity. (counter)

`snmpOutTooBigs` – The total number of SNMP PDUs generated by the SNMP protocol entity for which the value of the error-status field is "tooBig." (counter)

`snmpOutNoSuchNames` – The total number of SNMP PDUs generated by the SNMP protocol entity for which the value of the error-status is "noSuchName." (counter)

`snmpOutBadValues` – The total number of SNMP PDUs generated by the SNMP protocol entity for which the value of the error-status field is "badValue." (counter)

`snmpOutGenErrs` – The total number of SNMP PDUs generated by the SNMP protocol entity for which the value of the error-status field is "genErr." (counter)

`snmpOutGetRequests` – The total number of SNMP Get-Request PDUs which have been generated by the SNMP protocol entity. (counter)

`snmpOutGetNexts` – The total number of SNMP Get-Next PDUs generated by the SNMP protocol entity. (counter)

`snmpOutSetRequests` – The total number of SNMP Set-Request PDUs generated by the SNMP protocol entity. (counter)

`snmpOutGetResponses` – The total number of SNMP Get-Response PDUs generated by the SNMP protocol entity. (counter)

`snmpOutTraps` – The total number of SNMP Trap PDUs generated by the SNMP protocol entity. (counter)

`snmpEnableAuthenTraps` – Indicates whether the SNMP agent process is permitted to generate authentication-failure traps. The value of this object overrides any configuration information. As such, it provides a means whereby all authentication-failure traps may be disabled. (enum)

Note that this object must be stored in non-volatile memory, so that it remains constant between reinitializations of the network management system.

The following are Sun-specific group and table definitions.

sunSystem

The `sunSystem` group reports general system information.

`agentDescr` – The SNMP agent's description of itself. (string[255])

`hostID` – The unique Sun hardware identifier. The value returned is four byte binary string. (octet[4])

`motd` – The first line of `/etc/motd`. (string[255])

`unixTime` – The UNIX system time. Measured in seconds since January 1, 1970 UTC. (counter)

sunProcessTable

The `sunProcessTable` table reports UNIX process table information.

`psProcessID` – The process identifier for this process. (int)

`psParentProcessID` – The process identifier of this process's parent. (int)

`psProcessSize` – The combined size of the data and stack segments (in kilobytes.) (int)

`psProcessCpuTime` – The CPU time (including both user and system time) consumed so far. (int)

`psProcessState` – The run-state of the process. (octet[4])

R	Runnable
T	Stopped
P	In page wait

D	Non-interruptable wait
S	Sleeping (less than 20 seconds)
I	Idle (more than 20 seconds)
Z	Zombie

`psProcessWaitChannel` – Reason process is waiting. (octet[16])

`psProcessTTY` – Terminal, if any, controlling this process. (octet[16])

`psProcessUserName` – Name of the user associated with this process. (octet[16])

`psProcessUserID` – Numeric form of the name of the user associated with this process. (int)

`psProcessName` – Command name used to invoke this process. (octet[64])

`psProcessStatus` – Setting this variable will cause a signal of the set value to be sent to the process. (int)

sunHostPerf The `sunHostPerf` group reports hostperf information.

`rsUserProcessTime` – Total number of timeticks used by user processes since the last system boot. (counter)

`rsNiceModeTime` – Total number of timeticks used by "nice" mode since the last system boot. (counter)

`rsSystemProcessTime` – Total number of timeticks used by system processes since the last system boot. (counter)

`rsIdleModeTime` – Total number of timeticks in idle mode since the last system boot. (counter)

`rsDiskXfer1` – Total number of disk transfers since the last boot for the first of four configured disks. (counter)

`rsDiskXfer2` – Total number of disk transfers since the last boot for the second of four configured disks. (counter)

`rsDiskXfer3` – Total number of disk transfers since the last boot for the third of four configured disks. (counter)

`rsDiskXfer4` – Total number of disk transfers since the last boot for the fourth of four configured disks. (counter)

`rsVPagesIn` – Number of pages read in from disk. (counter)

`rsVPagesOut` – Number of pages written to disk. (counter)

`rsVSwapIn` – Number of pages swapped in. (counter)

`rsVSwapOut` – Number of pages swapped out. (counter)

`rsVIntr` – Number of device interrupts. (counter)

`rsIfInPackets` – Number of input packets. (counter)

`rsIfOutPackets` – Number of output packets. (counter)

`rsIfInErrors` – Number of input errors. (counter)

`rsIfOutErrors` – Number of output errors. (counter)

`rsIfCollisions` – Number of output collisions. (counter)

Files

`/etc/snmp/conf/snmpd.conf`	configuration information
`/etc/snmp/conf/mibiisa.acl`	access control file
`/var/snmp/mib/sun.mib`	standard SNMP MIBII file

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	service/network/snmp/mibiisa
Interface Stability	Obsolete

See Also `inetd(1M)`, `select(3C)`, `recvfrom(3SOCKET)`, `sendto(3SOCKET)`, `attributes(5)`, `gld(7D)`

Diagnostics

`cannot dispatch request`	The proxy cannot dispatch the request. The rest of the message indicates the cause of the failure.
`select(3C) failed`	A `select(3C)` call failed. The rest of the message indicates the cause of the failure.
`sendto(3SOCKET) failed`	A `sendto(3SOCKET)` call failed. The rest of the message indicates the cause of the failure.
`recvfrom(3SOCKET) failed`	A `recvfrom(3SOCKET)` call failed. The rest of the message indicates the cause of the failure.
`no response from system`	The SNMP agent on the target system does not respond to SNMP requests. This error might indicate that the

	SNMP agent is not running on the target system, the target system is down, or the network containing the target system is unreachable.
`response too big`	The agent could not fit the results of an operation into a single SNMP message. Split large groups or tables into smaller entities.
`missing attribute`	An attribute is missing from the requested group.
`bad attribute type`	An object attribute type received from the SNMP agent that does not match the attribute type specified by the proxy agent schema. The rest of the message indicates the expected type and received type.
`cannot get sysUpTime`	The proxy agent cannot get the variable *sysUpTime* from the SNMP agent.
`sysUpTime type bad`	The variable *sysUpTime* received from the SNMP agent has the wrong data type.
`unknown SNMP error`	An unknown SNMP error was received.
`bad variable value`	The requested specified an incorrect syntax or value for a set operation.
`variable is read only`	The SNMP agent did not perform the set request because a variable to set may not be written.
`general error`	A general error was received.
`cannot make request PDU`	An error occurred building a request PDU.
`cannot make request varbind list`	An error occurred building a request variable binding list.
`cannot parse response PDU`	An error occurred parsing a response PDU.

`request ID - response ID mismatch`	The response ID does not match the request ID.
`string contains non-displayable characters`	A displayable string contains non-displayable characters.
`cannot open schema file`	An error occurred opening the proxy agent schema file.
`cannot parse schema file`	The proxy agent couldn't parse the proxy agent schema file.
`cannot open host file`	An error occurred opening the file associated with the *na.snmp.hostfile* keyword in `/etc/snmp/conf/snmpd.conf`
`cannot parse host file`	The proxy agent was unable to parse the file associated with the *na.snmp.hostfile* keyword in `/etc/snmp/conf/snm.conf`.
`attribute unavailable for set operations`	The set could not be completed because the attribute was not available for set operations.

Bugs The `mibiisa` utility returns the wrong interface speed for the SBUS FDDI interface (for example, “bf0”).

The `mibiisa` utility does not return a MAC address for the SBUS FDDI interface (for example, “bf0”).

Process names retrieved from `mibiisa` contain a leading blank space.

When you change attribute values in the system group with an SNMP set request, the change is effective only as long as `mibiisa` is running. `mibiisa` does not save the changes to `/etc/snmp/conf/snmpd.conf`.

Name mkbootmedia – create bootable Solaris ISO image

Synopsis

```
/usr/bin/mkbootmedia -v [-l label] media-root iso
```

Description The `mkbootmedia` utility takes *media-root* (the root of an on-disk Solaris install media) as input and creates a bootable Solaris ISO image in the file *iso*, using `mkisofs(8)`. The file can then be burned onto a CD/DVD with utilities such as `cdrw(1)` or `cdrecord(1)`. (Neither `mkisofs(8)` nor `cdrecord(1)` are SunOS man pages.)

Caution – The directory tree *media-root* must contain the file `boot/grub/stage2_eltorito`, which will be written to the media boot sectors. This file will be modified with some boot information, thus it must be writable. If necessary, first save a copy prior to running this utility.

Options The following options are supported:

`-l` *label*
Sets *label* as the label/volume name of the ISO image.

`-v`
Verbose. Multiple `-v` options increase verbosity.

Operands The following operands are supported:

media-root
Top-level directory of an on-disk Solaris install media.

iso
Name of the output file which will contain the resulting ISO image.

Examples **EXAMPLE 1** Creating an ISO Image and Burning a CD/DVD

The following commands create an ISO image from the content of `s10u1` and burn the image to a CD/DVD.

```
# /usr/bin/mkbootmedia s10u1 s10u1.iso
# /usr/bin/cdrw -i s10u1.iso
```

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os
Interface Stability	Committed

See Also `cdrw(1)`, `attributes(5)`

`mkisofs(8)`, (`/usr/share/man/man8/mkisofs.8`), in the `SUNWfsman` package (not a SunOS man page)

Name mkdevalloc – Make device_allocate entries

Synopsis

```
/usr/sbin/mkdevalloc
```

Description The `mkdevalloc` command writes to standard out a set of `device_allocate(4)` entries describing the system's frame buffer, audio, and removable media devices.

The `mkdevalloc` command is used by the device allocation service:

```
svc:/system/device/allocate:default
```

...to create or update the `device_allocate(4)` file. The device allocation service is managed by `smf(5)` and described in `device_allocate(1M)`.

Entries are generated based on the device special files found in `/dev`. For the different categories of devices, the `mkdevalloc` command checks for the following files under `/dev`:

```
audio           /dev/audio, /dev/audioctl, /dev/sound/...
tape            /dev/rst*, /dev/nrst*, /dev/rmt/...
removable disk  /dev/sr*, /dev/nsr*, /dev/dsk/c0t?d0s?, /dev/rdsk/c0t?d0s?
frame buffer    /dev/fb
```

All entries set the *device-minimum* and *device-maximum* fields to the hex representations of `ADMIN_LOW` and `ADMIN_HIGH`, respectively. The *device-authorization* field is set to `solaris.device.allocate`, except for the `framebuffer` entry, where it is set to `*`. The *device-name*, *device-type* and *device-clean* fields are set to the following values:

```
                device-name      device-type   device-clean

audio           audio            audio         audio_clean_wrapper
tape            mag_tape_0,1,... st            st_clean
floppy          floppy_0,1,...   fd            disk_clean
removable disk  cdrom_0,1,...    sr            disk_clean
frame buffer    framebuffer      fb            /bin/true
```

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os
Interface Stability	Obsolete

See Also `allocate(1)`, `device_allocate(1M)`, `device_allocate(4)`, `attributes(5)`, `smf(5)`

Notes `mkdevalloc` might not be supported in a future release of the Solaris operating system.

Name mkdevmaps – make device_maps entries

Synopsis

```
/usr/sbin/mkdevmaps
```

Description The `mkdevmaps` command writes to standard out a set of `device_maps(4)` entries describing the system's frame buffer, audio, and removable media devices.

The `mkdevmaps` command is used by the device allocation service:

```
svc:/system/device/allocate:default
```

...to create or update the `device_maps(4)` file. The device allocation service is managed by `smf(5)` and described in `device_allocate(1M)`.

Entries are generated based on the device special files found in `/dev`. For the different categories of devices, the `mkdevmaps` command checks for the following files under `/dev`:

```
audio           /dev/audio, /dev/audioctl, /dev/sound/...
tape            /dev/rst*, /dev/nrst*, /dev/rmt/...
removable disk  /dev/dsk/c0t?d0s?, /dev/rdsk/c0t?d0s?
frame buffer    /dev/fb
```

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os
Interface Stability	Obsolete

See Also `allocate(1)`, `device_allocate(1M)`, `device_maps(4)`, `attributes(5)`, `smf(5)`

Notes `mkdevmaps` might not be supported in a future release of the Solaris operating system.

Name mkfifo – make FIFO special file

Synopsis

```
/usr/bin/mkfifo [-m mode] path...
```

Description The `mkfifo` utility creates the FIFO special files named by its argument list. The arguments are taken sequentially, in the order specified; and each FIFO special file is either created completely or, in the case of an error or signal, not created at all.

If errors are encountered in creating one of the special files, `mkfifo` writes a diagnostic message to the standard error and continues with the remaining arguments, if any.

The `mkfifo` utility calls the library routine `mkfifo(3C)`, with the *path* argument is passed as the *path* argument from the command line, and *mode* is set to the equivalent of `a=rw`, modified by the current value of the file mode creation mask `umask(1)`.

Options The following option is supported:

`-m` *mode* Set the file permission bits of the newly-created FIFO to the specified *mode* value. The *mode* option-argument will be the same as the *mode* operand defined for the `chmod(1)` command. In <*symbolic*mode> strings, the *op* characters + and – will be interpreted relative to an assumed initial mode of `a=rw`.

Operands The following operand is supported:

file A path name of the FIFO special file to be created.

Usage See `largefile(5)` for the description of the behavior of `mkfifo` when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

Environment Variables See `environ(5)` for descriptions of the following environment variables that affect the execution of `mkfifo`: `LANG`, `LC_ALL`, `LC_CTYPE`, `LC_MESSAGES`, and `NLSPATH`.

Exit Status The following exit values are returned:

`0` All the specified FIFO special files were created successfully.

`>0` An error occurred.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os
Interface Stability	Committed
Standard	See `standards(5)`.

See Also chmod(1), umask(1), mkfifo(3C), attributes(5), environ(5), largefile(5), standards(5)

Name mkfile – create a file

Synopsis

```
mkfile [-nv] size [t | g | k | b | m] filename...
```

Description `mkfile` creates one or more files that are suitable for use as NFS-mounted swap areas, or as local swap areas. When a root user executes `mkfile()`, the sticky bit is set and the file is padded with zeros by default. When non-root users execute `mkfile()`, they must manually set the sticky bit using `chmod(1)`. The default `size` is in bytes, but it can be flagged as terabytes, gigabytes, kilobytes, blocks, or megabytes, with the `t`, `g`, `k`, `b`, or `m` suffixes, respectively. Suffixes can be uppercase or lowercase.

Options

`-n` Create an empty *filename*. The size is noted, but disk blocks are not allocated until data is written to them. Files created with this option cannot be swapped over local UFS mounts.

`-v` Verbose. Report the names and sizes of created files.

Exit Status The following exit values are returned:

`0` Success.

`>0` An error occurred.

Usage See `largefile(5)` for the description of the behavior of `mkfile` when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os
Interface Stability	Committed

See Also `chmod(1)`, `swap(1M)`, `attributes(5)`, `largefile(5)`

Name mkfs – construct a file system

Synopsis

```
mkfs [-F FSType] [generic_options]
     [-o FSType-specific_options] raw_device_file
     [operands]
```

Description The `mkfs` utility constructs a file system on the *raw_device_file* by calling the specific `mkfs` module indicated by `-F` *FSType*.

`Note`: ufs file systems are normally created with the `newfs(1M)` command.

generic_options are independent of file system type. *FSType-specific_options* is a comma-separated list of *keyword=value* pairs (with no intervening spaces), which are *FSType*-specific. *raw_device_file* specifies the disk partition on which to write the file system. It is required and must be the first argument following the *specific_options* (if any). *operands* are *FSType*-specific. See the *FSType*-specific manual page of `mkfs` (for example, `mkfs_ufs(1M)`) for a detailed description.

Options The following are the generic options for `mkfs`:

`-F` Specify the *FSType* to be constructed. If `-F` is not specified, the *FSType* is determined from `/etc/vfstab` by matching the *raw_device_file* with a `vfstab` entry, or by consulting the `/etc/default/fs` file.

`-V` Echo the complete command line, but do not execute the command. The command line is generated by using the options and arguments provided and adding to them information derived from `/etc/vfstab` or `/etc/default/fs`. This option may be used to verify and validate the command line.

`-m` Return the command line which was used to create the file system. The file system must already exist. This option provides a means of determining the command used in constructing the file system.

`-o` Specify *FSType*-specific options. See the manual page for the `mkfs` module specific to the file system type.

Usage See `largefile(5)` for the description of the behavior of `mkfs` when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

Files

`/etc/default/fs` Default file system type. Default values can be set for the following flags in `/etc/default/fs`. For example: `LOCAL=ufs`

- `LOCAL` The default partition for a command if no *FSType* is specified.

`/etc/vfstab` List of default parameters for each file system

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also mkfs_ufs(1M), newfs(1M), vfstab(4), attributes(5), largefile(5)

Manual pages for the *FSType*-specific modules of mkfs.

Notes This command might not be supported for all *FSType*s.

You can use lofiadm to create a file that appears to a mkfs command as a raw device. You can then use a mkfs command to create a file system on that device. See lofiadm(1M) for examples of creating a UFS and a PC (FAT) file system (using mkfs_ufs(1M) and mkfs_pcfs(1M)) on a device created by lofiadm.

Name mkfs_pcfs – construct a FAT file system

Synopsis

```
mkfs -F pcfs [generic_options] [-o FSType_specific_options]
     raw_device_file
```

Description The `pcfs`-specific module of `mkfs` constructs a File Allocation Table (FAT) on removable media (JAZ disk, ZIP disk, PCMCIA card), a hard disk, or a file (see NOTES). FATs are the standard MS-DOS and Windows file system format.

`mkfs` for `pcfs` determines an appropriate FAT size for the medium, then it installs an initial boot sector and an empty FAT. A sector size of 512 bytes is used. `mkfs` for `pcfs` can also install the initial file in the file system (see the `pcfs`-specific `-o i` option). This first file can optionally be marked as read-only, system, and/or hidden.

If you want to construct a FAT with `mkfs` for `pcfs` on a medium that is not formatted, you must first perform a low-level format on the medium with `format(1M)`. The media must also be partitioned with the `fdisk(1M)` utility. Note that all existing data on a disk partition, if any, is destroyed when a new FAT is constructed.

generic_options are supported by the generic `mkfs` command. See `mkfs(1M)` for a description of these options.

raw_device_file indicates the device on which to write unless the `-o N` option has been specified, or if the `-V` or `-m` generic options are passed from the generic `mkfs` module.

Options See `mkfs(1M)` for the list of supported generic options.

The following options are supported:

`-o` *FSType_specific_options*
Specify `pcfs` file system-specific options in a comma-separated list with no intervening spaces. If invalid options are specified, a warning message is printed and the invalid options are ignored.

b=*label*
Label the media with volume label. The volume label is restricted to 11 uppercase characters.

B=*filename*
Install *filename* as the boot loader in the file system's boot sector. If you don't specify a boot loader, an MS-DOS boot loader is installed. The MS-DOS boot loader requires specific MS-DOS system files to make a disk bootable. See `NOTES` for more information.

fat=*n*
The size of a FAT entry. Currently, 12, 16, and 32 are valid values. The default is 16.

h
Mark the first file installed as a hidden file. The `-i` option must also be specified.

hidden=*n*
Set the number of hidden sectors to *n*. This is the number of sectors on the physical disk preceding the start of the volume (which is the boot sector itself). This defaults to a computed valued (based on the fdisk table) for disks. This option may be used only in conjunction with the `nofdisk` option.

i=*filename*
Install *filename* as the initial file in the new file system. The initial file's contents are guaranteed to occupy consecutive clusters at the start of the files area. When creating bootable media, a boot program should be specified as the initial file.

nofdisk
Do not attempt to find an `fdisk` table on the medium. Instead rely on the `size` option for determining the partition size. By default, the created FAT is 16 bits and begins at the first sector of the device. This origination sector can be modified with the hidden option (`-h`).

nsect=*n*
The number of sectors per track on the disk. If not specified, the value is determined by using a `dkio(7I)` ioctl to get the disk geometry.

ntrack=*n*
The number of tracks per cylinder on the disk. If not specified, the value is determined by using a `dkio(7I)` ioctl to get the disk geometry.

N
No execution mode. Print normal output, but do not actually write the file system to the medium. This is most useful when used in conjunction with the verbose option.

r
Mark the first file installed as read-only. The `-i` option must also be specified.

reserve=*n*
Set the number of reserved sectors to *n*. This is the number of sectors in the volume, preceding the start of the first FAT, including the boot sector. The value should always be at least 1, and the default value is exactly 1.

s
Mark the first file installed as a system file. The `-i` option must also be specified.

size=*n*
The number of sectors in the file system. If not specified, the value is determined from the size of the partition given in the fdisk table.

spc=*n*
The size of the allocation unit for space within the file system, expressed as a number of sectors. The default value depends on the FAT entry size and the size of the file system.

v
Verbose output. Describe, in detail, operations being performed.

Files *raw_device_file*

The device on which to build the FAT.

`mkfs_pcfs` supports MBR (Master Boot Record) partitions and GPT (GUID Partition Table) partitions. GPT is part of the EFI (Extensible Firmware Interface) standard. For both x86 and SPARC, for a GPT-labeled disk, you can specify the partition using the logical device pathname with no suffix, for example, `/dev/rdsk/c0t0d0s0`. In GPT, this corresponds to the first partition on the disk.

On x86 for MBR partitions, you can specify the proper partition using the logical device pathname corresponding to the partition. For example, `/dev/rdsk/c0t0d0p1` corresponds to first partition in the MBR, or `/dev/rdsk/c0t0d0p5` corresponds to first logical partition in the extended partition. Alternatively, using a suffix is also acceptable. For example, in `/dev/rdsk/c0t0d0p0:c`, `mkfs_pcfs` recognizes `:c` as the first partition that can accept a FAT file system.

For removable media with MBR partitions on SPARC, you need to specify a disk device name with a suffix to indicate the proper partition. For example, in the name `/dev/rdsk/c0t0d0s2:c`, the `:c` suffix indicates that the partition can accept the new FAT.

For a file, *raw_device_file* is the block device name returned by `lofiadm(1M)`.

Examples The media in these examples must be formatted before running `mkfs` for `pcfs`. See DESCRIPTION for more details.

EXAMPLE 1 Creating a FAT File System on a Disk

The following command creates a FAT file system on the second fdisk partition of a disk attached to an x86 based system:

```
mkfs -F pcfs /dev/rdsk/c0d0p0:d
```

EXAMPLE 2 Creating a FAT File System on a ZIP Disk

The following command creates a FAT file system on a ZIP disk located on a SPARC based system:

```
mkfs -F pcfs /dev/rdsk/c0t4d0s2:c
```

EXAMPLE 3 Creating a FAT File System on a JAZ Disk

The following command creates a FAT file system on a JAZ disk located on a SPARC based system and overrides the sectors/track and tracks/cylinder values obtained from the device's controller:

```
mkfs -F pcfs -o nsect=32,ntrack=64 /dev/rdsk/c0t3d0s2:c
```

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/file-system/pcfs
Interface Stability	Committed

See Also fdisk(1M), format(1M), lofiadm(1M), mkfs(1M), attributes(5), dkio(7I)

Notes You can use lofiadm to create a file that appears to a mkfs command (for example, mkfs_pcfs or mkfs_ufs) as a raw device. You can then use a mkfs command to create a file system on that device. See lofiadm(1M) for examples of creating a UFS and a PC (FAT) file system on a device created by lofiadm.

Name mkfs_udfs – construct a udfs file system

Synopsis

```
mkfs -F udfs [generic_options] [-o specific_options] raw_device_file
     [size]
```

Description This is the universal disk format file system (udfs) -specific module of the `mkfs` command. `mkfs` constructs a `udfs` file system with a root directory.

Options See `mkfs`(1M) for the list of supported *generic_options*.

The following options are supported:

`-o` *specific_options* Specify a `udfs`-specific option. Specify `udfs` file system specific options in a comma-separated list with no intervening spaces. If invalid options are specified, a warning message is printed and the invalid options are ignored.

The following *specific_options* are available:

N Print the file system parameters without actually creating the file system.

label=*string* Specify the label to be written into the volume header structures. Specify *string* as the name of the label. If *string* is not specified, a default *string* is generated in the form of `*NoLabel*`.

Operands The following operands are supported:

raw_device_file Specify the disk partition on which to write.

size Specify the number of 512-byte blocks in the file system.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/file-system/udfs

See Also `fsck`(1M),`mkfs`(1M), `attributes(5)`

Diagnostics

`not currently a valid file system`

The specified device does not contain a valid `udfs` file system.

`Invalid size: larger than the partition size`

Number of blocks given as parameter to create the file system is larger than the size of the device specified.

`is mounted can't mkfs`

Device is in use, cannot create file system when the device is in use.

```
preposterous size
```

Negative size parameter provided is invalid.

```
sector size must be between 512, 8192 bytes
```

Sector size given is not in the valid range.

```
Volume integrity sequence descriptors too long
File set descriptor too long.
```

Not enough space to create volume integrity sequence or file set descriptor.

```
mkfs: argument out of range
```

One of the arguments is out of range.

```
mkfs: bad numeric arg
```

One of the arguments is potentially a bad numeric.

Notes You can use `lofiadm` to create a file that appears to a `mkfs` command (for example, `mkfs_pcfs` or `mkfs_ufs`) as a raw device. You can then use a `mkfs` command to create a file system on that device. See `lofiadm(1M)` for examples of creating a UFS and a PC (FAT) file system on a device created by `lofiadm`.

Name mkfs_ufs – construct a UFS file system

Synopsis

```
mkfs -F ufs [generic_options] [-o FSType_specific_options] raw_device_file
    [size]
```

Description The UFS-specific module of `mkfs` builds a UFS file system with a root directory and a `lost+found` directory (see `fsck(1M)`).

The UFS-specific `mkfs` is rarely run directly. Use the `newfs(1M)` command instead.

raw_device_file indicates the disk partition on which to create the new file system. If the `-o N`, `-V`, or `-m` options are specified, the *raw_device_file* is not actually modified. *size* specifies the number of disk sectors in the file system, where a disk sector is usually 512 bytes. This argument must follow the *raw_device_file* argument and is required (even with `-o N`), unless the `-V` or `-m` generic options are specified.

generic_options are supported by the generic `mkfs` command. See `mkfs(1M)` for a description of these options.

Options The following generic options are supported:

`-m` Print the command line that was used to create the existing file system.

`-V` Print the current `mkfs` command line.

Options The following UFS-specific options are supported:

`-o` Use one or more of the following values separated by commas (with no intervening spaces) to specify UFS-specific options:

`apc=`*n* The number of alternate sectors per cylinder to reserve for bad block replacement for SCSI devices only. The default is `0`.

This option is not applicable for disks with EFI labels and is ignored.

`bsize=`*n* The logical block size of the file system in bytes, either `4096` or `8192`. The default is `8192`. The sun4u architecture does not support the `4096` block size.

`calcbinsb` Sends to stdout a binary (machine-readable) version of the superblock that would be used to create a file system with the specified configuration parameters.

`calcsb` Sends to stdout a human-readable version of the superblock that would be used to create a file system with the specified configuration parameters.

`cgsize=`*n* The number of cylinders per cylinder group, ranging from `16` to `256`. The default is calculated by dividing the number of sectors in the file system by the number of sectors in a gigabyte. Then, the result is multiplied by `32`. The default value is always between `16` and `256`.

The per-cylinder-group meta data must fit in a space no larger than what is available in one logical file system block. If too large a `cgsize` is requested, it is changed by the minimum amount necessary.

`fragsize=`*n* The smallest amount of disk space in bytes that can be allocated to a file. *fragsize* must be a power of 2 divisor of *bsize*, where:

bsize / *fragsize* is 1, 2, 4, or 8.

This means that if the logical block size is `4096`, legal values for *fragsize* are `512`, `1024`, `2048`, and `4096`. When the logical block size is `8192`, legal values are `1024`, `2048`, `4096`, and `8192`. The default value is `1024`.

For file systems greater than 1 terabyte or for file systems created with the `mtb=y` option, *fragsize* is forced to match block size (*bsize*).

`free=`*n* The minimum percentage of free space to maintain in the file system between 0% and 99%, inclusively. This space is off-limits to users. Once the file system is filled to this threshold, only the superuser can continue writing to the file system.

The default is ((64 Mbytes/partition size) * 100), rounded down to the nearest integer and limited between 1% and 10%, inclusively.

This parameter can be subsequently changed using the `tunefs(1M)` command.

`gap=`*n* Rotational delay. This option is obsolete in the Solaris 10 release. The value is always set to `0`, regardless of the input value.

`maxcontig=`*n* The maximum number of logical blocks, belonging to one file, that are allocated contiguously. The default is calculated as follows:

`maxcontig =` *disk drive maximum transfer size / disk block size*

If the disk drive's maximum transfer size cannot be determined, the default value for `maxcontig` is calculated from kernel parameters as follows:

If `maxphys` is less than `ufs_maxmaxphys`, which is typically 1 Mbyte, then `maxcontig` is set to `maxphys`. Otherwise, `maxcontig` is set to `ufs_maxmaxphys`.

You can set `maxcontig` to any positive integer value.

The actual value will be the lesser of what has been specified and what the hardware supports.

You can subsequently change this parameter by using tunefs(1M).

`mtb=y` Set the parameters of the file system to allow eventual growth to over a terabyte in total file system size. This option sets *fragsize* to be the same as *bsize*, and sets *nbpi* to 1 Mbyte, unless the `-i` option is used to make it even larger. If you explicitly set the *fragsize* or *nbpi* parameters to values that are incompatible with this option, the user-supplied value of *fragsize* or *nbpi* is ignored.

`N` Print out the file system parameters that would be used to create the file system without actually creating the file system.

`nbpi=`*n* The number of bytes per inode, which specifies the density of inodes in the file system. The number is divided into the total size of the file system to determine the number of inodes to create.

This value should reflect the expected average size of files in the file system. If fewer inodes are desired, a larger number should be used. To create more inodes, a smaller number should be given. The default is `2048`.

The number of inodes can increase if the file system is expanded with the `growfs` command.

`nrpos=`*n* The number of different rotational positions in which to divide a cylinder group. The default is `8`.

This option is not applicable for disks with EFI labels and is ignored.

`nsect=`*n* The number of sectors per track on the disk. The default is `32`.

`ntrack=`*n* The number of tracks per cylinder on the disk. The default is `16`.

This option is not applicable for disks with EFI labels and is ignored.

`opt=`*s* | *t* The file system can either be instructed to try to minimize the *time* spent allocating blocks, or to try to minimize the *space* fragmentation on the disk. The default is *time*.

This parameter can be subsequently changed with the tunefs(1M) command.

`rps=`*n* The rotational speed of the disk, in revolutions per second. The default is `60`.

Note that you specify *rps* for `mkfs` and *rpm* for `newfs`.

This option is not applicable for disks with EFI labels and is ignored.

Alternatively, parameters can be entered as a list of space-separated values (without keywords) whose meaning is positional. In this case, the `-o` option is omitted and the list follows the size operand. This is the way `newfs` passes the parameters to `mkfs`.

Operands The following operands are supported:

raw_device_file The disk partition on which to write.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also `fsck(1M)`, `mkfs(1M)`, `newfs(1M)`, `tunefs(1M)`, `dir_ufs(4)`, `attributes(5)`, `ufs(7FS)`

Diagnostics The following error message typically occurs with very high density disks. On such disks, the file system structure cannot encode the proper disk layout information. However, such disks have enough onboard intelligence to make up for any layout deficiencies, so there is no actual impact on performance. The warning that performance might be impaired can be safely ignored.

```
Warning: insufficient space in super block for
rotational layout tables with nsect sblock.fs_nsect
and ntrak sblock.fs_ntrak. (File system performance may be impaired.)
```

The following error message occurs when the disk geometry results in a situation where the last truncated cylinder group cannot contain the correct number of data blocks. Some disk space is wasted.

```
Warning: inode blocks/cyl group (grp) >= data blocks (num) in last cylinder
```

If there is only one cylinder group and if the above condition holds true, `mkfs` fails with the following error:

```
File system creation failed. There is only one cylinder group and that is
not even big enough to hold the inodes.
```

The following error message occurs when the best calculated file system layout is unable to include the last few sectors in the last cylinder group. This is due to the interaction between how much space is used for various pieces of meta data and the total blocks available in a cylinder group. Modifying `nbpi` and `cpg` might reduce this number, but it is rarely worth the effort.

```
Warning: num sector(s) in last cylinder group unallocated
```

Notes You can use `lofiadm` to create a file that appears to the `mkfs` command (for example, `mkfs_pcfs` or `mkfs_ufs`) as a raw device. You can then use the `mkfs` command to create a file system on that device. See `lofiadm(1M)` for examples of creating a UFS and a PC (FAT) file system on a device created by `lofiadm`.

Both the block and character devices, such as devices in `/dev/dsk` and `/dev/rdsk`, must be available prior to running the `mkfs` command.

Name mknod – make a special file

Synopsis

```
mknod name b major minor
mknod name c major minor
mknod name p
```

Description `mknod` makes a directory entry for a special file.

Options The following options are supported:

`b` Create a block-type special file.

`c` Create a character-type special file.

`p` Create a FIFO (named pipe).

Operands The following operands are supported:

major The *major* device number.

minor The *minor* device number; can be either decimal or octal. The assignment of major device numbers is specific to each system. You must be the super-user to use this form of the command.

name A special file to be created.

Usage See `largefile(5)` for the description of the behavior of `mknod` when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also `ftp(1)`, `mknod(2)`, `symlink(2)`, `attributes(5)`, `largefile(5)`

Notes If `mknod(2)` is used to create a device, the major and minor device numbers are always interpreted by the kernel running on that machine.

With the advent of physical device naming, it would be preferable to create a symbolic link to the physical name of the device (in the `/devices` subtree) rather than using `mknod`.

Name mkntfs – create an NTFS file system

Synopsis

```
mkntfs [options] device [number_of_sectors]

mkntfs [-C] [-c cluster-size] [-F] [-f] [-H heads] [-h] [-I]
     [-L volume-label] [-l] [-n] [-p part-start-sect] [-Q] [-q]
     [-S sectors-per-track] [-s sector-size] [-T] [-V] [-v]
     [-z mft-zone-multiplier] [--debug] device [number-of-sectors]
```

Description The `mkntfs` utility is used to create an NTFS file system on a device, usually a disk partition, or file. The *device* operand is the special file corresponding to the device; for example, `/dev/dsk/c0d0p0`. The *number-of-sectors* operand is the number of blocks on the device. If omitted, `mkntfs` automatically figures the file system size.

Options Supported options are listed below. Most options have both single-letter and full-name forms. Multiple single-letter options that do not take an argument can be combined. For example, `-fv` is the equivalent of `-f -v`. A full-name option can be abbreviated to a unique prefix of its name.

Options are divided among basic, advanced, output, and help options, as listed below.

Basic Options

`-C`, `--enable-compression`
Enable compression on the volume.

`-f`, `--fast` or `-q`, `--quick`
Perform quick (fast) format. This option skips both zeroing of the volume and bad sector checking.

`-L`, `--label` *string*
Set the volume label for the filesystem to *string*.

`-n`, `--no-action`
Causes `mkntfs` to not actually create a file system, but display what it would do if it were to create a file system. All formatting steps are carried out except the actual writing to the device.

Advanced Options

`-c`, `--cluster-size` *bytes*
Specify the size of clusters in bytes. Valid cluster size values are powers of two, with at least 256, and at most 65536, bytes per cluster. If omitted, `mkntfs` uses 4096 bytes as the default cluster size.

Note that the default cluster size is set to be at least equal to the sector size, as a cluster cannot be smaller than a sector. Also, note that values greater than 4096 have the side effect that compression is disabled on the volume. This is due to limitations in the NTFS compression algorithm used by Windows.

`-F`, `--force`
Force `mkntfs` to run, even if the specified device is not a block special device, or appears to be mounted.

`-H`, `--heads` *num*
Specify the number of heads. The maximum is 65535 (0xffff). If omitted, `mkntfs` attempts to determine the number of heads automatically. If that fails a default of 0 is used. Note that specifying *num* is required for Windows to be able to boot from the created volume.

`-I`, `--no-indexing`
Disable content indexing on the volume. This option is only meaningful on Windows 2000 and later. Windows NT 4.0 and earlier ignore this, as they do not implement content indexing.

`-p`, `--partition-start` *sector*
Specify the partition start sector. The maximum is 4294967295 (2^{32-1}). If omitted, `mkntfs` attempts to determine *sector*automatically. If that fails, a default of 0 is used. Note that specifying *sector* is required for Windows to be able to boot from the created volume.

`-S`, `--sectors-per-track` *num*
Specify the number of sectors per track. The maximum is 65535 (0xffff). If omitted, mkntfs attempts to determine the number of sectors-per-track automatically and if that fails a default of 0 is used. Note that sectors-per-track is required for Windows to be able to boot from the created volume.

`-s`, `--sector-size` *bytes*
Specify the size of sectors in bytes. Valid sector size values are 256, 512, 1024, 2048, and 4096. If omitted, `mkntfs` attempts to determine the sector-size automatically. If that fails, a default of 512 bytes per sector is used.

`-T`, `--zero-time`
Fake the time to be 00:00:00 UTC, Jan 1, 1970, instead of the current system time. This can be useful for debugging purposes.

`-z`, `--mft-zone-multiplier` *num*
Set the master file table (MFT) zone multiplier, which determines the size of the MFT zone to use on the volume. The MFT zone is the area at the beginning of the volume reserved for the MFT, which stores the on-disk inodes (MFT records). It is noteworthy that small files are stored entirely within the inode; thus, if you expect to use the volume for storing large numbers of very small files, it is useful to set the zone multiplier to a higher value. Although the MFT zone is resized on the fly as required during operation of the NTFS driver, choosing an optimal value reduces fragmentation. Valid values are `1`, `2`, `3`, and `4`. The values have the following meaning:

```
MFT zone        MFT zone size
multiplier      (% of volume size)
1       12.5% (default)
2       25.0%
3       37.5%
4       50.0%
```

Output Options

`--debug`
Includes the verbose output from the `-v` option, as well as additional output useful for debugging `mkntfs`.

`-q`, `--quiet`
Verbose execution. Errors are written to stderr, no output to stdout occurs at all. Useful if `mkntfs` is run in a script.

`-v`, `--verbose`
Verbose execution.

Help Options

`-h`, `--help`
Show a list of options with a brief description of each one.

`-l`, `--license`
Display the `mkntfs` licensing information and exit.

`-V`, `--version`
Display the `mkntfs` version number and exit.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/file-system/ntfsprogs
Interface Stability	Uncommitted

See Also `ntfsprogs(1M)`, `ntfsresize(1M)`, `ntfsundelete(1M)`, `attributes(5)`

`http://wiki.linux-ntfs.org`

Authors `mkntfs` was written by Anton Altaparmakov, Richard Russon, Erik Sornes and Szabolcs Szakacsits.

Name mkpwdict – maintain password-strength checking database

Synopsis

```
/usr/bin/mkpwdict [-s dict1,... ,dictN]
     [-d destination-path]
```

Description The `mkpwdict` command adds words to the dictionary-lookup database used by `pam_authtok_check(5)` and `passwd(1)`.

Files containing words to be added to the database can be specified on the command-line using the `-s` flag. These source files should have a single word per line, much like `/usr/share/lib/dict/words`.

If `-s` is omitted, `mkpwdict` will use the value of `DICTIONLIST` specified in `/etc/default/passwd` (see `passwd(1)`).

The database is created in the directory specified by the `-d` option. If this option is omitted, `mkpwdict` uses the value of `DICTIONDBDIR` specified in `/etc/default/passwd` (see `passwd(1)`). The default location is `/var/passwd`.

Options The following options are supported:

`-s` Specifies a comma-separated list of files containing words to be added to the dictionary-lookup database.

`-d` Specifies the target location of the dictionary-database.

Files

`/etc/default/passwd` See `passwd(1)`.

`/var/passwd` default destination directory

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os
Interface Stability	Committed

See Also `passwd(1)`, `attributes(5)`, `pam_authtok_check(5)`

Name modinfo – display information about loaded kernel modules

Synopsis `/usr/sbin/modinfo [-c] [-w] [-i` *module-id*`]`

Description The `modinfo` utility displays information about the loaded modules. The format of the information is as follows:

Id Loadaddr Size Info Rev Module Name

where *Id* is the module ID, *Loadaddr* is the starting text address in hexadecimal, *Size* is the size of text, data, and bss in hexadecimal bytes, *Info* is module specific information, *Rev* is the revision of the loadable modules system, and *Module Name* is the filename and description of the module.

The module specific information is the block and character major numbers for drivers, the system call number for system calls, and unspecified for other module types.

Options The following options are supported:

`-c` Display the number of instances of the module loaded and the module's current state.

`-i` *module-id* Display information about this module only.

`-w` Do not truncate module information at `80` characters.

Examples **EXAMPLE 1** Displaying the Status of a Module

The following example displays the status of module 2:

```
example% modinfo -i 2
Id   Loadaddr   Size Info Rev Module Name
 2    ff08e000   1734   -   1  swapgeneric (root and swap configuration)
```

EXAMPLE 2 Displaying the Status of Kernel Modules

The following example displays the status of some kernel modules:

```
example% modinfo
Id Loadaddr   Size Info Rev Module Name
 2  ff08e000   1734   -   1  swapgeneric
 4  ff07a000   3bc0   -   1  specfs (filesystem for specfs)
 6  ff07dbc0   2918   -   1  TS (time sharing sched class)
 7  ff0804d8    49c   -   1  TS_DPTBL (Time sharing dispatch table)
 8  ff04a000  24a30   2   1  ufs (filesystem for ufs)
 9  ff080978   c640 226   1  rpcmod (RPC syscall)
 9  ff080978   c640   -   1  rpcmod (rpc interface str mod)
10  ff08cfb8  2031c   -   1  ip (IP Streams module)
10  ff08cfb8  2031c   2   1  ip (IP Streams device)
```

EXAMPLE 3 Using the -c Option

Using the `modinfo` command with the -c option displays the number of instances of the module loaded and the module's current state.

```
example% modinfo -c
Id    Loadcnt Module Name                          State
  1          0 krtld                          UNLOADED/UNINSTALLED
  2          0 genunix                        UNLOADED/UNINSTALLED
  3          0 platmod                        UNLOADED/UNINSTALLED
  4          0 SUNW,UltraSPARC-IIi            UNLOADED/UNINSTALLED
  5          0 cl_bootstrap                   UNLOADED/UNINSTALLED
  6          1 specfs                         LOADED/INSTALLED
  7          1 swapgeneric                    UNLOADED/UNINSTALLED
  8          1 TS                             LOADED/INSTALLED
  9          1 TS_DPTBL                       LOADED/INSTALLED
 10          1 ufs                            LOADED/INSTALLED
 11          1 fssnap_if                      LOADED/INSTALLED
```

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os
Interface Stability	Committed

See Also

`modload(1M)`, `modunload(1M)`, `attributes(5)`

Name modload – load a kernel module

Synopsis

```
modload [-p] [-e exec_file] filename
```

Description The `modload` command loads the loadable module *filename* into the running system.

filename is an object file produced by `ld -r`. If *filename* is an absolute pathname then the file specified by that absolute path is loaded. If *filename* does not begin with a slash (`/`), then the path to load *filename* is relative to the current directory unless the `-p` option is specified.

The kernel's `modpath` variable can be set using the `/etc/system` file. The default value of the kernel's `modpath` variable is set to the path where the operating system was loaded. Typically this is `/kernel /usr/kernel`.

For example, the following command looks for `./drv/foo`:

```
example# modload drv/foo
```

The following command looks for `/kernel/drv/foo` and then `/usr/kernel/drv/foo`:

```
example# modload -p drv/foo
```

Options The following options are supported:

`-e` *exec_file* — Specify the name of a shell script or executable image file that is executed after the module is successfully loaded. The first argument passed is the module ID (in decimal). The other argument is module specific. The module specific information is: the block and character major numbers for drivers, the system call number for system calls, or, for other module types, the index into the appropriate kernel table. See `modinfo(1M)`

`-p` — Use the kernel's internal modpath variable as the search path for the module.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also `ld(1)`, `add_drv(1M)`, `kernel(1M)`, `modinfo(1M)`, `modunload(1M)`, `system(4)`, `attributes(5)`, `modldrv(9S)`, `modlinkage(9S)`, `modlstrmod(9S)`, `module_info(9S)`

Writing Device Drivers

Notes Use `add_drv(1M)` to add device drivers, not `modload`. See *Writing Device Drivers* for procedures on adding device drivers.

Name modunload – unload a module

Synopsis

```
modunload -i module_id [-e exec_file]
```

Description `modunload` unloads a loadable module from the running system. The *module_id* is the ID of the module as shown by `modinfo(1M)`. If `ID` is `0`, all modules that were autoloaded which are unloadable, are unloaded. Modules loaded by `modload(1M)` are not affected.

Options The following options are supported:

`-e` *exec_file* Specify the name of a shell script or executable image file to be executed before the module is unloaded. The first argument passed is the module id (in decimal). There are two additional arguments that are module specific. For loadable drivers, the second argument is the driver major number. For loadable system calls, the second argument is the system call number. For loadable exec classes, the second argument is the index into the `execsw` table. For loadable filesystems, the second argument is the index into the `vfssw` table. For loadable streams modules, the second argument is the index into the `fmodsw` table. For loadable scheduling classes, the second argument is the index into the class array. Minus one is passed for an argument that does not apply.

`-i` *module_id* Specify the module to be unloaded.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also `modinfo(1M)`, `modload(1M)`, `update_drv(1M)`, `attributes(5)`

Notes The `modunload` command is often used on driver modules to force the system to reread the associated driver configuration file. While this works in the current Solaris release, it is not the supported way to reread the configuration file and is not guaranteed to work in future releases. The supported way for rereading driver configuration file is through the `update_drv(1M)` command.

Name mofcomp – compile MOF files into CIM classes

Synopsis

```
/usr/sadm/bin/mofcomp [-c cimom_hostname ] [-h]
     [-j filename] [-n namespace] [-o dirname]
     [-p password ] [-CIQ] [-u username] [-v ] [-version]
     [-x] file
```

Description The `mofcomp` utility is executed during installation to compile managed object format (MOF) files that describe the Common Information Model (CIM) and Solaris Schemas into the CIM Object Manager Repository, a central storage area for management data. The CIM Schema is a collection of class definitions used to represent managed objects that occur in every management environment. The Solaris Schema is a collection of class definitions that extend the CIM Schema and represent managed objects in a typical Solaris operating environment.

The `mofcomp` utility must be run as root or equivalent privileges, or as a user with write access to the namespace in which you are compiling. When performing a privileged operation, you must enter the `-u` or `-u` and `-p` options, which are described below.

MOF is a language for defining CIM classes and instances. MOF files are ASCII text files that use the MOF language to describe CIM objects. A CIM object is a computer representation or model of a managed resource, such as a printer, disk drive, or CPU.

Many sites store information about managed resources in MOF files. Because MOF can be converted to Java, Java applications that can run on any system with a Java Virtual Machine can interpret and exchange this information. You can also use the `mofcomp` utility to compile MOF files at any time after installation.

Options The following options are supported:

`-c` *cimom_hostname*
: Specify a remote system running the CIM Object Manager.

`-C`
: Run the compiler set with the class option, which updates a class if it exists, and returns an error if the class does not exist. If you do not specify this option, the compiler adds a CIM class to the connected namespace, and returns an error if the class already exists.

`-h`
: List the arguments to the `mofcomp` utility.

`-I`
: Run the compiler set with the instance option, which updates an instance if it exists, and returns an error if the instance does not exist. If you do not specify this option, the compiler adds a CIM instance to the connected namespace, and returns an error if the instance already exists.

`-j` *filename*
: Generate Java Beans and Java Interfaces to manage the CIM instances related to the CIM classes in the MOF being compiled.

The contents of *filename* are:

```
PACKAGE=Java package name
IMPORTS=import1:...:importN
<EXCEPTIONS=exception1:...:exceptionN
```

`PACKAGE` is a valid Java package name to include in all generated Java source. `IMPORTS` is an optional colon separated list of valid Java classes to be imported in all generated Java source. `EXCEPTIONS` is an optional colon separated list of valid Java exceptions to be thrown by the methods in all generated Java source.

`-n` *namespace*
Requests that the compiler load the MOF file into the namespace specified as *namespace*. The default namespace (`root\cimv2`) is used unless this switch is used or a `#pragma` *namespace* (`namespace`) statement appears in the MOF file. If both the `-n` *namespace* switch and the `#pragma` *namespace* construct are used, all namespaces are created, but the objects are created only in the `#pragma namespaces`.

`-o` *dirname*
Run compiler in standalone mode, without the CIM Object Manager. Specify dirname as the directory in which the compiler output is to be stored. In this mode, the CIM Object Manager need not be running.

`-p` *password*
Specify a password for connecting to the CIM Object Manager. Use this option for compilations that require privileged access to the CIM Object Manager. If you specify both `-p` and `-u`, you must type the password on the command line, which can pose a security risk. A more secure way to specify a password is to specify `-u` but not `-p`, so that the compiler will prompt for the password.

`-Q`
Run the compiler set with the qualifier types option, which updates a qualifier type if it exists, and returns an error if the qualifier type does not exist. If you do not specify this option, the compiler adds a CIM qualifier type to the connected namespace, and returns an error if the qualifier type already exists.

`-u` *username*
Specify user name for connecting to the CIM Object Manager. Use this option for compilations that require privileged access to the CIM Object Manager. If you specify both `-p` and `-u`, you must type the password on the command line, which can pose a security risk. A more secure way to specify a password is to specify `-u` but not `-p`, so that the compiler will prompt for the password.

`-v`
Run the compiler in verbose mode, which displays compiler messages.

`-version`
Display the version of the MOF compiler.

`-x`
Generate XML documents for the CIM classes defined in the input MOF file.

Operands The following operands are supported:

file
The pathname of the file to be compiled.

Exit Status The `mofcomp` utility exits with `0` upon success and a positive integer upon failure.

Files MOF files are installed in `/usr/sadm/mof`.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWwbco

See Also `init.wbem(1M)`, `mofreg(1M)`, `wbemadmin(1M)`, `wbemlogviewer(1M)`, `attributes(5)`, `wbem(5)`

Name mofreg – register MOF classes with WBEM services

Synopsis

```
/usr/sadm/bin/mofreg -r tag file

/usr/sadm/bin/mofreg -s

/usr/sadm/bin/mofreg -u tag [file]
```

Description The `mofreg` command is used by package and patch install scripts, or by any applications that wish to register managed object format (MOF) classes with Sun The Web-Based Enterprise Management (WBEM) services.

The WBEM services daemon (Common Information Model or CIM object manager) processes at start up the files that are specified by `mofreg` commands. Files are processed in the order that the individual `mofreg` commands are executed.

As an alternative to using the `mofreg` command, MOFs can be registered or unregistered by manipulating directories in `/var/sadm/wbem/logr`. Instead of running the `mofreg -r tag file` version fo the command you can create a directory named `tag` under `/var/sadm/wbem/logr/preReg` and copy `file` to the `tag` directory.

Similarly, instead of running the `mofreg -u tag [file]` command, you can create a directory named `tag` under `/var/sadm/wbem/logr/preUnreg` and copy the optional `file` to the `tag` directory.

The entries are processed in increasing order of last modification time of the `tag` directories. If you issue `mofreg` commands in rapid succession, the timestamps might be the same. If you have a situation where the timestamp order is critical, you can place appropriate sleeps between the successive registration or unregistration operations. As with the `mofreg` command, processing is done at next restart or by using the `-s` option.

This alternative mechanism is typically used in package install scripts which do not have access to `/usr`, and therefore do not have access to the `mofreg` command. This case arises when packages are installed for diskless clients.

Options The following options are supported:

`-r` *tag file* The *file* argument is the actual MOF registration file. Its form is identical to the MOF syntax as defined by the Distributed Management Task Force (DMTF). The only difference is the addition of the following 3 new pseudo-pragmas, which are variations of the namespace pragma. The name of file cannot end in `.unreg`.

```
#pragma namespace("__create")
#pragma namespace("__delete")
#pragma namespace("__modify")
```

These three pragmas are used specify if the elements following the pragmas should be created, deleted, or modified by the CIM object manager. The `__delete` pragma can currently only be applied for a `mofreg -u` command.

The *tag* argument is a unique string that specifies the identity of the registry action. This tag can be set to the package name or the patch number if the `mofreg` script is being invoked through packages/patches, though any tag can be specified.

Errors and warnings that are encountered when the CIM object manager handles the `mofreg` script are logged. Processing of the `mofreg` script stops at the first error. Specific warnings include:

```
Element already defined - the element already exists and
cannot be created.

Element not found - the element does not exist and
cannot be modified.
```

The error conditions are:

```
Key modification - A class cannot be modified if its keys
are being changed.

Other mod compilation errors.
```

`-s` Forces the CIM object manager to immediately process outstanding registry requests, instead of at the next restart. This currently requires Java.

`-u` *tag* [*file*] Undoes the operations performed during `mof` registry.

The *tag* argument must correspond to the value set during the original `mofreg` invocation. If no `mofreg` was done with the original *tag*, the command does not succeed.

If required, an *unreg* file can be specified. If no *unreg* file is specified, the CIM object manager automatically undoes the actions of the registry. Any class created by the registry process is removed and any classes modified by the registry revert to the old state.

The `mofreg` command does not take care of cases where packages and patches make conflicting changes to classes. This should be taken care of by the standard patch and package conflict resolution.

Exit Status The following exit values are returned:

`0` Successful completion.

`1` An error occurred. The reason for error is displayed.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWwbco

See Also `init.wbem`(1M), `mofcomp`(1M), `wbemadmin`(1M), `wbemlogviewer`(1M), `attributes`(5), `wbem`(5)

Name monitor – SPARC system PROM monitor

Synopsis

```
STOP–A

BREAK

initial system power-on

exit from a client program, e.g., the Operating System
```

Description The CPU board of a workstation contains one or more EPROMs or EEPROMs. The program which executes from the PROMs is referred to as "the monitor". Among other things, the monitor performs system initialization at power-on and provides a user interface.

Monitor Prompt The monitor of earlier workstations was known as the `SunMON` monitor and displayed the > for its prompt. See the `SunMON MONITOR USAGE` section for further details.

Existing workstations use a monitor which is known as the OpenBoot monitor. The OpenBoot monitor typically displays `ok` as its prompt, but it may also display the > prompt under certain circumstances.

If the `'auto-boot?'` NVRAM parameter is set to `'false'` when the workstation is powered on, the system does not attempt to boot and the monitor issues its prompt. If `'auto-boot'` is set to `'true'`, the system initiates the boot sequence. The boot sequence can be aborted by simultaneously pressing two keys on the system's keyboard: L1 and A (on older keyboards), or Stop and A (on newer keyboards). Either a lower case `a` or an upper case `A` works for the keyboard abort sequence. If a console has been attached by way of one of the system's serial ports then the abort sequence can be accomplished by sending a BREAK. See `tip(1)`.

When the NVRAM `'security-mode'` parameter has been turned on, or when the value of the `'sunmon-compat?'` parameter is true, then the OpenBoot monitor displays the message: `Type b (boot), c (continue), or n (new command mode)`

and the > prompt appears.

Openboot Prom Usage Some of the more useful commands that can be issued from OpenBoot's `ok` prompt are described here. Refer to the *OpenBoot 2.x Command Reference Manual* book for a complete list of commands.

Help Help for various functional areas of the OpenBoot monitor can be obtained by typing `help`. The help listing provides a number of other key words which can then be used in the help command to provide further details.

NVRAM Parameters Each workstation contains one or more NVRAM devices which contains unique system ID information, as well as a set of user-configurable parameters. The NVRAM parameters allow the user a certain level of flexibility in configuring the system to act in a given manner under a specific set of circumstances.

See `eeprom(1M)` for a description of the parameters and information regarding setting the parameters from the OS level.

The following commands can be used at the OpenBoot monitor to access the NVRAM parameters.

`printenv`	Used to list the NVRAM parameters, along with their default values and current values.
`setenv` *pn pv*	Used to set or modify a parameter. The *pn* represents the parameter name, and *pv* represents the parameter value.
`set-default` *pn*	Used to set an individual parameter back to its default value.
`set-defaults`	Used to reset all parameters to their default values. (Note that `'set-defaults'` only affects parameters that have assigned default values.)

Security Parameters

Newer OpenBoot monitors contain user interfaces that support the storage and listing of keys for later use by client programs.

`list-security-keys`	Lists the names of keys currently stored on a machine.
`set-security-key` *keyname* [*keydata*]	Stores key data *keydata* in a key named *keyname*. Actual key data can be up to 32 bytes in length. The maximum length of *keyname* is 64 bytes, which allows for the hex-formatted ASCII used to present the key data. If *keydata* is not present, *keyname* and its corresponding data is deleted.

Hardware Checks and Diagnostics

The following commands are available for testing or checking the system's hardware. If the `'diag-switch?'` NVRAM parameter is set to true when the system is powered on, then a Power-On Self Test (POST) diagnostic is run, if present, sending its results messages to the system's serial port A. Not all of the commands shown are available on all workstations.

`test-all`	Run the diagnostic tests on each device which has provided a self-test.
`test /memory`	Run the main memory tests. If the NVRAM parameter `'diag-switch?'` is set to true, then all of main memory is tested. If the parameter is false then only the amount of memory specified in the `'selftest-#megs'` NVRAM parameter is tested.
`test net`	Test the network connection for the on-board network controller.
`watch-net`	Monitor the network attached to the on-board net controller.
`watch-net-all`	Monitor the network attached to the on-board net controller, as well as the network controllers installed in SBus slots.
`watch-clock`	Test the system's clock function.

System Information

The following commands are available for displaying information about the system. Not all commands are available on all workstations.

`banner`	Display the power-on banner.
`.enet-addr`	Display the system's Ethernet address.
`.idprom`	Display the formatted contents of the IDPROM.
`module-info`	Display information about the system's processor(s).
`probe-scsi`	Identify the devices attached to the on-board SCSI controller.
`probe-scsi-all`	Identify the devices attached to the on-board SCSI controller as well as those devices which are attached to SBus SCSI controllers.
`show-disks`	Display a list of the device paths for installed SCSI disk controllers.
`show-displays`	Display a list of the device paths for installed display devices.
`show-nets`	Display a list of the device paths for installed Ethernet controllers.
`show-sbus`	Display list of installed SBus devices.
`show-tapes`	Display a list of the device paths for installed SCSI tape controllers.
`show-ttys`	Display a list of the device paths for tty devices.
`.traps`	Display a list of the SPARC trap types.
`.version`	Display the version and date of the OpenBoot PROM.

Emergency Commands

These commands must be typed from the keyboard, they do not work from a console which is attached by way of the serial ports. With the exception of the Stop-A command, these commands are issued by pressing and holding down the indicated keys on the keyboard immediately after the system has been powered on. The keys must be held down until the monitor has checked their status. The Stop-A command can be issued at any time after the console display begins, and the keys do not need to be held down once they've been pressed. The Stop-D, Stop-F and Stop-N commands are not allowed when one of the security modes has been set. Not all commands are available on all workstations.

Stop (L1)	Bypass the Power-On Self Test (POST). This is only effective if the system has been placed into the diagnostic mode.
Stop-A (L1-A)	Abort the current operation and return to the monitor's default prompt.
Stop-D (L1-D)	Set the system's `'diag-switch?'` NVRAM parameter to `'true'`, which places the system in diagnostic mode. POST diagnostics, if present, are run, and the messages are displayed by way of the system's serial port A.
Stop-F (L1-F)	Enter the OpenBoot monitor before the monitor has probed the system for devices. Issue the `'fexit'` command to continue with system initialization.

Stop-N (L1-N) Causes the NVRAM parameters to be reset to their default values. Note that not all parameters have default values.

Line Editor Commands

The following commands can be used while the monitor is displaying the `ok` prompt. Not all of these editing commands are available on all workstations.

CTRL-A	Place the cursor at the start of line.
CTRL-B	Move the cursor backward one character.
ESC-B	Move the cursor backward one word.
CTRL-D	Erase the character that the cursor is currently highlighting.
ESC-D	Erase the portion of word from the cursor's present position to the end of the word.
CTRL-E	Place the cursor at the end of line.
CTRL-F	Move the cursor forward one character.
ESC-F	Move the cursor forward one word.
CTRL-H	Erase the character preceding the cursor (also use Delete or Back Space)
ESC-H	Erase the portion of the word which precedes the cursor (use also CTRL-W)
CTRL-K	Erase from the cursor's present position to the end of the line.
CTRL-L	Show the command history list.
CTRL-N	Recall the next command from the command history list
CTRL-P	Recall a previous command from the command history list.
CTRL-Q	Quote the next character (used to type a control character).
CTRL-R	Retype the current line.
CTRL-U	Erase from the cursor's present position to the beginning of the line.
CTRL-Y	Insert the contents of the memory buffer into the line, in front (to the left) of the cursor.

nvramrc

The `nvramrc` is an area of the system's NVRAM where users may store Forth programs. The programs which are stored in the `nvramrc` are executed each time the system is reset, provided that the `'use-nvramrc?'` NVRAM parameter has been set to `'true'`. Refer to the *OpenBoot 2.x Command Reference Manual*book for information on how to edit and use the `nvramrc`.

Restricted Monitor

The command `'old-mode'` is used to move OpenBoot into a restricted monitor mode, causing the > prompt to be displayed. Only three commands are allowed while in the restricted monitor; the `'go'` command (to resume a program which was interrupted with the

Stop-A command), the `'n'` command (to return to the normal OpenBoot monitor), and boot commands. The restricted monitor's boot commands approximate the older `SunMON` monitor's boot command syntax. If a `'security-mode'` has been turned on then the restricted monitor becomes the default monitor environment. The restricted monitor may also become the default environment if the `'sunmon-compat?'` NVRAM parameter is set to true. Not all workstations have the `'sunmon-compat?'` parameter.

SunMON Prom Usage

The following commands are available systems with older `SunMON`-based PROM:

`+|–`
Increment or decrement the current address and display the contents of the new location.

`^C` *source destination n*
(caret-C) Copy, byte-by-byte, a block of length *n* from the `source` address to the *destination* address.

`^I` *program*
(caret-I) Display the compilation date and location of *program*.

`^T` *virtual_address*
(caret-T) Display the physical address to which *virtual_address* is mapped.

`b` [`!`] [*device* [`(`*c*`,`*u*`,`*p* `)`]] [*pathname*] [*arguments_list*]

`b[?]`
Reset appropriate parts of the system and bootstrap a program. A '`!`' (preceding the *device* argument) prevents the system reset from occurring. Programs can be loaded from various devices (such as a disk, tape, or Ethernet). '`b`' with no arguments causes a default boot, either from a disk, or from an Ethernet controller. '`b?`' displays all boot devices and their *device*s.

device	one of	
	`le`	Lance Ethernet
	`ie`	Intel Ethernet
	`sd`	SCSI disk, CDROM
	`st`	SCSI 1/4– or 1/2–inch tape
	`fd`	Diskette
	`id`	IPI disk
	`mt`	Tape Master 9-track 1/2–inch tape
	`xd`	Xylogics 7053 disk
	`xt`	Xylogics 1/2–inch tape
	`xy`	Xylogics 440/450 disk

c A controller number (`0` if only one controller),

u A unit number (`0` if only one driver), and

p A partition.

pathname A pathname for a program such as `/stand/diag`.

arguments_list A list of up to seven arguments to pass to the program being booted.

`c` [*virtual_address*]
Resume execution of a program. When given, *virtual_address* is the address at which execution resumes. The default is the current PC. Registers are restored to the values shown by the `d`, and `r` commands.

`d` [*window_number*]
Display (dump) the state of the processor. The processor state is observable only after:

- An unexpected trap was encountered.
- A user program dropped into the monitor (by calling *abortent*).
- The user manually entered the monitor by typing L1–A or BREAK.

The display consists of the following:

- The special registers: PSR, PC, nPC, TBR, WIM, and Y
- Eight global registers
- 24 window registers (8 *in*, 8 *local*, and 8 *out*), corresponding to one of the 7 available windows. If a Floating-Point Unit is on board, its status register along with 32 floating-point registers are also shown.

window_number Display the indicated *window_number*, which can be any value between `0` and `6`, inclusive. If no window is specified and the PSR's current window pointer contains a valid window number, registers from the window that was active just prior to entry into the monitor are displayed. Otherwise, registers from window 0 are displayed.

`e` [*virtual_address*] [*action*] . . .
Open the 16-bit word at *virtual_address* (default zero). The address is interpreted in the address space defined by the `s` command. See the `a` command for a description of *action*.

`f` *virtual_address1 virtual_address2 pattern* [*size*]
Fill the bytes, words, or long words from *virtual_address1* (lower) to *virtual_address2* (higher) with the constant, *pattern*. The `size` argument can take one of the following values:

`b` byte format (the default)

`w` word format

`l` long word format

For example, the following command fills the address block from `0x1000` to `0x2000` with the word pattern, `0xABCD`:

```
f 1000 2000 ABCD W
```

`g` [*vector*] [*argument*]
`g` [*virtual_address*] [*argument*]
Goto (jump to) a predetermined or default routine (first form), or to a user-specified routine (second form). The value of *argument* is passed to the routine. If the *vector* or *virtual_address* argument is omitted, the value in the PC is used as the address to jump to.

To set up a predetermined routine to jump to, a user program must, prior to executing the monitor's `g` command, set the variable `*romp->v_vector_cmd` to be equal to the virtual address of the desired routine. Predetermined routines need not necessarily return control to the monitor.

The default routine, defined by the monitor, prints the user-supplied *vector* according to the format supplied in *argument*. This format can be one of:

`%x` hexadecimal

`%d` decimal

`g0`
Force a panic and produce a crash dump when the monitor is running as a result of the system being interrupted,

`g4`
(Sun-4 systems only) Force a kernel stack trace when the monitor is running as a result of the system being interrupted,

`h`
Display the help menu for monitor commands and their descriptions. To return to the monitor's basic command level, press ESCAPE or `q` before pressing RETURN.

`i` [*cache_data_offset*] [*action*] . . .
Modify cache data RAM command. Display and/or modify one or more of the cache data addresses. See the `a` command for a description of *action*.

`j` [*cache_tag_offset*] [*action*] . . .
Modify cache tag RAM command. Display and/or modify the contents of one or more of the cache tag addresses. See the `a` command for a description of *action*.

`k` [*reset_level*]
Reset the system, where *reset_level* is:

`0` Reset VMEbus, interrupt registers, video monitor (Sun-4 systems). This is the default.

`1` Software reset.

2 Power-on reset. Resets and clears the memory. Runs the EPROM-based diagnostic self test, which can take several minutes, depending upon how much memory is being tested.

`kb`
Display the system banner.

`l` [*virtual_address*] [*action*] . . .
Open the long word (32 bit) at memory address *virtual_address* (default zero). The address is interpreted in the address space defined by the `s` command (below). See the `a` command for a description of *action*.

`m` [*virtual_address*] [*action*] . . .
Open the segment map entry that maps *virtual_address* (default zero). The address is interpreted in the address space defined by the `s` command. See the `a` command for a description of *action*.

`ne`

`ni`
Disable, enable, or invalidate the cache, respectively.

`o` [*virtual_address*] [action*]* . . .
Open the byte location specified by *virtual_address* (default zero). The address is interpreted in the address space defined by the `s` command. See the `a` command for a description of *action*.

`p` [*virtual_address*] [*action*]. . .
Open the page map entry that maps *virtual_address* (default zero) in the address space defined by the `s` command. See the `a` command for a description of *action*.

`q` [*eeprom_offset*] [*action*]. . .
Open the EEPROM *eeprom_offset* (default zero) in the EEPROM address space. All addresses are referenced from the beginning or base of the EEPROM in physical address space, and a limit check is performed to insure that no address beyond the EEPROM physical space is accessed. This command is used to display or modify configuration parameters, such as: the amount of memory to test during self test, whether to display a standard or custom banner, if a serial port (A or B) is to be the system console, etc. See the `a` command for a description of *action*.

`r` [*register_number*]
`r` [*register_type*]
`r` [*w window_number*]
Display and/or modify one or more of the IU or FPU registers. A hexadecimal *register_number* can be one of:

`0x00`–`0x0f`	window(0,i0)–window(0,i7), window(0,i0)—window(0,i7)
`0x16`–`0x1f`	window(1,i0)–window(1,i7), window(1,i0)—window(1,i7)

`0x20–0x2f`	window(2,i0)–window(2,i7), window(2,i0)—window(2,i7)
`0x30–0x3f`	window(3,i0)–window(3,i7), window(3,i0)—window(3,i7)
`0x40–0x4f`	window(4,i0)–window(4,i7), window(4,i0)—window(4,i7)
`0x50–0x5f`	window(5,i0)–window(5,i7), window(5,i0)—window(5,i7)
`0x60–0x6f`	window(6,i0)–window(6,i7), window(6,i0)—window(6,i7)
`0x70–0x77`	`g0, g1, g2, g3, g4, g5, g6, g7`
`0x78–0x7d`	PSR, PC, nPC, WIM, TBR, Y.
`0x7e–0x9e`	FSR, f0–f31

Register numbers can only be displayed after an unexpected trap, a user program has entered the monitor using the *abortent* function, or the user has entered the monitor by manually typing L1–A or BREAK.

If a *register_type* is given, the first register of the indicated type is displayed. *register_type* can be one of:

`f`	floating-point
`g`	global
`s`	special

If `w` and a *window_number* (`0`—`6`) are given, the first *in*-register within the indicated window is displayed. If *window_number* is omitted, the window that was active just prior to entering the monitor is used. If the PSR's current window pointer is invalid, window 0 is used.

`s [`*asi*`])`

Set or display the Address Space Identifier. With no argument, `s` displays the current Address Space Identifier. The *asi* value can be one of:

`0x2`	control space
`0x3`	segment table
`0x4`	Page table
`0x8`	user instruction
`0x9`	supervisor instruction
`0xa`	user data
`0xb`	supervisor data
`0xc`	flush segment

`0xd` flush page

`0xe` flush context

`0xf` cache data

`u` [`echo`]

`u` [*port*] [*options*] [*baud_rate*]

`u` [`u`] [*virtual_address*]
With no arguments, display the current I/O device characteristics including: current input device, current output device, baud rates for serial ports A and B, an input-to-output echo indicator, and virtual addresses of mapped UART devices. With arguments, set or configure the current I/O device. With the `u` argument (`uu. . .`), set the I/O device to be the *virtual_address* of a UART device currently mapped.

`echo` Can be either `e` to enable input to be echoed to the output device, or `ne`, to indicate that input is not echoed.

port Assign the indicated *port* to be the current I/O device. *port* can be one of:

`a` serial port A

`b` serial port B

`k` the workstation keyboard

`s` the workstation screen

baud_rate Any legal baud rate.

options can be any combination of:

`i` input

`o` output

`u` UART

`e` echo input to output

`ne` do not echo input

`r` reset indicated serial port (`a` and `b` ports only)

If either `a` or `b` is supplied, and no *options* are given, the serial port is assigned for both input and output. If `k` is supplied with no options, it is assigned for input only. If `s` is supplied with no options, it is assigned for output only.

`v` *virtual_address1 virtual_address2* [`size`]
Display the contents of *virtual_address1* (lower) *virtual_address2* (higher) in the format specified by `size`:

`b` byte format (the default)

`w` word format

`l` long word format

Enter return to pause for viewing; enter another return character to resume the display. To terminate the display at any time, press the space bar.

For example, the following command displays the contents of virtual address space from address `0x1000` to `0x2000` in word format:

```
v 1000 2000 W
```

`w` [*virtual_address*] [*argument*]
Set the execution vector to a predetermined or default routine. Pass *virtual_address* and *argument* to that routine.

To set up a predetermined routine to jump to, a user program must, prior to executing the monitor's `w` command, set the variable `*romp->v_vector_cmd` to be equal to the virtual address of the desired routine. Predetermined routines need not necessarily return control to the monitor.

The default routine, defined by the monitor, prints the user-supplied *vector* according to the format supplied in *argument*. This format can be one of:

`%x` hexadecimal

`%d` decimal

`x`
Display a menu of extended tests. These diagnostics permit additional testing of such things as the I/O port connectors, video memory, workstation memory and keyboard, and boot device paths.

`y c` *context_number*
Display context number.

`y p|s` *context_number virtual_address*
Flush the indicated context, context page, or context segment.

`c` flush context *context_number*

`p` flush the page beginning at *virtual_address* within context *context_number*

`s` flush the segment beginning at *virtual_address* within context *context_number*

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	SPARC

See Also tip(1), boot(1M), eeprom(1M), attributes(5)

OpenBoot 2.x Command Reference Manual

Name mount, umount – mount or unmount file systems and remote resources

Synopsis

```
mount [-p | -v]

mount [-F FSType] [generic_options] [-o specific_options]
     [-O] special | mount_point

mount [-F FSType] [generic_options] [-o specific_options]
     [-O] special mount_point

mount -a [-F FSType] [-V] [current_options]
     [-o specific_options] [mount_point]...

umount [-f] [-V] [-o specific_options] special | mount_point

umount -a [-f] [-V] [-o specific_options] [mount_point]...
```

Description `mount` attaches a file system to the file system hierarchy at the *mount_point*, which is the pathname of a directory. If *mount_point* has any contents prior to the `mount` operation, these are hidden until the file system is unmounted.

`umount` unmounts a currently mounted file system, which may be specified either as a *mount_point* or as *special*, the device on which the file system resides.

The table of currently mounted file systems can be found by examining the mounted file system information file. This is provided by a file system that is usually mounted on `/etc/mnttab`. The mounted file system information is described in `mnttab(4)`. Mounting a file system adds an entry to the mount table; a `umount` removes an entry from the table.

When invoked with both the *special* and *mount_point* arguments and the `-F` option, `mount` validates all arguments except for *special* and invokes the appropriate *FSType*-specific `mount` module. If invoked with no arguments, `mount` lists all the mounted file systems recorded in the mount table, `/etc/mnttab`. If invoked with a partial argument list (with only one of *special* or *mount_point*, or with both *special* or *mount_point* specified but not *FSType*), `mount` will search `/etc/vfstab` for an entry that will supply the missing arguments. If no entry is found, and the special argument starts with `/`, the default local file system type specified in `/etc/default/fs` will be used. Otherwise the default remote file system type will be used. The default remote file system type is determined by the first entry in the `/etc/dfs/fstypes` file. After filling in missing arguments, `mount` will invoke the *FSType*-specific `mount` module.

For file system types that support it, a file can be mounted directly as a file system by specifying the full path to the file as the special argument. In such a case, the `nosuid` option is enforced. If specific file system support for such loopback file mounts is not present, you can still use `lofiadm(1M)` to mount a file system image. In this case, no special options are enforced.

Only a user with sufficient privilege (at least `PRIV_SYS_MOUNT`) can mount or unmount file systems using `mount` and `umount`. However, any user can use `mount` to list mounted file systems and resources.

Options `-F` *FSType*

Used to specify the *FSType* on which to operate. The *FSType* must be specified or must be determinable from `/etc/vfstab`, or by consulting `/etc/default/fs` or `/etc/dfs/fstypes`.

`-a` [*mount_points*. . .]

Perform mount or umount operations in parallel, when possible.

If mount points are not specified, `mount` will mount all file systems whose `/etc/vfstab` "mount at boot" field is `yes`. If mount points are specified, then `/etc/vfstab` "mount at boot" field will be ignored.

If mount points are specified, `umount` will only `umount` those mount points. If none is specified, then `umount` will attempt to `unmount` all file systems in `/etc/mnttab`, with the exception of certain system required file systems: `/`, `/usr`, `/var`, `/var/adm`, `/var/run`, `/proc`, `/dev/fd` and `/tmp`.

`-f`

Forcibly unmount a file system.

Without this option, `umount` does not allow a file system to be unmounted if a file on the file system is busy. Using this option can cause data loss for open files; programs which access files after the file system has been unmounted will get an error (`EIO`).

`-p`

Print the list of mounted file systems in the `/etc/vfstab` format. Must be the only option specified. See `BUGS`.

`-v`

Print the list of mounted file systems in verbose format. Must be the only option specified.

`-V`

Echo the complete command line, but do not execute the command. `umount` generates a command line by using the options and arguments provided by the user and adding to them information derived from `/etc/mnttab`. This option should be used to verify and validate the command line.

generic_options

Options that are commonly supported by most *FSType*-specific command modules. The following options are available:

`-m`

Mount the file system without making an entry in `/etc/mnttab`.

`-g`

Globally mount the file system. On a clustered system, this globally mounts the file system on all nodes of the cluster. On a non-clustered system this has no effect.

`-o`

Specify *FSType*-specific options in a comma separated (without spaces) list of suboptions and keyword-attribute pairs for interpretation by the *FSType*-specific module of the command. (See `mount_ufs(1M)`.) When you use `-o` with a file system that has an entry in `/etc/vfstab`, any `mount` options entered for that file system in `/etc/vfstab` are ignored.

The following options are supported:

`devices | nodevices`

Allow or disallow the opening of device-special files. The default is `devices`.

If you use `nosuid` in conjunction with `devices`, the behavior is equivalent to that of `nosuid`.

`exec | noexec`

Allow or disallow executing programs in the file system. Allow or disallow `mmap(2)` with `PROT_EXEC` for files within the file system. The default is `exec`.

`loop`

Ignored for compatibility.

`nbmand | nonbmand`

Allow or disallow non-blocking mandatory locking semantics on this file system. Non-blocking mandatory locking is disallowed by default.

If the file system is mounted with the `nbmand` option, then applications can use the `fcntl(2)` interface to place non-blocking mandatory locks on files and the system enforces those semantics. If you enable this option, it can cause standards conformant applications to see unexpected errors.

To avoid the possibility of obtaining mandatory locks on system files, do not use the `nbmand` option with the following file systems:

```
/
/usr
/etc
/var
/proc
/dev
/devices
/system/contract
/system/object
/etc/mnttab
/etc/dfs/sharetab
```

Do not use the `remount` option to change the `nbmand` disposition of the file system. The `nbmand` option is mutually exclusive of the global option. See `-g`.

`ro | rw`
Specify read-only or read-write. The default is `rw`.

`setuid | nosetuid`
Allow or disallow `setuid` or `setgid` execution. The default is `setuid`.

If you specify `setuid` in conjunction with `nosuid`, the behavior is the same as `nosuid`.

`nosuid` is equivalent to `nosetuid` and `nodevices`. When `suid` or `nosuid` is combined with `setuid` or `nosetuid` and `devices` or `nodevices`, the most restrictive options take effect.

This option is highly recommended whenever the file system is shared by way of NFS with the `root=` option. Without it, NFS clients could add `setuid` programs to the server or create devices that could open security holes.

`suid | nosuid`
Allow or disallow `setuid` or `setgid` execution. The default is `suid`. This option also allows or disallows opening any device-special entries that appear within the filesystem.

`nosuid` is equivalent to `nosetuid` and `nodevices`. When `suid` or `nosuid` is combined with `setuid` or `nosetuid` and `devices` or `nodevices`, the most restrictive options take effect.

This option is highly recommended whenever the file system is shared using NFS with the `root=`*option*, because, without it, NFS clients could add `setuid` programs to the server, or create devices that could open security holes.

`rstchown` | `norstchown`
Allow or disallow restricted `chown`. If the file system is mounted with `rstchown`, the owner of the file is prevented from changing the owner ID of the file. If the file system is mounted with `norstchown`, the user can permit ownership changes for files they own. Only the super-user or a user with appropriate privilege can arbitrarily change owner IDs.

`-O`
Overlay mount. Allow the file system to be mounted over an existing mount point, making the underlying file system inaccessible. If a mount is attempted on a pre-existing mount point without setting this flag, the mount will fail, producing the error “device busy”.

`-r`
Mount the file system read-only.

Examples

EXAMPLE 1 Mounting and Unmounting a DVD Image Directly

The following commands mount and unmount a DVD image.

EXAMPLE 1 Mounting and Unmounting a DVD Image Directly *(Continued)*

```
# mount -F hsfs /images/solaris.iso /mnt/solaris-image
# umount /mnt/solaris-image
```

Usage See `largefile(5)` for the description of the behavior of `mount` and `umount` when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

Files `/etc/mnttab`
Table of mounted file systems.

`/etc/default/fs`
Default local file system type. Default values can be set for the following flags in `/etc/default/fs`. For example: `LOCAL=ufs`

`LOCAL:`
The default partition for a command if no *FSType* is specified.

`/etc/vfstab`
List of default parameters for each file system.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also `lofiadm(1M)`, `mount_hsfs(1M)`, `mount_nfs(1M)`, `mount_pcfs(1M)`, `mount_smbfs(1M)`, `mount_tmpfs(1M)`, `mount_udfs(1M)`, `mount_ufs(1M)`, `mountall(1M)`, `umountall(1M)`, `fcntl(2)`, `mmap(2)`, `mnttab(4)`, `vfstab(4)`, `attributes(5)`, `largefile(5)`, `privileges(5)`, `lofs(7FS)`, `pcfs(7FS)`

Notes If the directory on which a file system is to be mounted is a symbolic link, the file system is mounted on the directory to which the symbolic link refers, rather than on top of the symbolic link itself.

Bugs The `mount -p` output is incorrect for `cachefs`.

Name mountall, umountall – mount, unmount multiple file systems

Synopsis

```
mountall [-F FSType] [-l | -r] [file_system_table]

umountall [-k] [-s] [-F FSType] [-l | -r] [-n]  [-Z]

umountall [-k] [-s] [-h host] [-n] [-Z]
```

Description `mountall` is used to mount file systems specified in a file system table. The file system table must be in `vfstab(4)` format. If no *file_system_table* is specified, `/etc/vfstab` is used. If – is specified as *file_system_table*, `mountall` reads the file system table from the standard input. `mountall` mounts only those file systems with the `mount at boot` field set to `yes` in the *file_system_table*.

For each file system in the file system table, the following logic is executed: if there exists a file`/usr/lib/fs/`*FSType*`/fsckall`, where *FSType* is the type of the file system, save that file system in a list to be passed later, and all at once, as arguments to the `/usr/lib/fs/`*FSType*`/fsckall` script. The `/usr/lib/fs/`*FSType*`/fsckall` script checks all of the file systems in its argument list to determine whether they can be safely mounted. If no `/usr/lib/fs/`*FSType*`/fsckall` script exists for the *FSType* of the file system, the file system is individually checked using `fsck(1M)`. If the file system does not appear mountable, it is fixed using `fsck` before the mount is attempted. File systems with a – entry in the `fsckdev` field are mounted without first being checked.

`umountall` causes all mounted file systems in the current zone except `root`, `/usr`, `/var`, `/var/adm`, `/var/run`, `/proc`, and `/dev/fd` to be unmounted. If the *FSType* is specified, `mountall` and `umountall` limit their actions to the *FSType* specified. There is no guarantee that `umountall` unmounts *busy* file systems, even if the `-k` option is specified.

Options The following options are supported:

`-F` Specify the *FSType* of the file system to be mounted or unmounted.

`-h` *host* Unmount all file systems listed in `/etc/mnttab` that are remote-mounted from host.

`-k` Use the `fuser -k` *mount-point* command. See the `fuser(1M)` for details. The `-k` option sends the `SIGKILL` signal to each process using the file. As this option spawns kills for each process, the kill messages might not show up immediately. There is no guarantee that `umountall` unmounts *busy* file systems, even if the `-k` option is specified.

`-l` Limit the action to local file systems.

`-n` List the actions that would be performed for the specified options, but do not actually execute these actions. Repeating the command without the `-n` option executes the listed actions, assuming that the `/etc/mnttab` file has not changed in the interval prior to repeating the command.

`-r` Limit the action to remote file system types.

`-s` Do not perform the `umount` operation in parallel.

`-Z` Apply the action(s) only to the file systems mounted in non-global zones. By default, `umountall` unmounts only file systems mounted in the current zone. Option `-Z` is ignored if used in a non-global zone.

Files `/etc/mnttab`
Mounted file system table

`/etc/vfstab`
Table of file system defaults

`/usr/lib/fs/`*FSType*`/fsckall`
Script called by `mountall` to perform the file system check of all file systems of type *FSType*

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os
Interface Stability	Committed
Output Stability	Uncommitted

See Also `fsck(1M)`, `fuser(1M)`, `mount(1M)`, `mnttab(4)`, `vfstab(4)`, `attributes(5)`

Diagnostics No messages are printed if the file systems are mountable and clean.

Error and warning messages come from `fsck(1M)` and `mount(1M)`.

Notes At this time, NFS is the only remote file system supported by the `-l`, `-r`, and `-h` options.

Name mountd – server for NFS mount requests and NFS access checks

Synopsis

```
/usr/lib/nfs/mountd [-v] [-r]
```

Description `mountd` is an RPC server that answers requests for NFS access information and file system mount requests. It reads the file `/etc/dfs/sharetab` to determine which file systems are available for mounting by which remote machines. See `sharetab(4)`. `nfsd` running on the local server will contact `mountd` the first time an NFS client tries to access the file system to determine whether the client should get read-write, read-only, or no access. This access can be dependent on the security mode used in the remoted procedure call from the client. See `share_nfs(1M)`.

The command also provides information as to what file systems are mounted by which clients. This information can be printed using the `showmount(1M)` command.

The `mountd` daemon is automatically invoked by `share(1M)`.

Only super user can run the `mountd` daemon.

SMF Management Since `mountd` must be running for `nfsd` to function properly, `mountd` is automatically started by the `svc:/network/nfs/server` service.

Startup SMF parameters for `mountd` can be manipulated using the `sharectl(1M)` command. The currently supported parameters are:

`server_versmax=`*num*
`server_versmin=`*num*
: The NFS server only uses NFS versions in the range specified by these variables. Valid values or versions are: 2, 3, and 4. If one or both of these parameters are left unset, the default minimum version is 2, while the default maximum version is 4.

`showmount_info={full | none}`
: If the value of this property is `none`, the following rules apply:

  - A client can see only the shares that it is allowed to access.
  - A client cannot see access lists for the shares defined at server.
  - A client cannot see remote mounts from the server done by other clients.

  If the value is `full`, these rules do not apply. The default value is `full`.

  For example, to place the restrictions specified above for users of `showmount(1M)`, enter:

```
# sharectl set -p showmount_info=none nfs
% sharectl get -p showmount_info nfs
showmount_info=none
```

Options The options shown below are supported for NFSv2/v3 clients. They are not supported for Solaris NFSv4 clients.

`-r` Reject mount requests from clients. Clients that have file systems mounted will not be affected.

`-v` Run the command in verbose mode. Each time `mountd` determines what access a client should get, it will log the result to the console, as well as how it got that result.

Files `/etc/dfs/sharetab` shared file system table

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	service/file-system/nfs

See Also `mount_nfs(1M)`, `nfsd(1M)`, `share(1M)`, `share_nfs(1M)`, `sharectl(1M)`, `showmount(1M)`, `sharetab(4)`, `attributes(5)`

Notes Some routines that compare hostnames use case-sensitive string comparisons; some do not. If an incoming request fails, verify that the case of the hostname in the file to be parsed matches the case of the hostname called for, and attempt the request again.

Name mount_hsfs – mount hsfs file systems

Synopsis

```
mount -F hsfs [generic_options]
     [-o FSType-specific_options] [-O ] special | mount_point

mount -F hsfs [generic_options]
     [-o FSType-specific_options] [-O] special mount_point
```

Description `mount` attaches an ISO 9660 filesystem (the High Sierra file system, `hsfs`, is a draft predecessor to ISO 9660, so the name reflects the filesystem's history) to the file system hierarchy at the *mount_point*, which is the pathname of a directory. If *mount_point* has any contents prior to the mount operation, these are hidden until the file system is unmounted.

If `mount` is invoked with *special* or *mount_point* as the only arguments, `mount` will search `/etc/vfstab` to fill in the missing arguments, including the *FSType-specific_options*; see `mount(1M)` for more details.

The `hsfs` file system supports direct mounting of files containing the file system as well as block devices. See `mount(1M)` and `lofiadm(1M)`.

A file system conforming to ISO 9660 can contain extensions that allow it to overcome limitations of the original ISO 9660:1988 (version 1) standard. The following types of extensions are supported by `hsfs`:

Rock Ridge extensions
: This is the preferred type of extension as it allows file attributes, name length, and types equivalent to those on other UNIX-style filesystems. Example of supported features are device special files, POSIX permissions, symbolic links, and filenames of up to 255 bytes in length. Rock Ridge extensions also remove the ISO9660:1988 restriction on maximum nesting depth for directories (eight levels). `hsfs` automatically detects the presence of Rock Ridge extensions and uses them, unless mount options are specified to disable the use of Rock Ridge or to use a different extension.

ISO9660:1999 (version 2) extensions
: The first version of ISO9660, released in 1988, supported only uppercase ASCII filenames of no more than 31 characters in length. ISO9660 version 2, released in 1999, provides an extension that allows filenames of at least 207 bytes that can use UTF-8 characters and removes the limitation on the nesting depth for directories. Unlike Rock Ridge, it does not provide support for UNIX-style file types and file attributes. `hsfs` automatically detects this extension and will use it for filename lookup if no Rock Ridge extensions are found on the media.

Joliet extensions
: The Joliet extension was devised by Microsoft to allow Unicode (UCS-2) long filenames with CDROM-based media. It allows filename lengths of up to 110 Unicode characters and does not support UNIX-style file types and attributes. `hsfs` falls back to using Joliet if such an extension is present and neither Rock Ridge nor ISO9660 version 2 extensions are found.

If filenames are longer than the 64 UCS-2 characters officially allowed by Microsoft (that is, 110 Unicode characters), they can translate to up to 330 UTF-8 octets. Filenames that translate to more than 255 UTF-8 octets will be truncated.

Options *generic_options*

See `mount(1M)` for the list of supported options.

`-o`

Specify `hsfs` file system specific options. If invalid options are specified, a warning message is printed and the invalid options are ignored. The following options are available:

`global | noglobal`

If `global` is specified and supported on the file system, and the system in question is part of a cluster, the file system will be globally visible on all nodes of the cluster. If `noglobal` is specified, the mount will not be globally visible. The default behavior is `noglobal`.

`ro`

Mount the file system read-only. This option is required.

`rr | nrr`

Enable (`rr`) or disable (`nrr`) the use of Rock Ridge. `rr` is the default and need not be specified. If you use `nrr` and Rock Ridge extensions are present in the file system, ignore them and search for other available extensions or fall back to plain ISO9660.

`vers2 | novers2`

Enable or disable the use of ISO9660 version 2 extensions. If `vers2` is specified and ISO9660 version 2 extensions are available, `hsfs` will use ISO9660 version 2 even if the file system contains the preferred Rock Ridge extensions as well. If `novers2` is specified, it will fall back to using either Joliet extensions or plain ISO9660 even if ISO9660 version 2 extensions are available.

`joliet | nojoliet`

Enable or disable the use of Joliet extensions. If `joliet` is specified and Joliet extensions are available, `hsfs` will use them even if the file system contains the preferred Rock Ridge and/or ISO9660 version 2 extensions. If `nojoliet` is specified, it will fall back to using plain ISO9660.

`notraildot`

File names on High Sierra file systems consist of a proper name and an extension separated by a '.' (dot) character. By default, the separating dot is always considered part of the file's name for all file access operations, even if there is no extension present. Specifying `notraildot` makes it optional to specify the trailing dot to access a file whose name lacks an extension.

Exceptions: This option is effective only on file systems for which Rock Ridge, ISO9660 version 2 or Joliet extensions are not active, either because they are not present on the CD-ROM, or they have been deliberately disabled via the `nrr`, `novers2` and `nojoliet` option. If either extension is active, `hsfs` quietly ignores this option.

`nomaplcase`
File names on High Sierra/ISO9660 CD–ROMs with no extensions present should be uppercase characters only. By default, `hsfs` maps file names read from a `non-Rock Ridge` disk to all lowercase characters. `nomaplcase` turns off this mapping. The exceptions for `notraildot` discussed above apply to `nomaplcase`.

`-O`
Overlay mount. Allow the file system to be mounted over an existing mount point, making the underlying file system inaccessible. If a mount is attempted on a preexisting mount point without setting this flag, the mount will fail, producing the error: `device busy`.

Examples

EXAMPLE 1 Mounting and Unmounting a DVD Image Directly

The following commands mount and unmount a DVD image.

```
# mount -F hsfs /images/solaris.iso /mnt/solaris-image
# umount /mnt/solaris-image
```

Files

`/etc/mnttab`
table of mounted file systems

`/etc/vfstab`
list of default parameters for each file system

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also

`lofiadm(1M)`, `mount(1M)`, `mountall(1M)`, `mount(2)`, `mnttab(4)`, `vfstab(4)`, `attributes(5)`

Notes

If the directory on which a file system is to be mounted is a symbolic link, the file system is mounted on the directory to which the symbolic link refers, rather than on top of the symbolic link itself.

Name mount_nfs – mount remote NFS resources

Synopsis

```
mount [-F nfs] [generic_options] [-o specific_options] [-O] resource

mount [-F nfs] [generic_options] [-o specific_options] [-O] mount_point

mount [-F nfs] [generic_options] [-o specific_options]
     [-O] resource mount_point
```

Description The `mount` utility attaches a named *resource* to the file system hierarchy at the pathname location *mount_point*, which must already exist. If *mount_point* has any contents prior to the `mount` operation, the contents remain hidden until the *resource* is once again unmounted.

`mount_nfs` starts the `lockd`(1M) and `statd`(1M) daemons if they are not already running.

If the resource is listed in the `/etc/vfstab` file, the command line can specify either *resource* or *mount_point*, and `mount` consults `/etc/vfstab` for more information. If the `-F` option is omitted, `mount` takes the file system type from `/etc/vfstab`.

If the resource is not listed in the `/etc/vfstab` file, then the command line must specify both the *resource* and the *mount_point*.

host can be an IPv4 or IPv6 address string. As IPv6 addresses already contain colons, enclose *host* in a pair of square brackets when specifying an IPv6 address string. Otherwise the first occurrence of a colon can be interpreted as the separator between the host name and path, for example, `[1080::8:800:200C:417A]:tmp/file`. See `inet`(7P) and `inet6`(7P).

host:*pathname*
: Where *host* is the name of the NFS server host, and *pathname* is the path name of the directory on the server being mounted. The path name is interpreted according to the server's path name parsing rules and is not necessarily slash-separated, though on most servers, this is the case.

nfs://*host*[:*port*]/*pathname*
: This is an NFS URL and follows the standard convention for NFS URLs as described in *NFS URL Scheme*, RFC 2224. See the discussion of URL's and the public option under `NFS FILE SYSTEMS` for a more detailed discussion.

host:*pathname* *nfs*://*host*[:*port*]/*pathname*
: *host*:*pathname* is a comma-separated list of *host*:*pathname*.

  See the discussion of replicated file systems and failover under `NFS FILE SYSTEMS` for a more detailed discussion.

hostlist pathname
: *hostlist* is a comma-separated list of hosts.

  See the discussion of replicated file systems and failover under `NFS FILE SYSTEMS` for a more detailed discussion.

The `mount` command maintains a table of mounted file systems in `/etc/mnttab`, described in `mnttab(4)`.

`mount_nfs` supports both NFSv3 and NFSv4 mounts. The default NFS version is NFSv4.

SMF Management

The NFS client service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/network/nfs/client:default
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. The service's status can be queried using the `svcs(1)` command.

Options

See `mount(1M)` for the list of supported *generic_options*. See `share_nfs(1M)` for a description of server options.

`-o` *specific_options*
Set file system specific options according to a comma-separated list with no intervening spaces.

`acdirmax=`*n*
Hold cached attributes for no more than *n* seconds after directory update. The default value is `60`.

`acdirmin=`*n*
Hold cached attributes for at least *n* seconds after directory update. The default value is `30`.

`acregmax=`*n*
Hold cached attributes for no more than *n* seconds after file modification. The default value is `60`.

`acregmin=`*n*
Hold cached attributes for at least *n* seconds after file modification. The default value is `3`.

`actimeo=`*n*
Set *min* and *max* times for regular files and directories to *n* seconds. See "File Attributes," below, for a description of the effect of setting this option to `0`.

See "Specifying Values for Attribute Cache Duration Options," below, for a description of how `acdirmax`, `acdirmin`, `acregmax`, `acregmin`, and `actimeo` are parsed on a `mount` command line.

`bg` | `fg`
If the first attempt fails, retry in the background, or, in the foreground. The default is `fg`.

`forcedirectio` | `noforcedirectio`
If `forcedirectio` is specified, then for the duration of the mount, forced direct I/O is used. If the filesystem is mounted using `forcedirectio`, data is transferred directly between client and server, with no buffering on the client. If the filesystem is mounted using `noforcedirectio`, data is buffered on the client. `forcedirectio` is a performance option that is of benefit only in large sequential data transfers. The default behavior is `noforcedirectio`.

`grpid`
By default, the GID associated with a newly created file obeys the System V semantics; that is, the GID is set to the effective GID of the calling process. This behavior can be overridden on a per-directory basis by setting the set-GID bit of the parent directory; in this case, the GID of a newly created file is set to the GID of the parent directory (see `open(2)` and `mkdir(2)`). Files created on file systems that are mounted with the `grpid` option obeys BSD semantics independent of whether the set-GID bit of the parent directory is set; that is, the GID is unconditionally inherited from that of the parent directory.

`hard` | `soft`
Continue to retry requests until the server responds (`hard`) or give up and return an error (`soft`). The default value is `hard`. Note that NFSv4 clients do not support soft mounts.

`intr` | `nointr`
Allow (do not allow) keyboard interrupts to kill a process that is hung while waiting for a response on a hard-mounted file system. The default is `intr`, which makes it possible for clients to interrupt applications that can be waiting for a remote mount.

`llock`
Use local locking (no lock manager). Note that this is a private interface.

`noac`
Suppress data and attribute caching. The data caching that is suppressed is the write-behind. The local page cache is still maintained, but data copied into it is immediately written to the server.

`nocto`
Do not perform the normal close-to-open consistency. When a file is closed, all modified data associated with the file is flushed to the server and not held on the client. When a file is opened the client sends a request to the server to validate the client's local caches. This behavior ensures a file's consistency across multiple NFS clients. When `nocto` is in effect, the client does not perform the flush on close and the request for validation, allowing the possiblity of differences among copies of the same file as stored on multiple clients.

This option can be used where it can be guaranteed that accesses to a specified file system are made from only one client and only that client. Under such a condition, the effect of `nocto` can be a slight performance gain.

`port=`*n*
The server IP port number. The default is `NFS_PORT`. If the `port` option is specified, and if the resource includes one or more NFS URLs, and if any of the URLs include a `port` number, then the `port` number in the option and in the URL must be the same.

`posix`
Request POSIX.1 semantics for the file system. Requires a mount version 2 `mountd(1M)` on the server. See `standards(5)` for information regarding POSIX.

`proto=`*netid* | `rdma`
By default, the transport protocol that the NFS mount uses is the first available RDMA transport supported both by the client and the server. If no RDMA transport is found, then it attempts to use a TCP transport or, failing that, a UDP transport, as ordered in the `/etc/netconfig` file. If it does not find a connection oriented transport, it uses the first available connectionless transport.

Use this option to override the default behavior.

`proto` is set to the value of *netid* or `rdma`. *netid* is the value of the `network_id` field entry in the `/etc/netconfig` file.

The UDP protocol is not supported for NFS version 4. If you specify a UDP protocol with the `proto` option, NFS version 4 is not used.

`public`
The `public` option forces the use of the public file handle when connecting to the NFS server. The resource specified might not have an NFS URL. See the discussion of URLs and the public option under `NFS FILE SYSTEMS` for a more detailed discussion.

`quota` | `noquota`
Enable or prevent `quota(1M)` to check whether the user is over quota on this file system; if the file system has quotas enabled on the server, quotas are still checked for operations on this file system.

`remount`
Remounts a read-only file system as read-write (using the `rw` option). This option cannot be used with other `-o` options, and this option works only on currently mounted read-only file systems.

`retrans=`*n*
Set the number of NFS retransmissions to *n*. The default value is `5`. For connection-oriented transports, this option has no effect because it is assumed that the transport performs retransmissions on behalf of NFS.

`retry=`*n*
The number of times to retry the `mount` operation. The default for the `mount` command is `10000`.

The default for the automounter is `0`, in other words, do not retry. You might find it useful to increase this value on heavily loaded servers, where automounter traffic is dropped, causing unnecessary server not responding errors.

`rsize=`*n*
Set the read buffer size to a maximum of *n* bytes. The default value is `1048576` when using connection-orientated transports with version 3 or version 4 of the NFS protocol, and `32768` when using connection-less transports. The default can be negotiated down if the server prefers a smaller transfer size. "`Read`" operations may not necessarily use the maximum buffer size. When using version 2, the default value is `32768` for all transports.

`sec=`*mode*
Set the security *mode* for NFS transactions. If `sec=` is not specified, then the default action is to use `AUTH_SYS` over NFS version 2 mounts, use a user-configured default `auth` over NFS version 3 mounts, or to negotiate a mode over version 4 mounts.

The preferred mode for NFS version 3 mounts is the default mode specified in `/etc/nfssec.conf` (see `nfssec.conf(4)`) on the client. If there is no default configured in this file or if the server does not export using the client's default mode, then the client picks the first mode that it supports in the array of modes returned by the server. These alternatives are limited to the security flavors listed in `/etc/nfssec.conf`.

NFS version 4 mounts negotiate a security mode when the server returns an array of security modes. The client attempts the mount with each security mode, in order, until one is successful.

Only one mode can be specified with the `sec=` option. See `nfssec(5)` for the available *mode* options.

`secure`
This option has been deprecated in favor of the `sec=`*dh* option.

`timeo=`*n*
Set the NFS timeout to *n* tenths of a second. The default value is `11` tenths of a second for connectionless transports, and `600` tenths of a second for connection-oriented transports. This value is ignored for connectionless transports. Such transports might implement their own timeouts, which are outside the control of NFS.

`vers=`*NFS version number*
By default, the version of NFS protocol used between the client and the server is the highest one available on both systems. The default maximum for the client is version 4. This can be changed by setting `client_versmax` to a valid version number (`2`, `3`, or `4`). Use the `sharectl(1M)` command to manipulate the `client_versmax` property. If the NFS server does not support the client's default maximum, the next lowest version attempted until a matching version is found.

`wsize=`*n*
Set the write buffer size to a maximum of *n* bytes. The default value is `1048576` when using connection-orientated transports with version 3 or version 4 of the NFS protocol, and `32768` when using connection-less transports. The default can be negotiated down if the server prefers a smaller transfer size. "`Write`" operations may not necessarily use the maximum buffer size. When using version 2, the default value is `32768` for all transports.

`xattr` | `noxattr`
Allow or disallow the creation and manipulation of extended attributes. The default is `xattr`. See `fsattr(5)` for a description of extended attributes.

`-O`
Overlay mount. Allow the file system to be mounted over an existing mount point, making the underlying file system inaccessible. If a mount is attempted on a pre-existing mount point without setting this flag, the mount fails, producing the error "device busy."

Nfs File Systems

Background versus Foreground
File systems mounted with the `bg` option indicate that `mount` is to retry in the background if the server's mount daemon (`mountd(1M)`) does not respond. `mount` retries the request up to the count specified in the `retry=`*n* option. (Note that the default value for `retry` differs between `mount` and `automount`. See the description of `retry`, above.) Once the file system is mounted, each NFS request made in the kernel waits `timeo=`*n* tenths of a second for a response. If no response arrives, the time-out is multiplied by `2` and the request is retransmitted. When the number of retransmissions has reached the number specified in the `retrans=`*n* option, a file system mounted with the `soft` option returns an error on the request; one mounted with the `hard` option prints a warning message and continues to retry the request.

Hard versus Soft
File systems that are mounted read-write or that contain executable files should always be mounted with the `hard` option. Applications using `soft` mounted file systems can incur unexpected I/O errors, file corruption, and unexpected program core dumps. The soft option is not recommended.

Authenticated requests
The server can require authenticated NFS requests from the client. `sec=`*dh* authentication might be required. See `nfssec(5)`.

URLs and the public option
If the `public` option is specified, or if the *resource* includes and NFS URL, `mount` attempts to connect to the server using the public file handle lookup protocol. See *WebNFS Client Specification*, RFC 2054. If the server supports the public file handle, the attempt is successful; `mount` does not need to contact the server's `rpcbind(1M)` and the `mountd(1M)` daemons to get the port number of the `mount` server and the initial file handle of *pathname*, respectively. If the NFS client and server are separated by a firewall that allows all outbound connections through specific ports, such as `NFS_PORT`, then this enables NFS operations

through the firewall. The public option and the NFS URL can be specified independently or together. They interact as specified in the following matrix:

	Resource Style	
	host:*pathname*	NFS URL
public option	Force public file handle and fail mount if not supported. Use Native paths.	Force public file handle and fail mount if not supported. Use Canonical paths.
default	Use MOUNT protocol.	Try public file handle with Canonical paths. Fall back to MOUNT protocol if not supported.

A Native path is a path name that is interpreted according to conventions used on the native operating system of the NFS server. A Canonical path is a path name that is interpreted according to the URL rules. See *Uniform Resource Locators (URL)*, RFC 1738. See "Examples," below, for uses of Native and Canonical paths.

Replicated file systems and failover

resource can list multiple read–only file systems to be used to provide data. These file systems should contain equivalent directory structures and identical files. The file systems can be specified either with a comma–separated list of *host:/pathname* entries and/or NFS URL entries, or with a comma –separated list of hosts, if all file system names are the same. If multiple file systems are named and the first server in the list is down, failover uses the next alternate server to access files. If the read–only option is not chosen, replication is disabled. File access, for NFS versions 2 and 3, is blocked on the original if NFS locks are active for that file.

File Attributes

To improve NFS read performance, files and file attributes are cached. File modification times get updated whenever a write occurs. However, file access times can be temporarily out-of-date until the cache gets refreshed.

The attribute cache retains file attributes on the client. Attributes for a file are assigned a time to be flushed. If the file is modified before the flush time, then the flush time is extended by the time since the last modification (under the assumption that files that changed recently are likely to change soon). There is a minimum and maximum flush time extension for regular files and for directories. Setting `actimeo=`*n* sets flush time to *n* seconds for both regular files and directories.

Setting `actimeo=0` disables attribute caching on the client. This means that every reference to attributes is satisfied directly from the server though file data is still cached. While this

guarantees that the client always has the latest file attributes from the server, it has an adverse effect on performance through additional latency, network load, and server load.

Setting the `noac` option also disables attribute caching, but has the further effect of disabling client write caching. While this guarantees that data written by an application is written directly to a server, where it can be viewed immediately by other clients, it has a significant adverse effect on client write performance. Data written into memory-mapped file pages (`mmap(2)`) are not written directly to this server.

Specifying Values for Attribute Cache Duration Options

The attribute cache duration options are `acdirmax`, `acdirmin`, `acregmax`, `acregmin`, and `actimeo`, as described under OPTIONS. A value specified for `actimeo` sets the values of all attribute cache duration options except for any of these options specified following `actimeo` on a `mount` command line. For example, consider the following command:

```
example# mount -o acdirmax=10,actimeo=1000 server:/path /localpath
```

Because `actimeo` is the last duration option in the command line, its value (`1000`) becomes the setting for all of the duration options, including `acdirmax`. Now consider:

```
example# mount -o actimeo=1000,acdirmax=10 server:/path /localpath
```

Because the `acdirmax` option follows `actimeo` on the command line, it is assigned the value specified (`10`). The remaining duration options are set to the value of `actimeo` (`1000`).

Examples

EXAMPLE 1 Mounting an NFS File System

To mount an NFS file system:

```
example# mount serv:/usr/src /usr/src
```

This is an example of the use of a native path.

EXAMPLE 2 Mounting An NFS File System Read-Only With No suid Privileges

To mount an NFS file system read-only with no suid privileges:

```
example# mount -r -o nosuid serv:/usr/src /usr/src
```

EXAMPLE 3 Mounting An NFS File System Over Version 2, with the UDP Transport

To mount an NFS file system over version 2, with the UDP transport:

```
example# mount -o vers=2,proto=udp serv:/usr/src /usr/src
```

EXAMPLE 4 Mounting an NFS File System Using An NFS URL

To mount an NFS file system using an NFS URL (a canonical path):

```
example# mount nfs://serv/usr/man /usr/man
```

EXAMPLE 5 Mounting An NFS File System Forcing Use Of The Public File Handle

To mount an NFS file system and force the use of the public file handle and an NFS URL (a canonical path) that has a non 7–bit ASCII escape sequence:

```
example# mount -o public nfs://serv/usr/%A0abc /mnt/test
```

EXAMPLE 6 Mounting an NFS File System Using a Native Path

To mount an NFS file system using a native path (where the server uses colons (":") as the component separator) and the public file handle:

```
example# mount -o public serv:C:doc:new /usr/doc
```

EXAMPLE 7 Mounting a Replicated Set of NFS File Systems with the Same Pathnames

To mount a replicated set of NFS file systems with the same pathnames:

```
example# mount serv—a,serv—b,serv—c:/usr/man /usr/man
```

EXAMPLE 8 Mounting a Replicated Set of NFS File Systems with Different Pathnames

To mount a replicated set of NFS file systems with different pathnames:

```
example# mount serv—x:/usr/man,serv—y:/var/man,nfs://serv-z/man /usr/man
```

Files

`/etc/mnttab`
table of mounted file systems

`/etc/dfs/fstypes`
default distributed file system type

`/etc/vfstab`
table of automatically mounted resources

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/file-system/nfs

See Also

`lockd(1M)`, `mountall(1M)`, `mountd(1M)`, `nfsd(1M)`, `quota(1M)`, `sharectl(1M)`, `statd(1M)`, `mkdir(2)`, `mmap(2)`, `mount(2)`, `open(2)`, `umount(2)`, `mnttab(4)`, `nfssec.conf(4)`, `attributes(5)`, `fsattr(5)`, `nfssec(5)`, `standards(5)`, `inet(7P)`, `inet6(7P)`, `lofs(7FS)`

Callaghan, Brent, *WebNFS Client Specification*, RFC 2054, October 1996.

Callaghan, Brent, *NFS URL Scheme*, RFC 2224, October 1997.

Berners-Lee, Masinter & McCahill , *Uniform Resource Locators (URL)*, RFC 1738, December 1994.

Notes An NFS server should not attempt to mount its own file systems. See `lofs`(7FS).

If the directory on which a file system is to be mounted is a symbolic link, the file system is mounted on *the directory to which the symbolic link refers,* rather than being mounted on top of the symbolic link itself.

SunOS 4.x used the `biod` maintenance procedure to perform parallel read-ahead and write-behind on NFS clients. SunOS 5.x made `biod` obsolete with multi-threaded processing, which transparently performs parallel read-ahead and write-behind.

Since the root (`/`) file system is mounted read-only by the kernel during the boot process, only the `remount` option (and options that can be used in conjunction with `remount`) affect the root (`/`) entry in the `/etc/vfstab` file.

Name mount_pcfs – mount pcfs file systems

Synopsis

```
mount -F pcfs [generic_options]
     [-o FSType-specific_options] special | mount_point

mount -F pcfs [generic_options]
     [-o FSType-specific_options] special mount_point
```

Description `mount` attaches an MS-DOS file system (`pcfs`) to the file system hierarchy at the *mount_point*, which is the pathname of a directory. If *mount_point* has any contents prior to the `mount` operation, these are hidden until the file system is unmounted.

The `pcfs` file system supports direct mounting of files containing the file system as well as block devices. See `mount(1M)` and `lofiadm(1M)`.

If `mount` is invoked with *special* or *mount_point* as the only arguments, `mount` will search `/etc/vfstab` to fill in the missing arguments, including the *FSType-specific_options*; see `mount(1M)` for more details.

The *special* argument is a special device file type, which is:

A DOS logical drive on a hard disk expressed as *device-name*:*logical-drive* , where *device-name* specifies the special block device-file for the whole disk and *logical-drive* is either a drive letter (c through z) or a drive number (1 through 24). Examples are `/dev/dsk/c0t0d0p0:`*c* and `/dev/dsk/c0t0d0p0:`*1*.

The *special* device file type must have a formatted MS-DOS file system with either a 12-bit, 16-bit, or 32-bit File Allocation Table.

Options *generic_options*

See `mount(1M)` for the list of supported options.

`-o`

Specify `pcfs` file system-specific options. The following options are supported:

`clamptime` | `noclamptime`

File timestamps in `pcfs` cover a range between January 1st 1980 and December 31st 2127. This is not equal to the range of `time_t` on Unix for either 32-bit or 64-bit applications. In particular, 32-bit applications fail with `EOVERFLOW` errors on the `stat(2)` system call when timestamps beyond the range of 32-bit `time_t` are encountered. In order to prevent such spurious failures, `pcfs` by default clamps timestamps to the common subset of possible `pcfs` timestamps and the range available to 32-bit applications in Unix. The `clamptime` mount option therefore is active by default. If you want access to the full range of possible timestamps on `pcfs`, mount the file system with the `noclamptime` mount option. Note that if `noclamptime` is used, only 64-bit applications will have access to timestamps beyond January 19th 2038, 03:14:06 UTC; 32-bit applications will encounter `EOVERFLOW` errors.

`foldcase` | `nofoldcase`
Force uppercase characters in filenames to lowercase when reading them from the filesystem. This is for compatibility with the previous behavior of `pcfs`. The default is `nofoldcase`.

`hidden` | `nohidden`
Allow or disallow listing of files with hidden or system bits set. Option `hidden` is the default. When `nohidden` is effect, hidden and system files are neither visible nor accessible. Note that PCFS in previous releases of the Solaris operating system used the `nohidden` option as the default.

`atime` | `noatime`
Enable or disable write access timestamps on DOS-formatted media. Default for fixed disks is `atime`, while for removable media `noatime` is used. The latter default is so that writes to flash-based media ("memory sticks") can be minimized, to prolong lifetime.

`timezone=`*timezone*
Timestamps on DOS-formatted media are recorded in the local time of the recording system. This can cause confusion when accessing removable media in which the recording and receiving system use different time zones. Use this option to force media timestamps to be interpreted for a specific time zone. The `mount_pcfs` command converts the given time zone name into a numerical offset that is passed to the `pcfs` kernel module, using the same rules as described in `environ(5)` for the `TZ` environment variable. By default, the `timezone` value is taken from the `TZ` environment variable.

`owner=`{*username*|*user_id*}
Specifies the owner for the contents of the file system. The value can either be a character-string for a user name, or an integer value for the user ID. The default owner is the user ID of current process.

`group=`{*groupname*|*group_id*}
Specifies the group ID for the contents of the file systems. The value can either be a character-string for a group name, or an integer value for the group ID. The default group is the group ID of the mounting process.

`mask=`*user_mask*
Specifies the mask used for setting maximum access permissions on contents of the file system. The *user_mask* is an octal value, a bitmask, where each bit represents the permissions that should be disabled. The default *user_mask* is the file mode creation mask of the mounting process.

See `Intro(2)` for more information on masks.

Files `/etc/mnttab`
table of mounted file systems

`/etc/vfstab`
list of default parameters for each file system

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/file-system/pcfs

See Also lofiadm(1M), mount(1M), mountall(1M), Intro(2), mount(2), stat(2), time(2), mnttab(4), vfstab(4), attributes(5), environ(5), pcfs(7FS)

Notes If the directory on which a file system is to be mounted is a symbolic link, the file system is mounted on the directory to which the symbolic link refers, rather than on top of the symbolic link itself.

Name mount_smbfs, umount_smbfs – mount and unmount a shared resource from an SMB file server

Synopsis

```
/usr/sbin/mount [-F smbfs] [generic-options] [-o name=value]
     [-O] resource

/usr/sbin/mount [-F smbfs] [generic-options] [-o name=value]
     [-O] mount-point

/usr/sbin/mount [-F smbfs] [generic-options] [-o name=value]
     [-O] resource mount-point

/usr/sbin/umount [-F smbfs] [generic-options] mount-point
```

Description The `mount` utility attaches a named resource, *resource*, to the file system hierarchy at the path name location, *mount-point*, which must already exist.

If *mount-point* has any contents prior to the `mount` operation, those contents remain hidden until the resource is unmounted. An authorized user with the `SYS_MOUNT` privilege can perform a `mount` operation. Also, a user can perform SMBFS mount operations on a directory the user owns.

If the resource is listed in the `/etc/vfstab` file, you can specify either *resource* or *mount-point* as the `mount` command will consult the `/etc/vfstab` file for more information. If the `-F` option is omitted, `mount` takes the file system type from the entry in the `/etc/vfstab` file.

If the resource is not listed in the `/etc/vfstab` file, the command line must specify both *resource* and *mount-point*.

The `umount` utility detaches a mounted file system from the file system hierarchy. An authorized user with the `SYS_MOUNT` privilege can perform a `umount` operation. Also, a user can perform SMBFS unmount operations on a directory the user owns.

The `network/smb/client` service must be enabled to successfully mount an SMB share. This service is enabled by default.

To enable the service, enter the following `svcadm(1M)` command:

```
# svcadm enable network/smb/client
```

Operands The `mount` command supports the following operands:

resource //*server*/*share*

The name of the resource to be mounted. In addition to its name, you can specify the following information about the resource:

- *server* is the DNS or NetBIOS name of the remote computer.
- *share* is the resource name on the remote server.

You can also specify the user account. See the “Options” section.

The `mount` command can read a password from standard input for the user account. If the password is not provided, `mount` first attempts to use the password stored by the `smbadm add-key` command (if any). If that password fails to authenticate, the `mount_smbfs` command prompts you for a password if standard input is a TTY.

mount-point
The path to the location where the file system is to be mounted or unmounted. The `mount` command maintains a table of mounted file systems in the `/etc/mnttab` file. See the `mnttab(4)` man page.

Options See the `mount(1M)` man page for the list of supported *generic-options*.

`-o` *name*=*value* or
`-o` *name*
Sets the file system-specific properties. You can specify more than one name-value pair as a list of comma-separated pairs. No spaces are permitted in the list. The properties are as follows:

`acdirmax`=*n*
Hold cached attributes for no more than *n* seconds after directory update. The default value is `60`.

`acdirmin`=*n*
Hold cached attributes for at least *n* seconds after directory update. The default value is `30`.

`acregmax`=*n*
Hold cached attributes for no more than *n* seconds after file modification. The default value is `60`.

`acregmin`=*n*
Hold cached attributes for at least *n* seconds after file modification. The default value is `3`.

`actimeo`=*n*
Set minimum and maximum times for regular files and directories to *n* seconds. See "File Attributes," below, for a description of the effect of setting this option to `0`.

See "Specifying Values for Attribute Cache Duration Options," below, for a description of how `acdirmax`, `acdirmin`, `acregmax`, `acregmin`, and `actimeo` are parsed on a `mount` command line.

`dirperms`=*octaltriplet*
Specifies the permissions to be assigned to directories. The value must be specified as an octal triplet, such as `755`. The default value for the directory mode is taken from the `fileperms` setting, with execute permission added where `fileperms` has read permission.

Note that these permissions have no relation to the rights granted by the SMB server.

domain=value
Specifies the name of the workgroup or the Windows domain in which the user name is defined. If the domain name is not specified, the default system's SMB domain is used.

`fileperms=`*octaltriplet*
Specifies the permissions to be assigned to files. The value must be specified as an octal triplet, such as `644`. The default value is `700`.

Note that these permissions have no relation to the rights granted by the SMB server.

`gid=`*groupid*
Assigns the specified group ID to files. The default value is the group ID of the directory where the volume is mounted.

`intr|nointr`
Enable (or disable) cancellation of `smbfs(7FS)` I/O operations when the user interrupts the calling thread (for example, by hitting Ctrl-C while an operation is underway). The default is `intr` (interruption enabled), so cancellation is normally allowed.

`noac`
Suppress attribute caching. Local `stat(2)` calls always request attributes from the SMB server.

`noprompt`
Suppresses the prompting for a password when mounting a share. This property enables you to permit anonymous access to a share. Anonymous access does not require a password.

The `mount` operation fails if a password is required, the `noprompt` property is set, and no password is stored by the `smbadm add-key` command.

`uid=`*userid*
Assigns the specified user ID files. The default value is the owner ID of the directory where the volume is mounted.

user=value
Specifies the remote user name. If *user* is omitted, the logged-in user ID is used.

`xattr|noxattr`
Enable (or disable) Solaris Extended Attributes in this mount point. This option defaults to `xattr` (enabled Extended Attributes), but note: if the SMB server does not support SMB "named streams", `smbfs(7FS)` forces this option to `noxattr`. When a mount has the `noxattr` option, attempts to use Solaris Extended attributes fail with EINVAL.

`-O`
Overlays mount. Allow the file system to be mounted over an existing mount point, making the underlying file system inaccessible. If a mount is attempted on a pre-existing mount point without setting this flag, the mount fails, producing the error "device busy."

File Attributes

To improve `smbfs` performance, file attributes are cached. File modification times get updated whenever any local modifications occur. However, file access times can be temporarily out-of-date until the cache gets refreshed.

The attribute cache retains file attributes on the client. Attributes for a file are assigned a time to be flushed. If the file is modified before the flush time, then the flush time is extended by the time since the last modification (under the assumption that files that changed recently are likely to change soon). There is a minimum and maximum flush time extension for regular files and for directories. Setting `actimeo=`*n* sets flush time to *n* seconds for both regular files and directories.

Setting `actimeo=`*n* disables attribute caching on the client. This means that every reference to attributes is satisfied directly from the server. While this guarantees that the client always has the latest file attributes from the server, it has an adverse effect on performance through additional latency, network load, and server load.

Setting the `noac` option also disables attribute caching. When `smbfs` is enhanced to support write caching, this option will have the further effect of disabling that write caching.

Specifying Values for Attribute Cache Duration Options

The attribute cache duration options are `acdirmax`, `acdirmin`, `acregmax`, `acregmin`, and `actimeo`, as described under `OPTIONS`, above. A value specified for `actimeo` sets the values of all attribute cache duration options except for any of these options specified following `actimeo` on a `mount` command line. For example, consider the following command:

```
# mount -F smbfs -o acdirmax=10,actimeo=1000 \
     //server/share  /mountpoint
```

Because `actimeo` is the last duration option in the command line, its value (`1000`) becomes the setting for all of the duration options, including `acdirmax`. Now consider:

```
# mount -F smbfs -o actimeo=1000,acdirmax=10 \
     //server/share  /mountpoint
```

Because the `acdirmax` option follows `actimeo` on the command line, it is assigned the value specified (`10`). The remaining duration options are set to the value of `actimeo` (`1000`).

Examples

EXAMPLE 1 Mounting an SMBFS Share

The following example shows how to mount the `/tmp` share from the `nano` server in the `SALES` workgroup on the local `/mnt` mount point. You must supply the password for the `root` user to successfully perform the mount operation.

```
# mount -F smbfs -o user=root,domain=SALES //nano.sfbay/tmp /mnt
Password:
```

EXAMPLE 2 Verifying That an SMBFS File System Is Mounted

The following example shows how to mount the `/tmp` share from the `nano` server on the local `/mnt` mount point. You must supply the password for the `root` user to successfully perform the mount operation.

EXAMPLE 2 Verifying That an SMBFS File System Is Mounted *(Continued)*

```
# mount -F smbfs -o user=root //nano.sfbay/tmp /mnt
Password:
```

You can verify that the share is mounted in the following ways:

- View the file system entry in the `/etc/mnttab` file.

```
# grep mnt /etc/mnttab
//nano.sfbay/tmp   /mnt    smbfs   dev=4900000     1177097833
```

- View the output of the `mount` command.

```
# mount | grep mnt
mnt on //nano.sfbay/tmp read/write/setuid/devices/dev=4900000 on
Tue Apr 20 13:37:13 2010
```

- View the output of the `df /mnt` command.

```
# df /mnt
/mnt            (//nano.sfbay/tmp): 3635872 blocks     -1 files
```

Obtain information about the mounted share by viewing the output of the `df -k /mnt` command.

```
# df -k /mnt
Filesystem            kbytes    used   avail capacity  Mounted on
//nano.sfbay/tmp
                      1882384   64448 1817936     4%    /mnt
```

EXAMPLE 3 Unmounting an SMB Share

This example assumes that an SMB share has been mounted on the `/mnt` mount point. The following command line unmounts the share from the mount point.

```
# umount /mnt
```

Files

`/etc/mnttab`
: Table of mounted file systems.

`/etc/dfs/fstypes`
: Default distributed file system type.

`/etc/vfstab`
: Table of automatically mounted resources.

`/var/smb/smbfspasswd`
: Stores per-user settings for the Solaris SMB client.

Attributes

See the attributes(5) man page for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/file-system/smb
Interface Stability	Committed

See Also mount(1M), mountall(1M), smbadm(1M), svcadm(1M), acl(2), fcntl(2), link(2), mknod(2), mount(2), stat(2), symlink(2), umount(2), mnttab(4), smb(4), vfstab(4), attributes(5), pcfs(7FS), smbfs(7FS)

Notes The Solaris SMB client always attempts to use `gethostbyname()` to resolve host names. If the host name cannot be resolved, the SMB client uses NetBIOS name resolution (NBNS). The Solaris SMB client permits the use of NBNS to enable Solaris SMB clients in Windows environments to work without additional configuration.

If the directory on which a file system is to be mounted is a symbolic link, the file system is mounted on the directory to which the symbolic link refers, rather than being mounted on top of the symbolic link itself.

Name mount_tmpfs – mount tmpfs file systems

Synopsis `mount [-F tmpfs] [-o` *specific_options*`] [-O]` *special mount_point*

Description `tmpfs` is a memory based file system which uses kernel resources relating to the VM system and page cache as a file system.

`mount` attaches a `tmpfs` file system to the file system hierarchy at the pathname location *mount_point*, which must already exist. If *mount_point* has any contents prior to the `mount` operation, these remain hidden until the file system is once again unmounted. The attributes (mode, owner, and group) of the root of the `tmpfs` filesystem are inherited from the underlying *mount_point*, provided that those attributes are determinable. If not, the root's attributes are set to their default values.

The *special* argument is usually specified as `swap` but is in fact disregarded and assumed to be the virtual memory resources within the system.

Options

`-o` *specific_options* Specify `tmpfs` file system specific options in a comma-separated list with no intervening spaces. If invalid options are specified, a warning message is printed and the invalid options are ignored. The following options are available:

size=*sz* The *sz* argument controls the size of this particular `tmpfs` file system. If the argument is has a 'k' suffix, the number will be interpreted as a number of kilobytes. An 'm' suffix will be interpreted as a number of megabytes. No suffix is interpreted as bytes. In all cases, the actual size of the file system is the number of bytes specified, rounded up to the physical pagesize of the system.

`xattr` | `noxattr` Allow or disallow the creation and manipulation of extended attributes. The default is `xattr`. See `fsattr(5)` for a description of extended attributes.

`-O` Overlay mount. Allow the file system to be mounted over an existing mount point, making the underlying file system inaccessible. If a mount is attempted on a pre-existing mount point without setting this flag, the mount will fail, producing the error`device busy`.

Files `/etc/mnttab` Table of mounted file systems

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also mount(1M), mkdir(2), mount(2), open(2), umount(2), mnttab(4), attributes(5), fsattr(5), tmpfs(7FS)

Notes If the directory on which a file system is to be mounted is a symbolic link, the file system is mounted on the directory to which the symbolic link refers, rather than on top of the symbolic link itself.

Name mount_udfs – mount a udfs file system

Synopsis

```
mount -F udfs [generic_options] [-o specific_options]
     [-O] special mount_point

mount -F udfs [generic_options] [-o specific_options]
     [-O] special | mount_point
```

Description The `mount` utility attaches a `udfs` file system to the file system hierarchy at the *mount_point*, which is the pathname of a directory. If *mount_point* has any contents prior to the mount operation, these are hidden until the file system is unmounted.

If mount is invoked with either *special* or *mount_point* as the only arguments, `mount` searches `/etc/vfstab` to fill in the missing arguments, including the *specific_options*. See `mount(1M)`.

The `udfs` file system supports direct mounting of files containing the file system as well as block devices. See `mount(1M)` and `lofiadm(1M)`.

If *special* and *mount_point* are specified without any *specific_options*, the default is `rw`.

If the directory on which a file system is to be mounted is a symbolic link, the file system is mounted on the directory to which the symbolic link refers, rather than on top of the symbolic link itself.

Options See `mount`(1M) for the list of supported *generic_options*.

The following options are supported:

`-o` *specific_options*
Specify `udfs` file system specific options in a comma-separated list with no intervening spaces. The following *specific_options* are available:

m
Mount the file system without making an entry in `/etc/mnttab`.

remount
Remount the file system as read-write. The option is used in conjunction with the `rw` option.

A file system mounted read-only can be remounted as read-write. This option fails if the file system is not currently mounted.

`-O`
Overlay mount. Allow the file system to be mounted over an existing mount point, making the underlying file system inaccessible. If a mount is attempted on a pre-existing mount point without setting this flag, the mount fails, producing the error `device busy`.

Files

`/etc/mnttab`
Table of mounted file systems

`/etc/vfstab`
List of default parameters for each file system

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/file-system/udfs

See Also

`fsck(1M)`, `fsck_udfs(1M)`, `lofiadm(1M)`, `mount(1M)`, `mountall(1M)`, `mount(2)`, `mnttab(4)`, `vfstab(4)`, `attributes(5)`

Diagnostics

not super user
The command is run by a non-root user. Run as root.

no such device
The device name specified does not exist.

not a directory
The specified mount point is not a directory.

is not an udfs file system
The device specified does not contain a udf 1.50 file system or the udfs file system module is not available.

is already mounted
The specified device is already in use.

not a block device
The device specified is not a block device. Use block device to mount.

write-protected
The device is read-only.

is corrupted. needs checking
The file system is in an inconsistent state. Run `fsck`.

Notes

Copy-protected files can be stored on DVD-ROM media using Universal Disk Format (UDF). Reading these copy-protected files is not possible as this involves an authentication process. Unless an authentication process between the host and the drive is completed, reading these copy-protected files after mounting and before the authentication process, returns an error.

Name mount_ufs – mount ufs file systems

Synopsis

```
mount -F ufs [generic_options] [-o specific_options]
     [-O] special | mount_point

mount -F ufs [generic_options] [-o specific_options]
     [-O] special mount_point
```

Description The `mount` utility attaches a `ufs` file system to the file system hierarchy at the *mount_point*, which is the pathname of a directory. If *mount_point* has any contents prior to the `mount` operation, these are hidden until the file system is unmounted.

The `ufs` file system supports direct mounting of files containing the file system as well as block devices. See `mount(1M)` and `lofiadm(1M)`.

If `mount` is invoked with *special* or *mount_point* as the only arguments, `mount` will search `/etc/vfstab` to fill in the missing arguments, including the *specific_options*. See `mount(1M)`.

If *special* and *mount_point* are specified without any *specific_options*, the default is `rw`.

If the directory on which a file system is to be mounted is a symbolic link, the file system is mounted on the directory to which the symbolic link refers, rather than on top of the symbolic link itself.

Options See `mount(1M)` for the list of supported *generic_options*.

The following options are supported:

`-o` *specific_options*
Specify `ufs` file system specific options in a comma-separated list with no intervening spaces. If invalid options are specified, a warning message is printed and the invalid options are ignored. The following options are available:

`dfratime` | `nodfratime`
By default, writing access time updates to the disk may be deferred (`dfratime`) for the file system until the disk is accessed for a reason other than updating access times. `nodfratime` disables this behavior.

If power management is enabled on the system, do not set `nodfratime` unless `noatime` is also set. If you set `nodfratime` without setting `noatime`, the disk is spun up every time a file within a file system on the disk is accessed - even if the file is not modified.

`forcedirectio` | `noforcedirectio`
If `forcedirectio` is specified and supported by the file system, then for the duration of the mount, forced direct I/O will be used. If the filesystem is mounted using `forcedirectio`, data is transferred directly between user address space and the disk. If the filesystem is mounted using `noforcedirectio`, data is buffered in kernel address space when data is transferred between user address space and the disk. `forcedirectio` is a performance option that is of benefit only in large sequential data transfers. The default behavior is `noforcedirectio`.

`global` | `noglobal`
If `global` is specified and supported on the file system, and the system in question is part of a cluster, the file system will be globally visible on all nodes of the cluster. If `noglobal` is specified, the mount will not be globally visible. The default behavior is `noglobal`.

`intr` | `nointr`
Allow (do not allow) keyboard interrupts to kill a process that is waiting for an operation on a locked file system. The default is `intr`.

`largefiles` | `nolargefiles`
If `nolargefiles` is specified and supported by the file system, then for the duration of the mount it is guaranteed that all regular files in the file system have a size that will fit in the smallest object of type `off_t` supported by the system performing the mount. The mount will fail if there are any files in the file system not meeting this criterion. If `largefiles` is specified, there is no such guarantee. The default behavior is `largefiles`.

If `nolargefiles` is specified, `mount` will fail for `ufs` if the file system to be mounted has contained a large file (a file whose size is greater than or equal to 2 Gbyte) since the last invocation of `fsck` on the file system. The large file need not be present in the file system at the time of the mount for the mount to fail; it could have been created previously and destroyed. Invoking `fsck` (see `fsck_ufs(1M)`) on the file system will reset the file system state if no large files are present. After invoking `fsck`, a successful mount of the file system with `nolargefiles` specified indicates the absence of large files in the file system; an unsuccessful mount attempt indicates the presence of at least one large file.

`logging` | `nologging`
If `logging` is specified, then logging is enabled for the duration of the mounted file system. Logging is the process of storing transactions (changes that make up a complete UFS operation) in a log before the transactions are applied to the file system. Once a transaction is stored, the transaction can be applied to the file system later. This prevents file systems from becoming inconsistent, therefore reducing the possibility that `fsck` might run. And, if `fsck` is bypassed, logging generally reduces the time required to reboot a system.

The default behavior is `logging` for all UFS file systems.

The log is allocated from free blocks in the file system, and is sized approximately 1 Mbyte per 1 Gbyte of file system, up to a maximum of 256 Mbytes. The log size may be larger (up to a maximum of 512 Mbytes) dependent upon the number of cylinder groups present in the file system.

Logging is enabled on any UFS file system, including root (`/`), except under the following conditions:

- When logging is specifically disabled.
- If there is insufficient file system space for the log. In this case, the following message is displayed and file system is still mounted:

```
# mount /dev/dsk/c0t4d0s0 /mnt
  /mnt: No space left on device
  Could not enable logging for /mnt on  /dev/dsk/c0t4d0s0.
```

The log created by UFS logging is continually flushed as it fills up. The log is totally flushed when the file system is unmounted or as a result of the `lockfs -f` command.

`m`
Mount the file system without making an entry in `/etc/mnttab`.

`noatime`
By default, the file system is mounted with normal access time (`atime`) recording. If `noatime` is specified, the file system will ignore access time updates on files, except when they coincide with updates to the `ctime` or `mtime`. See `stat(2)`. This option reduces disk activity on file systems where access times are unimportant (for example, a Usenet news spool).

`noatime` turns off access time recording regardless of `dfratime` or `nodfratime`.

The POSIX standard requires that access times be marked on files. `-noatime` ignores them unless the file is also modified.

`nosec`
By default, Access Control Lists (ACLs) are supported on a mounted UFS file system. Use this option to disallow the setting or any modification of an ACL on a file within a mounted UFS file system. See `getfacl(1)` for background on ACLs.

`onerror` = *action*
This option specifies the action that UFS should take to recover from an internal inconsistency on a file system. Specify *action* as `panic`, `lock`, or `umount`. These values cause a forced system shutdown, a file system lock to be applied to the file system, or the file system to be forcibly unmounted, respectively. The default is `panic`.

`quota`
Quotas are turned on for the file system.

`remount`
Remounts a file system with a new set of options. All options not explicitly set with `remount` revert to their default values.

`rq`
Read-write with quotas turned on. Equivalent to `rw, quota`.

`xattr | noxattr`
Allow or disallow the creation and manipulation of extended attributes. The default is `xattr`. See `fsattr(5)` for a description of extended attributes.

`-O`
Overlay mount. Allow the file system to be mounted over an existing mount point, making the underlying file system inaccessible. If a mount is attempted on a pre-existing mount

point without setting this flag, the mount will fail, producing the error “device busy”.

Examples

EXAMPLE 1 Turning Off (and On) Logging

The following command turns off logging on an already mounted file system. The subsequent command restores logging.

```
# mount -F ufs -o remount,nologging /export
# (absence of message indicates success)
# mount -F ufs -o remount,logging /export
```

In the preceding commands, the `-F ufs` option is not necessary.

Files

`/etc/mnttab`
table of mounted file systems

`/etc/vfstab`
list of default parameters for each file system

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also

`getfacl(1)`, `fsck(1M)`, `fsck_ufs(1M)`, `lofiadm(1M)`, `mount(1M)`, `mountall(1M)`, `fcntl(2)`, `mount(2)`, `stat(2)`, `mnttab(4)`, `vfstab(4)`, `attributes(5)`, `fsattr(5)`, `largefile(5)`

Notes

Since the root (`/`) file system is mounted read-only by the kernel during the boot process, only the `remount` option (and options that can be used in conjunction with `remount`) affect the root (`/`) entry in the `/etc/vfstab` file.

Name mpathadm – multipath discovery and administration

Synopsis `mpathadm` *subcommand direct-object* [*options*] [*operand*]

Description The `mpathadm` command enables multipathing discovery and management. The `mpathadm` command is implemented as a set of subcommands, many with their own options, that are described in the section for that subcommand. Options not associated with a particular subcommand are described under OPTIONS. The `mpathadm` subcommands operate on a *direct-object*. These are described in this section for each subcommand. The *direct-objects*, *initiator-port*, *target-port*, and*logical-unit* in the subcommands are consistent with SCSI standard definitions.

The `mpathadm` command supports the following subcommands, which are described in detail in subsections that follow.

`list`	Display a list of discovered instances for a given object.
`show`	Display information about a given object instance.
`modify`	Modify properties of an object.
`enable`	Enable an object.
`disable`	Disable an object.
`failover`	Cause target port group failover for a logical-unit.
`override`	Set a path to be used over other paths on a logical-unit.

The `mpathadm` subcommands operate on a *direct-object*. These are described in this section for each subcommand.

list Subcommand The syntax for the `list` subcommand is:

```
# mpathadm list direct-object [operands...]
```

The `list` subcommand displays data for following direct-objects:

`mpath-support` [*mpath-support-name*, ...]
List the multipathing support that can be administered by this CLI. This presents itself in the form of a library name registered through the MPAPI framework. If no mpath-support name *mpath-support-name* is specified, all registered multipathing support libraries will be displayed.

`initiator-port` [*initiator-port-name*, ...]
List the initiator ports that are discovered on this system. If no *initiator-port-name* is specified, all discovered initiator ports are displayed.

{`logical-unit` | `lu`} [*options*] [*logical-unit-name*, ...]
List the information on multipath logical units. If no *logical-unit-name* is specified, all discovered logical-units will be displayed.

Options for `list logical-unit` are as follows:

`-n`, `--name` *name*
: Return the logical unit name that is associated with the given name string. This name can be extracted from the output of the `mpathadm show lu` command.

`-t`, `--target-port` *target-port-name*
: Return the list of logical units names that are associated with the given *target-port-name*.

show Subcommand

The syntax for the `show` subcommand is:

```
# mpathadm show direct-object [operands...]
```

The `show` subcommand displays detailed information for following the direct-objects:

`mpath-support` [*mpath-support-name*, ...]
: Show the detailed information on the given *mpath-support-name* if the name exists. If the given *mpath-support-name* supports only a limited set of device products, the list of device products will be listed as part of the output.

`initiator-port` *initiator-port-name*[,*initiator-port-name*, ...]
: Show the detailed information for the given *initiator-port-name*.

{`logical-unit` | `lu`} [*logical-unit-name*, ...]
: Display the detailed information on multipath logical unit(s), including path and target port group information. Note that the name property in the logical unit information represents the identifier for this LUN, derived from the hardware, and used by this system. If the name is derived from SCSI Inquiry Vital Product Data (VPD) page 83h, the name type property represents an associated identifier type defined by the SCSI standards.

modify Subcommand

The syntax for the `modify` subcommand is:

```
# mpathadm modify direct-object [options] [operands...]
```

The `modify` subcommand modifies characteristics of the following direct-objects:

`mpath-support` [*options*] *mpath-support-name*, ...
: Configuration management of an *mpath-support*. Options to modify mpath-support are as follows:

  Set `autofailback` [`enable`|`disable`]
  : Applicable only when mpath-support provides auto failback support.

  Set `autoprobing` [`enable`|`disable`]
  : Applicable only when mpath-support provides auto probing support.

  Change the default load-balance type.
  : The `loadbalance` type is one of the

supported types listed in the `show mpath-support` output.

{`logical-unit` | `lu`} [*options*] *logical-unit-name*, ... — Configuration management of a logical unit. Options to modify `logical-unit` are as follows:

Set auto failback. Applicable only when `mpath-support` provides auto failback support

Set auto probing. Applicable only when `mpath-support` provides auto probing support.

Set load balance type for the logical unit. Applicable only when load balance configuration is supported at the logical unit level.

enable Subcommand

The syntax for the `enable` subcommand is:

```
# mpathadm enable [options]
```

The `enable` subcommand supports the following direct-objects to be enabled:

`path -i` *initiator-port-name* `-t` *target-port-name* `-l` *logical-unit-name* — The path that consists of the specified initiator port, target port, and logical unit will be enabled.

disable Subcommand

The syntax for the `disable` subcommand is:

```
# mpathadm disable [options]
```

The `disable` subcommand supports the following direct-objects to be disabled:

`path -i` *initiator-port-name* `-t` *target-port-name* `-l` *logical-unit-name* — The path that consists of the specified initiator port, target port, and logical unit will be disabled.

failover Subcommand

The syntax for the `failover` subcommand is:

```
# mpathadm failover direct-object [operand]
```

The `failover` subcommand supports failover for the following direct-objects:

{`logical-unit` | `lu`} *logical-unit-name* — The target port group will failover when the given logical-unit is asymmetric and supports explicit

state change. The currently active target port group will be changed to the standby state and the standby target port group will be active.

override Subcommand

The syntax for the `override` subcommand is:

```
# mpathadm override [options]
```

The `override` subcommand controls whether or not the following direct-objects override another:

path {`-i` *initiator-port-name* `-t` *target-port-name* | `-c`}
`-l` *logical-unit-name*

Cause a path that consists of the specified initiator port, target port, and logical unit to override other paths on the logical unit. Once a path overrides other paths, the mpath-support uses only that path regardless of any other path selection configuration. The `-c` option cancels the setting. The path that consists of the specified initiator port, target port, and logical unit will be disabled.

Options for override path are as follows:

`-i`, `--initiator-port` *initiator-port-name*	Represent the initiator port element of a path. Options `-t` and `-l` must also be included.
`-t`, `--target-port` *target-port-name*	Represent the target port element of a path. Options `-i` and `-l` must also be included.
`-l`, `--logical-unit` *logical-unit*	Represent the logical unit element of a path. Options `-i` and `-t` must also be included.
`-c`, `--cancel`	Cancels overriding setting for the given logical unit. Option `-l` must also be included.

Options The following options are supported:

`-V`, `--version`	Display the version information.
`-?`, `--help`	Display context help. Can be used following an `mpathadm` command with no arguments, following a subcommand, or following a subcommand direct-object combination. Responds with help information appropriate for your entry. For example, if you enter: **`# mpathadm add mpath-support-help`** ... `mpathadm` responds with a display of the options available for that combination of subcommand and direct-object.

Examples

EXAMPLE 1 Obtaining a List of Multipathing Support

The following command uses the `list` subcommand to list all currently registered `mpath-support` libraries.

```
# mpathadm list mpath-support
mpath-support: libmpscsi_vhci.so
```

EXAMPLE 2 Displaying the Properties of a Particular Multipathing Support

The following command uses the `show` subcommand to display the properties for a currently registered `mpath-support` library.

```
# mpathadm show mpath-support libmpscsi_vhci.so
mpath-support:    libmpscsi_vhci.so
       Vendor:  Sun Microsystems
       Driver Name:  scsi_vhci
       Default Load Balance:  round-robin
       Supported Load Balance Types:
              round-robin
          logical-block
Allows To Activate Target Port Group Access:  yes
Allows Path Override:  no
Supported Auto Failback Config:     1
Auto Failback:    on
Failback Polling Rate (current/max):  0/0
Supported Auto Probing Config:    0
Auto Probing:  NA
Probing Polling Rate (current/max):  NA/NA
Supported Devices:
       Vendor:     SUN
       Product:  T300
       Revision:
       Supported Load Balance Types:
              round-robin
       Vendor:     SUN
       Product:  T4
       Revision:
       Supported Load Balance Types:
              round-robin
```

EXAMPLE 3 Obtaining a List of Initiator Ports Discovered Through the mpath-support Libraries

The following command uses the `list initiator-port` subcommand to display a list of initiator ports discovered by the currently registered `mpath-support` libraries.

```
# mpathadm list initiator-port
Initiator-Port: iqn.1986-03.com.sun:01:080020b7ac2b.437a3b3e,4000002a0000
Initiator-Port: 2000000173018713
Initiator-Port: 2000000173818713
```

EXAMPLE 4 Displaying the Properties of a Particular Initiator Port

The following command uses the `show initiator-port` subcommand to display the properties of a particular initiator port discovered using the `list initiator-port` subcommand in an example above.

```
# mpathadm show initiator-port 2000000173018713
initiator-port:     2000000173018713
        Transport Type:     Fibre Channel
        OS device File:     devices/pci@1f,4000/pci@2/SUNW,qlca@5/fp@0,0:fc
```

EXAMPLE 5 Displaying the Properties of a Particular Logical Unit

The following command uses the `show logical-unit` subcommand to display the properties of the logical unit with the specified name.

```
# mpathadm show lu /dev/rdsk/c4t60003BA27D2120004204AC2B000DAB00d0s2
Logical Unit: /dev/rdsk/c4t60003BA27D2120004204AC2B000DAB00d0s2
           mpath-support libmpscsi_vhci.so
           Vendor: SUN
           Product: T4
           Revision: 0301
    Name Type: SCSI Inquiry VPD Page 83 type 3
    Name: 60003ba27d2120004204ac2b000dab00
    Asymmetric: yes
    Current Load Balance: round-robin
    Logical Unit Group ID: NA
    Aauto Failback: on
    Auto Probing: NA

Paths:
    Initiator Port Name: 2000000173818713
    Target Port Name: 20030003ba27d212
    Override Path: NA
    Path State: OK
    Disabled: no

    Initiator Port Name: 2000000173018713
    Target Port Name: 20030003ba27d095
    Override Path: NA
    Path State: OK
    Disabled: no

Target Port Group:
    ID: 2
    Explicit Failover: yes
    Access State: standby
    Target Ports:
            Name: 20030003ba27d212
```

EXAMPLE 5 Displaying the Properties of a Particular Logical Unit *(Continued)*

```
            Relative ID: 0

    ID: 5
    Explicit Failover: yes
    Access State: active
    Target Ports
            Name: 20030003ba27d095
           Relative ID: 0
```

EXAMPLE 6 Enabling a Path

The following command uses the `enable path` subcommand to enable the path with the specified initiator port, target port, and logical unit.

```
# mpathadm enable path -i 2000000173018713 -t 20030003ba27d095     \
-l /dev/rdsk/c4t60003BA27D2120004204AC2B000DAB00d0s2
```

EXAMPLE 7 Modifying mpath-support To Turn On autofailback

```
# mpathadm modify mpath-support -a on libmpscsi_vhci.so
```

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/storage/multipath-utilities
	system/library/storage/libmpapi (`exec_attr(4)` entry)
Interface Stability	Committed

See Also

`stmsboot(1M)`, `libMPAPI(3LIB)`, `exec_attr(4)`, `attributes(5)`

Name mpstat – report per-processor or per-processor-set statistics

Synopsis

```
/usr/bin/mpstat [-amq] [-A core|soc|bins] [-k keys] [-o num]
     [-p | -P set] [-T d | u] [-I statfile] [-O statfile]
     [interval [count]]
```

Description The `mpstat` command reports processor statistics in tabular form. Each row of the table represents the activity of one processor. The first table summarizes all activity since boot. Each subsequent table summarizes activity for the preceding interval. All values are rates listed as events per second unless otherwise noted.

During execution of the kernel status command, the `state` of the kernel can change. If relevant, a state change message is included in the `mpstat` output, in one of the following forms:

```
<<processor 3 moved from pset: -1 to: 1>>
<<pset destroyed: 1>>
<<pset created: 1>>
<<processors added: 1, 3>>
<<processors removed: 1, 3>>
```

The `mpstat` command reports the following information:

`CPU|SET|COR|SOC|BIN`

`CPU`
: Processor ID for which statistics are shown, when the `-a` and `-A` options are omitted.

`SET`
: Processor set ID for which statistics are aggregated, for the `-a` option.

`COR`
: Core ID for which statistics are aggregated, for the `-A core` option.

`SOC`
: Socket ID for which statistics are aggregated, for the `-A soc` option.

`BIN`
: Bin ordinal for which statistics are aggregated, for the `-A` *bins* option.

`minf`
: minor faults

`mjf`
: major faults

`xcal`
: inter-processor cross-calls

`intr`
: interrupts

`ithr`
interrupts as threads (not counting clock interrupt)

`csw`
context switches

`icsw`
involuntary context switches

`migr`
thread migrations (to another processor)

`smtx`
spins on mutexes (lock not acquired on first try)

`srw`
spins on readers/writer locks (lock not acquired on first try)

`syscl`
system calls

`usr`
percent user time

`sys`
percent system time

`wt`
the I/O wait time is no longer calculated as a percentage of CPU time, and this statistic will always return zero.

`idl`
percent idle time

`sze`
number of processors in the requested processor set

`set`
processor set membership of each CPU

Options The following options are supported:

`-a`
Aggregate output by processor set. Sort the output by set. The default output is sorted by CPU number.

`-A core`
Aggregate CPU output by core ID. Data rows having the same core ID are aggregated into one row. The columns are replaced with subtotals, by default. The `-m` option prints column averages, instead.

The `-A` option is incompatible with the `-a` option for aggregating by processor set.

`-A soc`
Aggregate CPU output by socket ID. Data rows having the same socket ID are aggregated into one row. The columns are replaced with subtotals, by default. The `-m` option prints column averages, instead.

The `-A` option is incompatible with the `-a` option for aggregating by processor set.

`-A` *bins*
Aggregate the rows into bins within each sampling period, grouping them in the order in which they appear, and aggregate over rows for each bin. The `-k` option may be used to change the row order prior to the binning step. The `sze` column prints the number of CPUs in each bin. The `BIN` column replaces the `CPU` column and prints the ordinal of each bin.

Aggregation by ID (`-A core|soc`) is processed before sorting (`-k`). Grouping by bins (`-A` *bins*) is done next. Finally, the number of output lines printed per interval may be limited by `-o`.

The `-A` option is incompatible with the `-a` option for aggregating by processor set.

`-I` *statfile*
Replay data previously saved in *statfile*. Create data files for replay by specifying `-O`. This option is especially useful for analyzing statistics on machines with large numbers of CPUs. The file may be reprocessed multiple times using different sorting and aggregation options.

The `-I` option is incompatible with an interval and count specification.

Read from the standard input if the file name is – (hyphen).

`-k` *key1*,...
Sort rows within each sampling period from highest to lowest by *key1*, then *key2*, and so on. Each key may be any of the column headers such as `xcal`, `intr`, `sys`, and so forth.

`-m`
Print the arithmetic mean value rather than the sum when the `-a` or `-A` options are used to aggregate data over multiple CPUs.

`-o` *num*
Print only the first *num* rows within each sampling period, after applying sorting and aggregation options.

`-O` *statfile*
Save all data to *statfile*. This data may be replayed at a later time using `-I`.

Write to the standard output if the file name is – (hyphen).

The purpose of `-O` is to capture all available data. It is incompatible with the data reduction options: `-a`, `-A`, `-k`, `-m`, `-o`, `-p`, and `-P`.

`-p`
Report processor set membership of each CPU. Sort the output by set. The default output is sorted by CPU number.

`-P` *set*
Display only those processors in the specified *set*.

`-q`
Suppress messages related to state changes.

`-T u | d`
Specify `u` for a printed representation of the internal representation of time. See `time(2)`. Specify `d` for standard date format. See `date(1)`.

interval
Report once each *interval* seconds.

count
Only print *count* reports.

Examples

On displays of 80–character width, example output below wraps by one to seven characters. By making a window wider, you can eliminate this wrap.

EXAMPLE 1 Using `mpstat` to Generate User and System Operation Statistics

The following command generates processor statistics over a five–second interval in two reports. The command shows the processor set membership of each CPU. The default output is sorted by CPU number, aggregated by *processor set*, for user (`usr`) and system (`sys`) operations.

```
example% mpstat -ap 5 2


SET minf mjf xcal intr ithr csw icsw migr smtx srw syscl usr sys wt idl sze
  0    6   0  355  291  190  22    0    0    0   0    43   0   2  0  43   1
  1   24  17  534  207  200  70    1    0    2   0   600   4   1  0  84   2
  2   19   7  353  325  318  44    0    0    5   0   345   1   1  0  94   3
  3   36   2  149  237  236  14    0    0    4   0    97   0   0  0  98   2
SET minf mjf xcal intr ithr csw icsw migr smtx srw syscl usr sys wt idl sze
  0    1   0  720  405  304  55    0    0   18   0    12   0  15  0  81   1
  1    0  69 1955  230  200 313   33    4   41   9  7086  34  10  0  19   2
  2    0  46  685  314  300 203   11    0   54   1  5287  36   6  0  28   3
  3    0   0   14  386  384   0    0    0    0   0     0   0   0  0 100   2
```

EXAMPLE 2 Displaying CPUs That Meet Filter Requirement

The following command displays the three CPUs with the highest `intr` rates.

```
example% mpstat -k intr -o 3
CPU minf mjf xcal  intr ithr  csw icsw migr smtx  srw syscl  usr sys  wt idl
 56 1143   5  975  4238   82  465   74   53  124    0 198163  42  17   0  41
```

EXAMPLE 2 Displaying CPUs That Meet Filter Requirement *(Continued)*

```
123 1189   6 1315  1030  890  461   65   53  122    0 24383   27  12   0  62
  4 1184   5 1040   149   70  502   73   55  113    0 82039   31  13   0  56
```

EXAMPLE 3 Aggregating Multiple CPUs into Quartiles

The following command aggregates 256 CPUs into quartiles by `sys` time.

```
example% mpstat -A 4 -k sys
BIN minf mjf xcal intr ithr  csw icsw migr smtx  srw syscl  usr sys  wt idl sze

0   18   0 5811  7105    1 22154   17 9529 1176   0    72    1  79   0  19  64
1    0   0 4624  1006   12 1321   42  418  175    0  3591   36  37   0  27  64
2 1195   5 1056    92   10  526   74   56  104    0 45876   27  12   0  61  64
3    0   0    2    18    8   10    0    0    0    0     1    0   0   0 100  64
```

EXAMPLE 4 Saving Statistics for Later Reprocessing

The following command saves statistics for later reprocessing and aggregates by core ID on a machine with eight CPUs per core.

```
example% mpstat -O /tmp/t1; mpstat -I /tmp/t1 -A core
COR minf mjf xcal  intr ithr  csw icsw migr smtx  srw syscl  usr sys  wt idl sze
514    0   0  124    45    0   21    0    3    1    0     0    0   3   0  97   8
521    0   0   16     5    0    1    0    0    0    0     0    0   0   0 100   8
528    0   0   11     5    0    3    0    0    0    0     0    0   0   0 100   8
535    0   0    7     4    0    1    0    0    0    0     0    0   0   0 100   8
542    0   0    7     4    0    1    0    0    0    0     0    0   0   0 100   8
549    0   0   10     4    0    1    0    0    0    0     0    0   0   0 100   8
556    0   0   10     5    0    1    0    0    0    0     0    0   0   0 100   8
563    0   0    8     4    0    1    0    0    0    0     0    0   0   0 100   8
```

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os
Interface Stability	See below.

Invocation is evolving. Human readable output is unstable.

See Also

`sar(1)`, `date(1)`, `iostat(1M)`, `sar(1M)`, `vmstat(1M)`, `time(2)`, `attributes(5)`

Notes

The sum of CPU utilization might vary slightly from 100 due to rounding errors in the production of a percentage figure.

The total time used for CPU processing is the sum of `usr` and `sys` output values, reported for user and system operations. The `idl` value reports the time that the CPU is idle for any reason other than pending disk I/O operations.

Run the `iostat` command with the `-x` option to report I/O service times in `svc_t` output. The `iostat` utility also reports the same `wt`, user (`us`), and system (`sy`) statistics. See `iostat(1M)` for more information.

When executing in a `zone` and if the pools facility is active, `mpstat`(1M) will only provide information for those processors which are a member of the processor set of the pool to which the `zone` is bound.

Name msgid – generate message IDs

Synopsis

```
/usr/sbin/msgid
```

Description The `msgid` utility generates message IDs.

A message ID is a numeric identifier that uniquely identifies a message. Although the probability of two distinct messages having the same ID is high, this can be greatly reduced with the appropriate priority or facility.level designator (see `syslogd(1M)`). Specifically, the message ID is a hash signature on the message's unexpanded format string, generated by `STRLOG_MAKE_MSGID()` as defined in `<sys/strlog.h>`.

`syslogd(1M)` is a simple filter that takes strings as input and produces those same strings, preceded by their message IDs, as output. Every message logged by `syslogd`(1M) includes the message ID. The message ID is intended to serve as a small, language-independent identifier.

Examples **EXAMPLE 1** Using the msgid command to generate a message ID

The following example uses the `msgid` command to generate a message ID for the `echo` command.

```
example# echo hello | msgid
205790 hello
example#
```

EXAMPLE 2 Using the msgid command to Generate a Message Catalog

The following example uses the `msgid` command to enumerate all of the messages in the binary `zfs`, on an x86 machine, to generate a message catalog.

```
example# strings /kernel/fs/amd64/zfs | msgid
...
726970 stride_hits
766819 stride_misses
929857 reclaim_successes
412490 reclaim_failures
234331 streams_resets
737841 streams_noresets
471619 bogus_streams
878613 onloan_read_buf
...
```

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also syslogd(1M), attributes(5), log(7D)

Name mvdir – move a directory

Synopsis `/usr/sbin/mvdir` *dirname name*

Description `mvdir` moves directories within a file system. `dirname` must be a directory. If *name* does not exist, it will be created as a directory. If *name* does exist, and is a directory, `dirname` will be created as *name*/*dirname*. `dirname` and *name* may not be on the same path; that is, one may not be subordinate to the other. For example:

```
example% mvdir x/y x/z
```

is legal, but

```
example% mvdir x/y x/y/z
```

is not.

Operands

`dirname` The name of the directory that is to be moved to another directory in the filesystem.

name The name of the directory into which `dirname` is to be moved. If *name* does not exist, it will be created. It may not be on the same path as `dirname`.

Usage See `largefile(5)` for the description of the behavior of `mvdir` when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

Exit Status

`0` Successful operation.

`>0` Operation failed.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also `mkdir(1)`, `mv(1)`, `attributes(5)`, `largefile(5)`

Name named, in.named – Internet domain name server

Synopsis

```
named [-fgsVv] [-c config-file] [-d debug-level] [-m flag]
     [-n #cpus] [-p port] [-S #max-socks] [-t directory]
     [-u user] [-x cache-file] [-4 | -6]
```

Description The `named` utility is a Domain Name System (DNS) server, part of the BIND 9 distribution from ISC. For more information on the DNS, see RFCs 1033, 1034, and 1035.

When invoked without arguments, `named` reads the default configuration file `/etc/named.conf`, reads any initial data, and listens for queries.

`in.named` is a link to `named`.

Options The following options are supported:

`-4`
Use only IPv4 transport. By default, both IPv4 and IPv6 transports can be used. Options `-4` and `-6` are mutually exclusive.

`-6`
Use only IPv6 transport. By default, both IPv4 and IPv6 transports can be used. Options `-4` and `-6` are mutually exclusive.

`-c` *config-file*
Use *config-file* as the configuration file instead of the default `/etc/named.conf`. To ensure that reloading the configuration file continues to work after the server has changed its working directory due to to a possible *directory* option in the configuration file, *config-file* should be an absolute pathname.

`-d` *debug-level*
Set the daemon's debug level to *debug-level*. Debugging traces from `named` become more verbose as the debug level increases.

`-f`
Run the server in the foreground (that is, do not run as a daemon).

`-g`
Run the server in the foreground and force all logging to `stderr`.

`-m` *flag*
Turn on memory usage debugging flags. Possible flags are `usage`, `trace`, and `record`, `size`, and `mctx`. These correspond to the `ISC_MEM_DEBUG`*XXXX* flags described in `<isc/mem.h>`.

`-n` *#cpus*
Create *#cpus* worker threads to take advantage of multiple CPUs. If not specified, `named` will try to determine the number of CPUs present and create one thread per CPU. If it is unable to determine the number of CPUs, a single worker thread will be created.

`-p` *port*
Listen for queries on port *port*. If not specified, the default is port 53.

`-S` *#max-socks*
Allow `named` to use up to *#max-socks* sockets.

This option should be unnecessary for the vast majority of users. The use of this option could even be harmful, because the specified value might exceed the limitation of the underlying system API. It therefore should be set only when the default configuration causes exhaustion of file descriptors and the operational environment is known to support the specified number of sockets. Note also that the actual maximum number is normally a little smaller than the specified value because `named` reserves some file descriptors for its internal use.

`-s`
Write memory usage statistics to *stdout* on exit.

This option is mainly of interest to BIND 9 developers and might be removed or changed in a future release.

`-t` *directory*
Change the root directory using `chroot(2)` to *directory* after processing the command line arguments, but before reading the configuration file.

This option should be used in conjunction with the `-u` option, as chrooting a process running as root does not enhance security on most systems; the way `chroot()` is defined allows a process with root privileges to escape a `chroot` jail.

`-u` *user*
Set the real user ID using `setuid(2)` to *user* after completing privileged operations, such as creating sockets that listen on privileged ports.

`-V`
Report the version number and build options, and exit.

`-v`
Report the version number and exit.

`-x` *cache-file*
Load data from *cache-file* into the cache of the default view.

Do not use this option. It is of interest only to BIND 9 developers and might be removed or changed in a future release.

Extended Description

This section describes additional attributes of `named`.

SMF Properties

When starting named from the service management facility, `smf(5)`, `named` configuration is read from the service configuration repository. Use `svcprop(1)` to list the properties and `svccfg(1M)` to make changes.

The following application configuration properties are available to administrators:

`options/`*server*
Specifies the server executable to be used instead of the default server, `/usr/sbin/named`.

`options/`*configuration_file*
Specifies the configuration file to be used instead of the default, `/etc/named.conf`. A directory option might be specified in the configuration file. To ensure that reloading the configuration file continues to work in such a situation, *configuration_file* should be specified as an absolute pathname. This pathname should not include the *chroot_dir* pathname. This property is the equivalent of the `-c` option.

`options/`*ip_interfaces*
Specifies over which IP transport, IPv4 or IPv6, BIND will transmit. Possible values are `IPv4` or `IPv6`. Any other setting assumes `all`, the default. This property is the equivalent of command line option `-4` or `-6`

`options/`*listen_on_port*
Specifies the default UDP and TCP port to be used for listening to DNS requests. This property is the equivalent of the command line option `-p` *port*.

`options/`*debug_level*
Specifies the default debug level. The default is 0, which means no debugging. The higher the number the more verbose debug information becomes. Equivalent of the command line option `-d` *debug_level*.

`options/`*threads*
Specifies the number of CPU worker threads to create. The default of 0 causes `named` to try and determine the number of CPUs present and create one thread per CPU. Equivalent of command line option `-n` *#cpus*.

`options/`*chroot_dir*
Specifies the directory to be used as the root directory after processing SMF properties and the command line arguments but before reading the configuration file. Use this property when using a `chroot(2)` environment. Synonymous to command line option `-t` *pathname*.

When using `chroot(2)`, `named` is unable to disable itself when receiving `rndc(1M)` `stop` or `halt` commands. Instead, you must use the `svcadm(1M)` `disable` command.

In the event of a configuration error originating in one of the above SMF application options, `named` displays a message providing information about the error and the parameters that need correcting. The process then exits with exit code `SMF_EXIT_ERR_CONFIG`.

At startup, in the event of an error other than a configuration error, `named` exits with exit code `SMF_EXIT_ERR_FATAL`. Both of this code and `SMF_EXIT_ERR_CONFIG` cause the start method, `smf_method(5)`, to place the service in the maintenance state, which can be observed with the `svcs(1)` command `svcs -x`.

In addition to the properties listed above, the following property can be used to invoke `named` as a user other than root:

`start/`*user*
Specifies the identity of the user that is invoking `named`. See `smf_method(5)` and `chroot(2)`. Note that the user must have `solaris.smf.manage.bind` authorization. Without this role the `named` will be unable to manage its SMF FMRI and `named` will automatically be restarted by the SMF after an `rndc(1M)` `stop` or `halt` command. See `EXAMPLES` for a sequence of commands that establishes the correct authorization.

SIGNALS In routine operation, signals should not be used to control the nameserver; `rndc(1M)` should be used instead.

`SIGHUP`
Force a reload of the server.

`SIGINT`, `SIGTERM`
Shut down the server.

The result of sending any other signals to the server is undefined.

Configuration The `named` configuration file is too complex to describe in detail here. A list of configuration options is provided in the `named.conf` man page shipped with the BIND 9 distribution. A complete description is provided in the *BIND 9 Administrator Reference Manual*.

Examples

EXAMPLE 1 Configuring `named` to Transmit Only over IPv4 Networks

The following command sequence configures `named` such that it will transmit only over IPv4 networks.

```
# svccfg -s svc:network/dns/server:default setprop \
> options/ip_interfaces=IPv4
# svcadm refresh svc:network/dns/server:default
#
```

EXAMPLE 2 Listing Current Configuration File and Setting an Alternative File

The following sequence of commands lists the current `named` configuration file and sets an alternative file.

```
# svcprop -p options/configuration_file dns/server:default
/etc/named.conf
# svccfg -s dns/server:default setprop \
> options/configuration_file=/var/named/named.conf
# svcadm refresh dns/server:default
# svcprop -p options/configuration_file dns/server:default
/var/named/named.conf
```

EXAMPLE 3 Establishing Appropriate Authorization for `named`

To have `named` start with the `solaris.smf.manage.bind` authorization, perform the steps shown below.

Add the user `dnsadmin` to the `solaris.smf.manage.bind` role:

EXAMPLE 3 Establishing Appropriate Authorization for named *(Continued)*

```
# usermod -A solaris.smf.manage.bind dnsadmin
Observe effect of command:
# tail -1 /etc/user_attr
dnsadmin::::type=normal;auths=solaris.smf.manage.bind
```

Modify the service properties:

```
# svccfg
svc:> select svc:/network/dns/server:default
svc:/network/dns/server:default> setprop start/user = dnsadmin
svc:/network/dns/server:default> setprop start/group = dnsadmin
svc:/network/dns/server:default> exit
# svcadm refresh svc:/network/dns/server:default
# svcadm restart svc:/network/dns/server:default
```

Because only root has write access to create the default process-ID file, `/var/run/named/named.pid`, named must be configured to use an alternative path for the user `dnsadmin`. Here is an example of how to accomplish this:

```
# mkdir /var/named/tmp
# chown dnsadmin /var/named/tmp
```

Shown below is what you must add to `named.conf` to make use of the directory created above.

```
# head /etc/named.conf
options {
directory "/var/named";
pid-file "/var/named/tmp/named.pid";
};
```

Files

`/etc/named.conf`
default configuration file

`/var/run/named/named.pid`
default process-ID file

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	service/network/dns/bind
Interface Stability	Volatile

See Also `svcs(1)`, `named-checkconf(1M)`, `named-checkzone(1M)`, `rndc(1M)`, `rndc-confgen(1M)`, `svcadm(1M)`, `svccfg(1M)`, `svcprop(1)`, `chroot(2)`, `setuid(2)`, `bind(3SOCKET)`, `attributes(5)`, `smf(5)`, `smf_method(5)`

RFC 1033, *RFC 1034*, *RFC 1035*

See the BIND 9 *Administrator's Reference Manual*. As of the date of publication of this man page, this document is available at `https://www.isc.org/software/bind/documentation`.

The `named.conf` man page shipped with the BIND 9 distribution

Name named-checkconf – named configuration file syntax checking tool

Synopsis `named-checkconf [-hjvz] [-t` *directory*`]` *filename*

Description The `named-checkconf` utility checks the syntax, but not the semantics, of a specified configuration file.

Options The following options are supported:

`-h`
Display the usage summary and exit.

`-j`
When loading a zonefile, read the journal if it exists.

`-t` *directory*
Change the root directory to *directory* so that include directives in the configuration file are processed as if run by a named configuration whose root directory has been similarly changed.

`-v`
Print the version of the `named-checkconf` program and exit.

`-z`
Perform a test load of the master zones found in `named.conf`.

Operands The following operands are supported:

filename
The name of the configuration file to be checked. If not specified, it defaults to `/etc/named.conf`.

Exit Status `0`
No errors were detected.

`1`
An error was detected.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	service/network/dns/bind
Interface Stability	Volatile

See Also `named(1M)`, `named-checkzone(1M)`, `attributes(5)`

See the BIND 9 *Administrator's Reference Manual*. As of the date of publication of this man page, this document is available at `https://www.isc.org/software/bind/documentation`.

Name named-checkzone, named-compilezone – zone file validity checking or converting tool

Synopsis

```
named-checkzone [-Ddhjqv] [-c class] [-F format] [-f format]
     [-i mode] [-k mode] [-M mode] [-m mode] [-n mode]
     [-o filename] [-S mode] [-s style] [-t directory]
     [-W mode] [-w directory] zonename filename

named-compilezone [-Ddjqv] [-C mode] [-c class] [-F format]
     [-f format] [-i mode] [-k mode] [-m mode] [-n mode]
     [-o filename] [-s style] [-t directory]
     [-W mode] [-w directory] zonename filename
```

Description The `named-checkzone` utility checks the syntax and integrity of a zone file. It performs the same checks as `named`(1M) does when loading a zone. The `named-checkzone` utility is useful for checking zone files before configuring them into a name server.

`named-compilezone` is similar to `named-checkzone`, differing in that it always dumps the zone contents to a specified file in a specified format. Additionally, it applies stricter check levels by default, since the dump output will be used as an actual zone file loaded by `named`(1M). Unless manually specified otherwise, the check levels must be at least as strict as those specified in the `named` configuration file.

Options For either or both utilities, the following options are supported:

`-c` *class*
Specify the class of the zone. If not specified, "IN" is assumed.

`-D`
Dump zone file in canonical format.

`-d`
Enable debugging.

`-F` *format*
Specify the format of the output file specified. Possible formats are `text` (default) and `raw`. For `named-checkzone`, this does not cause any effects unless it dumps the zone contents.

`-f` *format*
Specify the format of the zone file. Possible formats are `text` (default) and `raw`.

`-h`
Display usage message for `named-checkzone`.

`-i` *mode*
Perform post-load zone integrity checks. Possible modes are `full` (default), `full-sibling`, `local`, `local-sibling`, and `none`.

Mode `full` checks that MX records refer to the A or AAAA record (both in-zone and out-of-zone hostnames). Mode `local` checks only MX records that refer to in-zone hostnames.

Mode `full` checks that SRV records refer to the A or AAAA record (both in-zone and out-of-zone hostnames). Mode `local` checks only SRV records that refer to in-zone hostnames.

Mode `full` checks that delegation NS records refer to A or AAAA record (both in-zone and out-of-zone hostnames). It also checks that glue address records in the zone match those advertised by the child. Mode `local` checks only NS records that refer to in-zone hostnames or check that some required glue exists, that is, when the nameserver is in a child zone.

Mode `full-sibling` and `local-sibling` disable sibling glue checks, but are otherwise the same as `full` and `local`, respectively.

Mode `none` disables the checks.

`-k` *mode*
Perform "check-name" checks with the specified failure mode. Possible modes are `fail` (default for `named-compilezone`), `warn` (default for `named-checkzone`) and `ignore`.

`-j`
Read the journal, if it exists, when loading the zone file.

`-M` *mode*
Check if an MX record refers to a `CNAME`. Possible modes are `fail`, `warn` (default) and `ignore`.

`-m` *mode*
Specify whether MX records should be checked to see if they are addresses. Possible modes are `fail`, `warn` (default) and `ignore`.

`-n` *mode*
Specify whether NS records should be checked to see if they are addresses. Possible modes are `fail` (default for `named-compilezone`), `warn` (default for `named-checkzone`) and `ignore`.

`-o` *filename*
Write zone output to *filename*. If filename is `-` (a hyphen), then write to standard out. The hyphen mandatory for `named-compilezone`

`-q`
Run in quiet mode, reporting only the exit status.

`-S` *mode*
Check if a SRV record refers to a `CNAME`. Possible modes are `fail`, `warn` (default) and `ignore`.

`-s` *style*
Specify the style of the dumped zone file. Possible styles are `full` (default) and `relative`. The `full` format is most suitable for processing automatically by a separate script. The `relative` format is more human-readable and is thus suitable for editing by hand. For

`named-checkzone` this option does not cause any effects unless it dumps the zone contents. It also has no effect if the output format is not text.

`-t` *directory*
`chroot` to directory so that include directives in the configuration file are processed as if run by a similarly `chrooted` `named`.

`-v`
Print the version of the `named-checkzone` program and exit.

`-W` *mode*
Specify whether to check for non-terminal wildcards. Non-terminal wildcards are almost always the result of a failure to understand the wildcard matching algorithm (RFC 1034). Possible modes are `warn` (default) and `ignore`.

`-w` *directory*
`chdir` to *directory* so that relative filenames in master file `$INCLUDE` directives work. This is similar to the directory clause in `named.conf`.

Operands The following operands are supported:

filename
The name of the zone file.

zonename
The domain name of the zone being checked.

Exit Status `0`
No errors were detected.

`1`
An error was detected.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	service/network/dns/bind
Interface Stability	Volatile

See Also `named(1M)`, `named-checkconf(1M)`, `attributes(5)`

RFC 1035

See the BIND 9 *Administrator's Reference Manual*. As of the date of publication of this man page, this document is available at `https://www.isc.org/software/bind/documentation`.

Name ncaconfd – Solaris Network Cache and Accelerator (NCA) configuration daemon

Synopsis `/usr/lib/inet/ncaconfd [-al ]` *interface1* `[`*interface2* `...]`

Description Use the `ncaconfd` utility to set up NCA on a system. At boot time, the `ncakmod` initialization script reads in `nca.if(4)` to determine on which interface(s) NCA should run. `ncaconfd` then sets up the interface.

`ncaconfd` also operates as a daemon if the `nca_active` key is set to enabled in `ncakmod.conf(4)` file. In this case, `ncaconfd` will continue as a daemon after all the NCA interfaces have been set up, listening for routing changes. The changes are then passed to NCA to control which interface NCA should use to make active outgoing TCP connections.

Options The following options are supported:

`-a` Enable active connections.

`-l` Enable logging.

Files `/etc/nca/ncakmod.conf`

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/network/http-cache-accelerator
Interface Stability	Committed

See Also `ncakmod(1)`, `nca.if(4)`, `ncakmod.conf(4)`, `attributes(5)`, `nca(7d)`

Name ncheck – generate a list of path names versus i-numbers

Synopsis

```
ncheck [-F FSType] [-V] [generic_options]
     [-o FSType-specific_options] [special]...
```

Description `ncheck` with no options generates a path-name versus i-number list of all files on *special*. If *special* is not specified on the command line the list is generated for all *special*s in `/etc/vfstab` which have a numeric `fsckpass`. *special* is the raw device on which the file system exists.

Options

`-F` Specify the *FSType* on which to operate. The *FSType* should either be specified here or be determinable from `/etc/vfstab` by finding an entry in the table that has a numeric `fsckpass` field and an `fsckdev` that matches *special*.

`-V` Echo the complete command line, but do not execute the command. The command line is generated by using the options and arguments provided by the user and adding to them information derived from `/etc/vfstab`. This option may be used to verify and validate the command line.

generic_options Options that are commonly supported by most FSType-specific command modules. The following options are available:

- `-i` *i-list* Limit the report to the files on the *i-list* that follows. The *i-list* must be separated by commas with no intervening spaces.
- `-a` Print the names "." and ". ." which are ordinarily suppressed.
- `-s` Report only special files and files with set-user-ID mode. This option may be used to detect violations of security policy.

`-o` Specify *FSType-specific_options* in a comma separated (without spaces) list of suboptions and keyword-attribute pairs for interpretation by the FSType-specific module of the command.

Usage See `largefile(5)` for the description of the behavior of `ncheck` when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

Files `/etc/vfstab` list of default parameters for each file system

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also `vfstab(4)`, `attributes(5)`, `largefile(5)` Manual pages for the FSType-specific modules of `ncheck`

Notes This command may not be supported for all *FSTypes*.

Name ncheck_ufs – generate pathnames versus i-numbers for ufs file systems

Synopsis

```
ncheck -F ufs [generic_options] [-o m] [special]...
```

Description `ncheck -F ufs` generates a pathname versus i-number list of files for the `ufs` file system residing on *special*. Names of directory files are followed by '`/.`'.

Options See `ncheck(1M)` for the list of *generic_options* supported.

`-o` Specify `ufs` file system specific options. The available option is:

`m` Print mode information.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also `ff(1M)`, `ncheck(1M)`, `attributes(5)`

Diagnostics When the file system structure is improper, '`??`' denotes the "parent" of a parentless file and a pathname beginning with '`. . .`' denotes a loop.

Name ndd – get and set driver configuration parameters

Synopsis

```
ndd [-set] driver parameter [value]
```

Description ndd gets and sets selected configuration parameters in some kernel drivers. Currently, ndd only supports the drivers that implement the TCP/IP Internet protocol family. Each driver chooses which parameters to make visible using ndd. Since these parameters are usually tightly coupled to the implementation, they are likely to change from release to release. Some parameters may be read-only.

Note – It is strongly encouraged that you use ipadm(1M), rather than ndd, to modify or retrieve TCP/IP Internet protocols. The current ndd command will be made obsolete in a future release, replaced by ipadm(1M). Please see NOTES for more information.

If the ndd -set option is omitted, ndd queries the named *driver*, retrieves the value associated with the specified *parameter*, and prints it. If the -set option is given, ndd passes *value*, which must be specified, down to the named *driver* which assigns it to the named *parameter*.

By convention, drivers that support ndd also support a special read-only *parameter* named "?" which can be used to list the parameters supported by the driver.

Examples **EXAMPLE 1** Getting Parameters Supported By The TCP Driver

To see which parameters are supported by the TCP driver, use the following command:

```
example% ndd /dev/tcp \?
```

The parameter name "?" may need to be escaped with a backslash to prevent its being interpreted as a shell meta character.

The following command sets the value of the parameter *ip_forwarding* in the dual stack IP driver to zero. This disables IPv4 packet forwarding.

```
example% ndd -set /dev/ip ip_forwarding 0
```

Similarly, in order to disable IPv6 packet forwarding, the value of parameter *ip6_forwarding*

```
example% ndd -set /dev/ip ip6_forwarding 0
```

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also dladm(1M), ipadm(1M), ioctl(2), attributes(5), nca(7d), arp(7P), ip(7P), ip6(7P), tcp(7P), udp(7P)

Notes The parameters supported by each driver may change from release to release. Like programs that read `/dev/kmem`, user programs or shell scripts that execute `ndd` should be prepared for parameter names to change.

The `ioctl()` command that `ndd` uses to communicate with drivers is likely to change in a future release. User programs should avoid making dependencies on it.

The use of `ndd` to administer Layer 2 (Data Link layer) drivers is strongly discouraged as this capability is to be obsoleted in a future release, replaced by `dladm`(1M). Please refer to the driver-specific man page in section 7D of the SunOS man pages.

The use of `ndd` to administer the drivers that implement the TCP/IP Internet protocol family (IP/TCP/SCTP/UDP/ICMP) is strongly discouraged as this capability is to be obsoleted in a future release, replaced by `ipadm`. Please see `ipadm`(1M) for instructions for modifying and retrieving supported protocol properties.

The meanings of many `ndd` parameters make sense only if you understand how the driver is implemented.

Name ndmpadm – administer Network Data Management Protocol activities

Synopsis `/usr/sbin/ndmpadm [-? ]` *subcommand* `[`*options*`] [`*direct-object*`]`

Description The `ndmpadm` command can be used to query the `ndmpd(1M)` daemon to get the status of active sessions, terminate a session, query backup devices, and set or get the current NDMP (Network Data Management Protocol) service variables and properties. `ndmpadm` is implemented as a set of subcommands, many with their own direct object, which are described in the section for a given subcommand. Certain subcommands support options, which are described along with the subcommand.

The `ndmpadm` command supports the following subcommands:

`disable`
Disable the specified authentication password handling.

`enable`
Enable the specified authentication password handling.

`get`
Get the value of an NDMP configuration property.

`kill-sessions`
Terminate an active session.

`set`
Set the value of an NDMP configuration property.

`show-devices`
Get a list of tape devices connected to the server.

`show-sessions`
Display the details of active NDMP sessions.

Options The following option is supported:

`-?` Display a list of all subcommands and options.

Sub-commands The `ndmpadm` command supports the subcommands described below.

`disable` Subcommand The syntax for the `disable` subcommand is:

```
# ndmpadm disable -a auth-type
```

This subcommand disables the authentication type specified by *auth-type* for an NDMP client's remote access. Valid values for *auth-type* are `cram-md5` or `cleartext`.

`enable` Subcommand The syntax for the `enable` subcommand is:

```
# ndmpadm enable -a auth-type -u username
```

This subcommand prompts for the user's password twice for confirmation and activates the specified authentication type with the given username and password for NDMP client access. Valid values for *auth-type* are `cram-md5` or `cleartext`.

`get` Subcommand

The syntax for the `get` subcommand is:

```
# ndmpadm get [-p] [property] [[-p] property=value]...
```

The property names are the same as used for the `set` subcommand and are described below. If you do not specify a property, the `get` subcommand returns all configuration properties.

`kill-sessions` Subcommand

The `kill-sessions` subcommand allows you to terminate the session number *ID*.

The syntax for the `kill-sessions` subcommand is:

```
# ndmpadm kill-sessions ID
```

`set` Subcommand

The syntax for the `set` subcommand is:

```
# ndmpadm set [-p] property=value [[-p] property=value]...
```

The properties you can set with the `set` subcommand are described in the ndmp(4) man page.

`show-devices` Subcommand

The syntax for the `show-devices` subcommand is:

```
# ndmpadm show-devices
```

This subcommand lists the name, vendor, serial number, and other information about the current tape drive and libraries connected to the system.

`show-sessions` Subcommand

The `show-sessions` subcommand displays details of a session. The syntax for the `show-sessions` subcommand is:

```
# ndmpadm show-sessions [-i tape,scsi,data,mover] [ID]
```

The `show-sessions` subcommand supports the following arguments:

`-i` `tape`,`scsi`,`data`,`mover`
: Identify a type of interface about which to obtain data. If no interface is specified, `show-sessions` displays information for all types of interfaces.

ID
: Identifies a particular session about which to display data. If no *ID* is specified, `show-sessions` displays data for all sessions.

Examples

EXAMPLE 1 Obtaining the Status of All NDMP Connections

The following command obtains status on all connections.

```
# ndmpadm show-devices
```

EXAMPLE 2 Obtaining the Status of Certain Types of Connections

The following command obtains status on tape and SCSI interfaces.

```
# ndmpadm show-sessions -i scsi,tape
```

EXAMPLE 3 Limiting Protocol Version

The following command limits the use of the NDMP protocol to version 3.

```
# ndmpadm set -p version=3
```

EXAMPLE 4 Obtaining Current Version Number

The following command obtains the version number of the currently running NDMP.

```
# ndmpadm get -p version
```

EXAMPLE 5 Disconnecting a Specific Session

The command shown below disconnects session 5. The session number was previously obtained from an `ndmpadm show-sessions` command.

```
# ndmpadm kill-session 5
```

EXAMPLE 6 Obtaining the Values for All NDMP Properties

The following command obtains the values for all NDMP properties.

```
# ndmpadm get
```

EXAMPLE 7 Enabling CRAM-MD5 Authentication

The following command enables CRAM-MD5 authentication.

```
# ndmpadm enable -a cram-md5 -u admin
Enter new password:*****
Re-enter password:*****
```

EXAMPLE 8 Disabling Clear Text Password Authentication

The following command disables clear text password authentication.

```
# ndmpadm disable -a cleartext
```

Exit Status

`0` Successful completion.

`1` An error occurred, such as the `ndmpd` daemon is not running, that prevented `ndmpadm` from contacting the demon.

`2` Invalid command-line options were specified.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	service/storage/ndmp
Interface Stability	Committed

See Also dump(1), tar(1), ndmpd(1M), ndmpstat(1M), svccfg(1M), syslogd(1M), ndmp(4), attributes(5), smf(5)

Notes The ndmpd(1M) daemon is managed by the service management facility (smf(5)), under the service identifier:

```
svc:/system/ndmpd
```

Administrative actions on this service, such as setting and getting a property can be alternatively performed using svccfg(1M). For example to enable Direct Access Recovery (DAR) mode:

```
# svccfg -s svc:/system/ndmpd
svc:/system/ndmpd> setprop ndmpd/dar-support = yes
```

...and to get the list of properties:

```
# svccfg -s svc:/system/ndmpd
svc:/system/ndmpd> listprop
```

Name ndmpd – Network Data Management Protocol daemon

Synopsis

```
/usr/lib/ndmp/ndmpd
```

Description The `ndmpd` daemon handles client Network Data Management Protocol (NDMP) requests. NDMP is an open, enterprise-wide, network-based data management protocol used for backup and recovery. The `ndmpd` daemon enables users to manage data backup and recovery using Data Management Application (DMA) clients. The NDMP protocol is used to coordinate data movement and control between a DMA and an NDMP server or between two NDMP servers.

By default, `ndmpd` is disabled.

Exit Status

`0` Successful completion.

`1` An error occurred that prevented the `ndmpd` daemon from initializing, such as failure to fork a process, mutex initialization.

`2` Invalid command-line options were specified.

Files

`/usr/lib/ndmp/ndmpd`
Network data management protocol server binary.

`/var/ndmp/ndmp.log`
Network data management protocol log messages file. This file is deleted upon reboot.

`/var/ndmp/dumpdates`
A text file that stores information about the date and the level of dump backups.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	service/storage/ndmp
Interface Stability	Committed

See Also `svcs(1)`, `ndmpadm(1M)`, `svcadm(1M)`, `svccfg(1M)`, `syslogd(1M)`, `ndmp(4)`, `attributes(5)`, `smf(5)`

Notes The `ndmpd` daemon is managed by the service management facility (`smf(5)`), under the service identifier:

```
svc:/system/ndmpd:default
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. The service's status can be queried using the `svcs(1)` command.

Name ndmpstat – show NDMP backup progress statistics

Synopsis `ndmpstat [`*tapes*`] [`*interval* `[`*count*`]]`

Description The `ndmpstat` utility reports Network Data Management Protocol (NDMP) statistics, among which are NDMP worker threads, disk IO, tape IO, files operation, performance, and backup activity.

`ndmpstat` reports the aggregate statistics for all tapes and disks. In order to obtain statistics for specific tape devices, the tape device name should be passed as argument to the utility.

When invoked, `ndmpstat` begins its display with a one-line summary of the NDMP daemon activity since the NDMP service was invoked.

Display Fields The fields in `ndmpstat` output are described as follows:

`wthr`
Report the number of worker threads in each of the four following states:

`r`
the number of worker threads running

`w`
the number of blocked worker threads that are waiting for resources such as I/O and paging

`b`
the number of backup operations currently running

`r`
the number of restore operations currently running

`file`
Report on usage of filesystem.

`rd`
the number of files being read

`wr`
the number of files being written

`disk`
Report the number of disk operations per interval.

`rd`
the number of disk blocks being read

`wr`
the number of disk blocks being written

`tape`
Report the number of tape operations per interval. There are slots for up to four tapes, labeled with a single number. The number indicates the name of the device under `/dev/rmt`.

`rd`
the number of tape blocks being read

`wr`
the number of tape blocks being written

`bytes`
Report the number of bytes transferred. This is the aggregate value of both tape and disk devices. The number is in kilobytes.

`rd`
the number of kilobytes being read

`wr`
the number of kilobytes being written

`perf`
Displays a rough estimate of performance of the backup/restore operation in megabytes per second.

`bk`
backup performance

`rs`
restore performance

`prcnt`
Display the comparative usage of resources, in percent.

`dsk`
disk I/O time

`tpe`
tape I/O time

`otr`
other time (memory or idle)

See `EXAMPLES`.

Operands The following operands are supported:

count Specifies the number of times that the statistics display is repeated.

tape Specifies which tapes are to be given priority in the output. A command line is limited to a maximum of four tape devices. A common tape name is `/dev/rmt/`*n*, where *n* is an integer.

interval Specifies the number of seconds over which `ndmpstat` summarizes activity. The specified interval remains in effect till the command is terminated.

Examples

EXAMPLE 1 Using `ndmpstat`

The following command displays a summary of NDMP daemon activity at five-second intervals.

```
example% ndmpstat 5
wthr     file      disk      tape        bytes      perf       prcnt
r w b r rd  wr    rd  wr   rd    wr     rd    wr   bk  rs  dsk  tpe  otr
1 0 3 6 50   9   1250  0  32544 4455   42335 3234   5  4   20   40   40
1 0 0 1  1   0    128  0    0   128      64    64    1  0    0   80   20
1 0 0 1  2   0    128  0    0     0      64     0    1  0   80    0   20
1 0 0 1  1   0    128  0    0     0      64     0    1  0   80    0   20
1 0 0 1  3   0    128  0    0     0      64     0    0  0   80    0   20
1 0 0 1  1   0    128  0    0   128      64    64    1  0    0   80   20
^C
example%
```

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	service/storage/ndmp
Interface Stability	See below

Invocation is evolving. Human readable output is unstable.

See Also

`iostat(1M)`, `mpstat(1M)`, `ndmpd(1M)`, `ndmpadm(1M)`, `attributes(5)`

Notes

Performance numbers are not accurate and are rounded up at the MB/sec. boundary.

Name netadm – administer network configuration profiles

Synopsis

```
netadm enable [ -p profile-type ] [ -c ncu-class ] profile-name

netadm disable [ -p profile-type ] [ -c ncu-class ] profile-name

netadm list [ -x ] [ -p profile-type ] [ -c ncu-class ]
     [ profile-name ]

netadm show-events

netadm scan-wifi linkname

netadm select-wifi linkname

netadm help
```

Description The `netadm` utility is used to administer network profiles and interact with the NWAM daemon.

There are three types of network profiles: Network Configuration Profiles (NCPs), Locations, and External Network Modifiers (ENMs).

At any given time, there is one active NCP and one active Location on a system. Enabling a different NCP or Location (with activation-mode `manual`) will implicitly disable the current active NCP or Location. The current Location (if its activation-mode is `manual`) can also be disabled, though the effect of this will be to "turn off" some aspects of the system's networking capabilities, such as name services. Explicitly disabling an NCP is not permitted, as that would effectively shut down the basic network connectivity of the system. An NCP is only disabled implicitly when a different NCP is enabled.

Conversely, there can be zero or more active ENMs at any given time. Thus enabling or disabling one ENM has no effect on other active ENMs.

Enabling and disabling of individual NCUs is also allowed; the specified NCU must be part of the currently active NCP, and must have its activation mode set to `manual`. If an NCU class is not specified, all NCUs (one link and/or one interface) with the given name will be enabled or disabled.

Enabling and disabling of objects is performed asynchronously. Thus, the request to enable or disable can succeed, while the action itself fails. A failure of this sort will be reflected in the object state; `maintenance` state indicates that the last action taken failed. Note that enabling NCPs and locations in particular can be time-consuming, depending on the configuration. Completion can be verified by checking the state of the appropriate SMF service (`svc:network/physical:default` for NCPs, and `svc:network/location:default` for locations). The state of the individual NCUs that make up an NCP may also be verified with the `netadm list` command.

There are two system-defined NCPs: DefaultFixed, and Automatic. The DefaultFixed NCP represents a manually configured network environment, while Automatic is the default

NWAM-managed environment, which attempts to configure all connected physical interfaces using DHCP. You can use netcfg(1M) to create additional NWAM-managed NCPs.

Sub-commands The following subcommands are supported:

`enable` [`-p` *profile-type*] [`-c` *ncu-class*] *profile-name*
Enable the specified profile. If the profile name is not unique, the profile type must be specified to identify the profile to be enabled. If the profile type is NCU and the name is not unique (that is, there is both a link and interface NCU with the same name), both NCUs will be enabled, unless the `-c` option is used to specify the NCU class. Profile type must be one of `ncp`, `ncu`, `loc`, or `enm`; NCU class must be one of `phys` or `ip`.

`disable` [`-p` *profile-type*] [`-c` *ncu-class*] *profile-name*
Disable the specified profile. If the profile name is not unique, the profile type must be specified to identify the profile to be disabled. If the profile type is NCU and the name is not unique (that is, there is both a link and interface NCU with the same name), both NCUs will be disabled, unless the `-c` option is used to specify the NCU class. Profile type must be one of `ncu`, `loc`, or `enm`; NCU class must be one of `phys` or `ip`.

`list` [`-x`] [`-p` *profile-type*] [`-c` *ncu-class*] [*profile-name*]
List all available profiles and their current state. If a particular profile is specified by name, list only the current state of that profile. If the profile name is not unique, all profiles with the given name will be listed; or the profile type and/or NCU class can be included to identify a specific profile. If only a type is provided, list all profiles of that type. Listing the active NCP will include the NCUs that make up that NCP.

The `-x` option causes the `list` subcommand to display a fourth column of output, headed `AUXILIARY STATE`, after the first three column headings in the default display, `TYPE`, `PROFILE`, and `STATE`. The `AUXILIARY STATE` column shows why a profile is in a given state.

Possible `STATE` values are:

`disabled`
A manually-activated profile that has not been activated.

`offline`
A conditionally- or system-activated profile that has not been activated. It might not be active because its conditions have not been satisfied; or it might be that another profile has more specific conditions that are met and has been activated instead (in the case of profile types that must be activated one at a time, such as Locations).

`online`
A conditionally- or system-activated profile whose conditions have been met and that has been successfully activated; or a manually-activated profile that has been successfully activated at the request of the user.

`maintenance`
Activation of the profile was attempted, but failed.

`initialized`
The profile represents a valid configuration object for which no action has yet been taken.

`uninitialized`
The profile represents a configuration object not present in the system; for example, an NCU corresponding to a physical link that has been removed.

`show-events`
Listen for stream of events from the NWAM daemon and display them.

`scan-wifi` *linkname*
Initiate a wireless scan on link linkname.

`select-wifi` *linkname*
Select a wireless network to connect to from scan results on link linkname. Prompts for selection, WiFi key, and so forth, if necessary.

`help`
Display a usage message with short descriptions for each subcommand.

Examples

EXAMPLE 1 Enabling a User-Specified Location

The following command enables a user-specified location.

```
# netadm enable -p loc office
Disabled loc 'home'.
Enabled loc 'office'
```

EXAMPLE 2 Disabling an ENM

The following command disables an ENM.

```
# netadm disable -p enm myvpn
Disabled enm 'myvpn'.
```

EXAMPLE 3 Listing All NCPs

The following command lists all NCPs.

```
# netadm list -xp ncp
TYPE         PROFILE        STATE          AUXILIARY STATE
ncp          Automatic      disabled       disabled by administrator
ncp          User           online         active
ncu:phys     nge0           online         interface/link is up
ncu:ip       nge0           online         interface/link is up
ncu:phys     nge1           offline        interface/link is down
ncu:ip       nge1           offline        conditions for activation are
                                           unmet
```

EXAMPLE 4 Listing NCUs in Active NCP

The following command lists all `ip` NCUs in the active NCP.

```
# netadm list -c ip
TYPE        PROFILE        STATE
ncu:ip      bge0           online
ncu:ip      bge1           disabled
```

EXAMPLE 5 Forcing a Scan

The following command forces a scan on the wireless interface `wpi0`.

```
# netadm scan-wifi wpi0
```

EXAMPLE 6 Selecting a WiFi Network

The following command selects a WiFi network that is broadcasting its ESSID.

```
# netadm select-wifi wpi0
1: ESSID testing BSSID 0:40:96:29:e9:d8
2: ESSID sunwifi BSSID 0:b:e:9f:b5:80
3: ESSID sunwifi BSSID 0:b:e:85:26:c0
4: ESSID sunwifi BSSID 0:b:e:49:2f:80
5: Other

Choose WLAN to connect to [1-5]: 2
#
```

EXAMPLE 7 Selecting a WiFi Network (Alternative)

The following command selects a WiFi network that is not broadcasting its ESSID.

```
# netadm select-wifi wpi0
1: ESSID testing BSSID 0:40:96:29:e9:d8
2: ESSID sunwifi BSSID 0:b:e:85:26:c0
3: ESSID sunwifi BSSID 0:b:e:9f:b5:80
4: ESSID sunwifi BSSID 0:b:e:49:2f:80
5: ESSID sunwifi BSSID 0:b:e:49:62:c0
6: Other

Choose WLAN to connect to [1-6]: 6

Enter WLAN name: oraclewifi
1: None
2: WEP
3: WPA
Enter security mode: 2

Enter WLAN key for ESSID oraclewifi: 123456

Enter key slot [1-4]: 1
```

EXAMPLE 7 Selecting a WiFi Network (Alternative) *(Continued)*

```
#
```

EXAMPLE 8 Monitoring nwamd

The following command monitors nwamd(1M) when switching locations.

```
# netadm show-events
EVENT           DESCRIPTION
OBJECT_ACTION   loc Automatic -> action refresh
OBJECT_STATE    loc Automatic -> state offline*, method/service executi
OBJECT_STATE    loc Automatic -> state online, active
OBJECT_ACTION   loc home -> action refresh
OBJECT_ACTION   loc NoNet -> action refresh
OBJECT_ACTION   loc User -> action refresh
OBJECT_ACTION   loc home -> action enable
OBJECT_STATE    loc home -> state offline*, method/service executing
OBJECT_STATE    loc Automatic -> state offline, conditions for activati
OBJECT_STATE    loc home -> state online, active
^C
#
```

Attributes

See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os
Interface Stability	Committed

See Also

dladm(1M), ipadm(1M), netcfg(1M), netcfgd(1M), nwamd(1M), attributes(5)

See also nwam-manager(1M), available in the JDS/GNOME man page collection.

Name netcfg – create and modify network configuration profiles

Synopsis

```
netcfg
netcfg subcommand [options...]
netcfg [-d] -f command-file
netcfg help [subcommand]
```

Description The `netcfg` utility manipulates system network configuration profiles. `netcfg` can be invoked interactively, with an individual subcommand, or by specifying a command file that contains a series of subcommands.

The `netcfg` utility operates on several different types of configuration profiles:

- Network Configuration Profiles (NCPs)
- Locations
- External Network Modifiers (ENMs)
- Known WLANs

For more details on these profile types, refer to the "Profiles" section.

`netcfg` commands are performed within a scope. There are three scopes: global, profile, and NCP. When `netcfg` is invoked without any arguments, the editing session begins in the global scope. In the global scope, NCPs, Location and ENM profiles, and Known WLAN entries are available to operate on. Selecting an NCP will move the editing session to the NCP scope; from there, individual Network Configuration Units (NCUs) may be created or selected to move into the profile scope. Also, at the global scope, selecting or creating a Location, ENM, or Known WLAN will move the editing session to the profile scope.

Within a given profile scope, profile properties may be viewed and modified.

In interactive mode, changes are not stored to persistent storage until commit is invoked. Commit is implicitly invoked at "end" or "exit", or can be explicitly invoked by the user. When commit is invoked, the entire profile is committed. In order to maintain the consistency of persistent storage, the commit operation includes a verify step; if verification fails, the commit also fails. If an implicit commit fails, the user will be given the option of ending or exiting without committing the current changes, or remaining in the current scope to make further changes.

Profiles The NWAM service manages network configuration by storing desired property values in profiles. It then determines which profile should be active at a given time, depending on current network conditions, and activates that profile. In addition to the Network Configuration Profiles (NCPs) discussed in the previous section, `nwamd` also manages Location and ENM profiles.

Network Configuration Profiles (NCPs)

An NCP specifies the configuration of the local network components, including all datalinks and interfaces. These components are collectively referred to as Network Configuration Units, or NCUs.

NCPs are either 'reactive' or 'fixed'. Reactive NCPs include policy rules which determine when each NCU should be enabled. The policy may be set up such that the network configuration can change in response to changes in the network conditions. Fixed NCPs do not include these policy rules; their configuration is fully applied at the time the NCP is enabled, and will not be changed by the system, regardless of any failures encountered or changes in the network state.

A single fixed NCP, called `DefaultFixed`, is created by the system. On a newly installed system, this NCP will contain any network configuration that was created during installation, either during an interactive install or with a profile applied during Automated Installation. If no network configuration was specified during system installation, the `DefaultFixed` NCP will initially be empty.

The system also creates a single reactive NCP. The Automatic NCP is created and managed by `nwamd`, and cannot be modified by the user. This NCP consists of one link NCU and one interface NCU for each physical link present in the system. As links are added or removed from the system, their corresponding NCUs are added or removed from the Automatic NCP. The policy implemented in this NCP is to prefer wired links over wireless, and to plumb IP on all connected wired links, or one wireless link if no wired links are connected.

The Automatic NCP should not be modified by the user. It is managed by the system to match the current set of links and a specific configuration policy; the system may make change at any time which could overwrite or interfere with changes made by the user. If modifications are desired, a copy may be created and then modified; refer to the `create` subcommand and its `-t` option.

Finally, the user can create any number of additional reactive NCPs. These NCPs are managed entirely by the user; NCUs must be added or removed explicitly, and it is possible to add NCUs that do not map to any link currently installed in the system, or to remove NCUs that do map to a link present in the system. The user can determine the policy for these NCPs.

There must always be one active NCP. The active NCP on a newly installed system is determined by the installation method; it will either be the Automatic NCP (in the case of a LiveCD install or an Automated Install with a custom profile specifying the active NCP) or the `DefaultFixed` for other installation methods. The system will never change the active NCP. The user can do this at any time using the GUI or the `netadm(1M)` command.

Locations

A Location specifies system-wide network configuration, including name services, domain, IP Filter, and IPsec configuration.

External Network Modifiers (ENMs)

External Network Modifiers are, as the name suggests, applications external to the NWAM service that can modify and/or create network configuration. `nwamd` activates or deactivates an ENM depending on conditions that are specified as part of the ENM profile. Alternatively, the user might choose to manually activate/deactivate ENMs as needed.

While Location profiles allow a specific set of network-related services to be configured automatically based on current network conditions, that set of services is limited. ENMs provide additional flexibility, allowing the user to specify changes to SMF service properties and/or state, or any other system settings, to be applied under specific conditions.

Properties `netcfg` supports the following types of properties:

- NCP properties
- NCU properties
- properties of interface NCUs
- properties common to all link NCUs
- location properties
- ENM properties
- known WLAN properties

These properties are described in the following subsections.

|NCP Properties Each NCP has a single, read-only property.

`management-type`: enumerated value: `fixed` | `reactive`
Determines the way in which the NCP is managed by the system. Fixed NCPs have all configuration applied at the time the NCP is activated, with no subsequent system intervention. Reactive NCPs are more actively managed, with nwamd enforcing policy rules that determine when each link or interface should be configured.

The user cannot set or change the value of this property. All NCPs created using `netcfg` will have `management-type` set to reactive. There is only one fixed NCP, called `DefaultFixed`, which is created by the system.

NCU Properties The following properties are common to all NCUs.

`type`: enumerated value: *link* | *interface*
Specifies the NCU type, either *link* or *interface*. The value is implicitly determined based on the specified class.

`class`: enumerated value: *phys* for link NCUs; *ip* for interface NCU
Specifies the NCU class.

`parent`: string: *name of parent NCP*
Specifies the NCP of which this NCU is a component.

The `type`, `class`, and `parent` properties are set when the NCU is created and cannot be changed later.

`enabled`: boolean: `true` | `false`
If the activation-mode is `manual`, the enabled property reflects the NCU's current state. This property is read-only; it is changed indirectly by enabling or disabling the NCU using `netadm(1M)`.

The default value is `true`.

Properties of Interface NCUs

`ip-version`: list of enumerated values: `ipv4` | `ipv6`
The version(s) of IP that should be used on this NCU.

The default value is `ipv4,ipv6`.

`ipv4-addrsrc`: list of enumerated values: `dhcp` | `static`
Identifies the source of IPv4 addresses assigned to this NCU; multiple values may be assigned. If one of the values assigned is `static`, the `ipv4-addr` property must include at least one IPv4 address to be assigned to the NCU.

The default value is `dhcp`.

`ipv4-addr`: list of IPv4 address(es)
One or more IPv4 addresses to be assigned to this NCU.

`ipv4-default-route`:*IPv4 address*
The IPv4 address of the default router; if this property is set, a default route for IPv4 traffic will be associated with this interface when it is brought up.

`ipv6-addrsrc`: list of enumerated values: `dhcp` | `autoconf` | `static`
Identifies the source of IPv6 addresses assigned to this NCU; multiple values can be assigned. If one of the values assigned is `static`, the `ipv6-addr` property must include at least one IPv6 address to be assigned to the NCU.

The default value is `dhcp,autoconf`.

`ipv6-addr`: list of IPv6 address(es)
One or more IPv6 addresses to be assigned to this NCU.

`ipv6-default-route`: *IPv6 address*
The IPv6 address of the default router; if this property is set, a default route for IPv6 traffic will be associated with this interface when it is brought up.

Properties Common to All LINK NCUs

`activation-mode`: enumerated value: `manual` | `prioritized`
The type of trigger for automatic activation of this NCU.

The default value is `manual`.

`priority-group`: `uint64`: *group priority number*
If `activation-mode` is set to `prioritized`, this property specifies the priority group to which this NCU belongs. A group consists of one or more NCUs; smaller numbers are higher priority. The highest priority group that is determined to be available will be activated by nwamd(1M), and will remain so until it is no longer available, or until a higher priority group becomes available.

`priority-mode`: enumerated value: `exclusive` | `shared` | `all`
If `activation-mode` is set to prioritized, this property specifies the mode used to determine availability and activation behavior for a priority group.

`exclusive`
At least one NCU must be available to make the group available, and only one NCU will be activated. If more than one member NCU is available, `nwamd(1M)` will randomly choose one to activate.

`shared`
At least one NCU must be available to make the group available; all available NCUs will be activated.

`all`
All group member NCUs must be available to make the group available; all member NCUs will be activated.

`mac-address`: string: 48-bit MAC address
The MAC address assigned to this link. By default, NWAM will request the factory-assigned or default MAC address for the link; set a different value here to override that selection.

`autopush`: string: modules to be pushed over the link
Identifies a list of modules that should be automatically pushed over the link when it is opened. The string may contain more than one module name; module names are separated by the single character " . " (period).

`mtu`: `string`: MTU size for this link
This property will be automatically set to the default MTU for the physical link; that value may be overridden by setting this property to a different value.

Note that the range of allowed MTU values depends on the underlying hardware. Because NCUs may be created before the underlying hardware is present with driver attached, it is not possible to verify the value set at the time the NCU is edited. If an NCU fails to activate because of an invalid MTU size, this will be indicated in the system log, and the NCU will be placed in maintenance state.

Location Properties

`activation-mode`: enumerated value: `manual` | `system` | `conditional-all` | `conditional-any`
The type of trigger for automatic activation of this location.

The default value is `manual`.

`enabled`: boolean: `true` | `false`
If the activation-mode is `manual`, the enabled property reflects the location's current state. This property is read-only; it is changed indirectly by enabling or disabling the location using `netadm(1M)`.

The default value is `false`.

`conditions`: list of strings: *conditional expressions*
If `activation-mode` is set to `conditional-all` or `conditional-any`, this property specifies the test to determine whether this location should be activated. The conditional

expression is made up of a sequence of conditions that can be assigned a boolean value, such as "advertised-domain is sun.com" or "ip:bge0 is-not active." The format of these expressions is defined in the "Condition Expressions" section, below. If multiple conditions are specified, either all must be true to meet the activation requirements (when `activation-mode` is `conditional-all`) or any one may be true (when `activation-mode` is `conditional-any`).

Note the distinction between `advertised-domain` and `system-domain`. The advertised domain is learned by means of external communication, such as the `DNSdmain` or `NISdmain` advertised by a DHCP server. This attribute is useful for conditional activation of locations; for example, if the advertised domain is `mycompany.com`, activate the "work" location. The system domain is the domain which is currently assigned to the system; that is, it is the value returned by the `domainname(1M)` command. This attribute is useful for conditional activation of ENMs, as it will only become true after a location has been activated and the system is configured for that particular domain.

`default-domain`: string: *system domain name*
The domain name that should be applied to the system; this domain name will be used by NIS and LDAP.

`nameservices`: enum value list: `files` | `dns` | `nis` | `ldap`
Specifies the name services that should be configured, such as DNS, NIS, and LDAP.

`nameservices-config-file`: string: *path to* `nsswitch.conf` *file*
Specifies the `nsswitch.conf` file to be used. This property must always have a value. If the `nameservices` property specifies a single name service, this property will, by default, contain `/etc/nsswitch.`*nameservice* for the name service specified in the `nameservices` property; this value can be changed by the user. If the `nameservices` property identifies more than one name service, the user must specify an additional value for this property.

`dns-nameservice-configsrc`: enum value list: `manual` | `dhcp`
Specifies the source(s) that should be used to obtain configuration information for the DNS name service. If `manual` is included, the remaining `dns-nameservice-*` properties will be used.

`dns-nameservice-domain`: string: *domain name*
`dns-nameservice-servers`: string list: *name server address(es)*
`dns-nameservice-search`: string list: *domain search string*
`dns-nameservice-sortlist`: string list: *IP address*, *netmask pair(s)*
`dns-nameservice-options`: string list: *resolver variable(s) to be modified*
If DNS is one of the configured name services and `manual` is one of its configuration sources, these properties specify its configuration parameters. Only the `servers` property is required; `search` and `domain` are mutually exclusive and only one can be specified, which will override the domain provided by the DHCP server if `dhcp` is also used. The format of these values are the same as the respective options in `resolv.conf(4)`.

`nis-nameservice-configsrc`: enum value list: `manual` | `dhcp`
Specifies the source(s) that should be used to obtain configuration information for the NIS name service. If `manual` is included, the remaining `nis-nameservice-*` properties will be used.

`nis-nameservice-servers`: string list: *name server address(es)*
If NIS is one of the configured name services and `manual` is one of its configuration sources, this property specifies its server address. If this property is not specified, the NIS client will be started in broadcast mode.

`ldap-nameservice-configsrc`: enum value list: `manual`
Specifies the source that should be used to obtain configuration information for the LDAP name service. `manual` is currently the only option for LDAP; thus the `ldap-nameservice-*` properties must be provided to complete LDAP configuration.

`ldap-nameservice-servers`: string list: *name server address(es)*
If LDAP is one of the configured name services, this property specifies its server address. This property is required, and it is expected that the specified server will have a client profile which will be used to complete client configuration.

`nfsv4-domain`: string: *domain name to be used for NVSv4*
Specifies the NVSv4 domain to be used. If this value is unspecified, the name service domain will be used.

`ipfilter-config-file`: string: *path to ipfilter IPv4 configuration file*
`ipfilter-v6-config-file`: string: *path to ipfilter IPv6*
`ipnat-config-file`: string: *path to* `ipnat` *configuration file*
`ippool-config-file`: string:*path to* `ippool` *configuration file*
These properties each specify the path to the rules file for a component of the `ipfilter(5)` configuration. If a given `config-file` property is set, the corresponding IP filter component will be enabled, reading configuration from the specified file.

`ike-config-file`: string: *path to IKE configuration file*
Specifies the IKE configuration file. If a value is specified, the `svc:/network/ipsec/ike` service will be enabled, reading configuration from the specified file.

`ipsecpolicy-config-file`: string: *path to IPsec policy configuration file*
Specifies the IPsec policy configuration file. If a value is specified, the `svc:/network/ipsec/policy` service will be enabled, reading configuration from the specified file.

ENM Properties

`activation-mode`: enumerated value: `manual` | `conditional-all` | `conditional-any`
The type of trigger for automatic activation of this ENM.

The default value is `manual`.

`enabled`: boolean: `true` | `false`
If the activation-mode is `manual`, the enabled property reflects the ENM's current state. This property is read-only; it is changed indirectly by enabling or disabling the ENM using `netadm(1M)`.

The default value is `false`.

`conditions`: list of strings: *conditional expressions*
If activation-mode is set to `conditional-all` or `conditional-any`, this property specifies the test to determine whether or not this ENM should be activated. The conditional expression is made up of a sequence of conditions that can be assigned a boolean value, such as "system-domain is sun.com" or "ip:bge0 is-not active." The format of these expressions is defined in the "Condition Expressions" section below. If multiple conditions are specified, either all must be true to meet the activation requirements (when `activation-mode` is `conditional-all`) or any one may be true (when `activation-mode` is `conditional-any`).

Note the distinction between `advertised-domain` and `system-domain`. The advertised domain is learned by means of external communication, such as the `DNSdmain` or `NISdmain` advertised by a DHCP server. This attribute is useful for conditional activation of locations; for example, if the advertised domain is `mycompany.com`, activate the "work" location. The system domain is the domain which is currently assigned to the system; that is, it is the value returned by the |domainname|(1M) command. This attribute is useful for conditional activation of ENMs, as it will only become true after a location has been activated and the system configured for that particular domain.

`fmri`: string: *service FMRI*
If this ENM is implemented as an SMF service, this property identifies that service. If this property is specified, the ENM will be activated by enabling the service and deactivated by disabling the service.

`start`: string: *start command*
If this ENM is not implemented as an SMF service, this property identifies the command that should be exec'ed to start or activate the ENM. This property will be ignored if the FMRI property is set.

`stop`: string: *stop command*
If this ENM is not implemented as an SMF service, this property identifies the command that should be exec'ed to stop/deactivate the ENM. This property will be ignored if the FMRI property is set.

Known WLAN Properties

`bssids`: list of strings: WLAN BSSID(s) (AP MAC addresses)
If a specific Access Point should be preferred over others with the same name/ESSID, this property allows the AP's BSSID to be specified.

`priority`: `uint64`: a numeric priority value
The relative priority of this WLAN; a smaller number represents higher priority.

Note that this number can be changed if subsequent changes to the set of Known WLAN objects require such a change. For example, consider a system that is configured with two Known WLAN objects, `wlanA` and `wlanB`. `wlanA` has priority 1, and `wlanB` has priority 2. A new Known WLAN, `wlanC`, is created and assigned priority 2. In this case, the complete list will be updated such that `wlanA` has priority 1, `wlanC` has priority 2, and `wlanB` has priority 3. No two WLANs can have the same priority value, so the addition of `wlanC` at priority 2 forces `wlanB` to be shifted down in priority. If any additional WLANs existed at lower priorities than `wlanB`, they would be shifted accordingly.

`keyslot`: `uint64`: the key slot to be used for this WLAN
If the WLAN uses an encryption mode that supports multiple keyslots, the slot to place the key can be specified in this property. If unspecified, slot 1 is used by default.

`keyname`: list of strings: Secure object name(s)
Allows the user to associate secure objects created using `dladm(1M)` with Known WLANs.

Condition Expressions

Locations and ENMs can be activated based on a set of user-specified conditions. The following table summarizes the syntax of those condition expressions.

Object Type	Object	Condition
ncp|ncu|enm|loc	name	is/is-not active

```
Object Type       Condition                                 Object
-----------------------------------------------------------
essid             is/is-not/contains/does-not-contain name string
bssid             is/is-not                           bssid string
ip-address        is/is-not                           IPv4 or IPv6
                                                      address
ip-address        is-in-range/is-not-in-range         IPv4 or IPv6
                                                      address plus
                                                      netmask/prefixlen
advertised-domain is/is-not/contains/does-not-contain name string
system-domain     is/is-not/contains/does-not-contain name string
```

Options

The following options are not associated with any particular `netcfg` subcommand:

`-d`
Removes all configuration before reading subcommands from the command file (see following option).

`-f` *command_file*
Reads and executes `netcfg` subcommands from *command_file*.

Sub-commands

The following subcommands are supported.

`cancel`
End the current profile specification without committing the current changes to persistent storage, and pop up to the next higher scope.

This subcommand is valid in the NCP and profile scopes.

`clear` *prop-name*
Clear the value for the specified property.

This subcommand is valid in the profile scope.

`commit`
Commit the current profile to persistent storage. Because a configuration must be correct to be committed, this operation automatically performs a verify on the object as well. The commit operation is attempted automatically upon leaving the current scope (with either the `end` or `exit` subcommand).

This subcommand is valid in the profile scope.

Note that, in non-interactive mode, a commit is not required, as the commit is implicit for any subcommand that changes a value.

`create` [`-t` *template*] *object-type* [*class*] *object-name*
Create an in-memory profile of the specified type and name. The `-t` *template* option specifies that the new object should be identical to *template*, where *template* is the name of an existing object of the same type. If the `-t` option is not used, the new object is created with default values. For NCPs, only a "user" NCP can be created.

This subcommand is valid in the global and NCP scopes.

`destroy` { `-a` | *object-type* [*class*] *object-name* }
Remove all or the specified profile from memory and persistent storage. This action is immediate; it does not need to be committed. A destroyed object cannot be reverted.

This subcommand is valid in the global and NCP scopes if a specific object is specified; the `-a` option is only valid in the global scope.

`end`
End the current profile specification, and pop up to the next higher scope. The current object is verified and committed before ending; if either the verify or commit fails, an appropriate error message is issued and the user is given the opportunity to end without committing the current changes, or to remain in the current scope and continue editing.

This subcommand is valid in any scope.

`exit`
Exit the `netcfg` session. The current profile is verified and committed before ending; if either fails, an appropriate error message is issued and the user is given the opportunity to exit without committing the current changes, or to remain in the current scope and continue editing.

This subcommand is valid in any scope.

`export` [`-d`] [`-f` *output-file*] [*object-type* [*class*] *object-name*]
Print the current configuration at the current or specified scope to standard out, or to a file specified with the `-f` option. The `-d` option generates a `destroy -a` as the first line of output. This subcommand produces output in a form suitable for use in a command file. System created objects, including the Automatic NCP and the Automatic, NoNet, and User locations, cannot be exported.

This subcommand is limited in that it can only export configuration that can be created using the `netcfg` command; an NCP may contain NCUs or NCU properties that can only be set using `ipadm`(1M) or `dladm`(1M). This sort of configuration will be omitted from what is output by the `export` subcommand.

This subcommand is valid in any scope.

`get` [`-V`] *prop-name*
Get the current (in-memory) value of the specified property. By default, both the property name and value are printed; if the `-V` option is specified, only the property value is displayed.

This subcommand is valid in the profile scope.

`help` [*subcommand*]
Display general help or help about a specific subcommand.

This subcommand is valid in any scope.

`list` [`-a`] [*object-type* [*class*] *object-name*]
List all profiles, property-value pairs and resources that exist at the current or specified scope. When listing properties of an object, the default behavior is to only list properties that apply to the specified configuration. That is, if listing an IP NCU for which `ipv4-addrsrc` is `dhcp`, the `ipv4-addr` property will not be listed. Including the `-a` option will result in all properties being listed, whether or not they apply to the current settings.

This subcommand is valid in any scope.

`revert`
Delete any current changes to the current profile and revert to the values from persistent storage.

This subcommand is valid in the profile scope.

`select` *object-type* [*class*] *object-name*
Select one of the profiles available at the current scope level and jump down into that object's scope. The selected object will be loaded into memory from persistent storage.

This subcommand is valid in the global and NCP scopes.

`set` *prop-name=value1*[,*value2*...]
Set the current (in-memory) value of the specified property. If performed in non-interactive mode, the change is also committed to persistent storage.

The delimiter for values of multi-valued properties is "`,`" (comma). If any of the individual values in such a property contains a comma, it must be escaped (that is, written as `\,`). Commas within properties that can only have a single value are not interpreted as delimiters and need not be escaped.

This subcommand is valid in the profile scope.

`verify`
Verify that the current in-memory object has a valid configuration.

This subcommand is valid in the profile scope.

`walkprop` [`-a`]
Walk each property associated with the current profile. For each property, the name and current value are displayed, and a prompt is given to allow the user to change the current value.

The delimiter for values of multi-valued properties is "," (comma). If any of the individual values in such a property contains a comma, it must be escaped (that is, written as `\,`). Commas within properties that can only have a single value are not interpreted as delimiters and need not be escaped.

By default, only properties that are required based on properties that are already set will be walked; that is, if `ipv4-addrsrc` is set to `dhcp`, `ipv4-addr` will not be walked. Including the `-a` option will result in all available properties being walked.

This subcommand is valid in the profile scope.

This subcommand is only meaningful in interactive mode.

Examples

EXAMPLE 1 Setting an NCU Property

The following command sets an NCU property from the command line (that is, in non-interactive mode).

```
# netcfg "select ncp User; select ncu phys net1; set mtu=1492"
```

EXAMPLE 2 Listing Top-Level Profiles

The following command lists all top-level profiles from the command line.

```
# netcfg list
NCPs:
        Automatic
        User
Locations:
        Automatic
        home
        NoNet
        office
```

EXAMPLE 2 Listing Top-Level Profiles *(Continued)*

```
ENMs:
        enmtest
        myenm
WLANs:
        sunwifi
        coffeeshop
        linksys
```

EXAMPLE 3 Destroying a Location Profile

The following command destroys a Location profile from the command line.

```
# netcfg destroy loc home
Destroyed loc 'home'
```

EXAMPLE 4 Creating an NCU Profile

The following command sequence interactively creates an NCU profile.

```
# netcfg
netcfg> select ncp user
netcfg:ncp:User> create ncu ip net1
Created ncu 'net1'.  Walking properties ...
ip-version (ipv4,ipv6) [ipv4|ipv6]> ipv4
ipv4-addrsrc (dhcp) [dhcp|static]> static
ipv4-addr> 168.1.2.3
netcfg:ncp:User:ncu:net1> list
ncu:net1
        type:             interface
        class:            ip
        parent:           "User"
        enabled           true
        ip version:       ipv4
        ipv4-addrsrc:     static
        ipv4-addr:        "168.1.2.3"
        ipv6-addrsrc      dhcp,autoconf
netcfg:ncp:User:ncu:net1> commit
Committed changes
netcfg:ncp:User:ncu:net1> end
netcfg:ncp:User> exit
```

EXAMPLE 5 Manipulating an ENM

The following command sequence selects an existing ENM, display its contents, and changes a property value.

```
# netcfg
netcfg>select enm myenm
netcfg:enm:myenm>list
```

EXAMPLE 5 Manipulating an ENM *(Continued)*

```
ENM:myenm
        activation-mode manual
        enabled         true
        start           "/usr/local/bin/myenm start"
        stop            "/usr/local/bin/myenm stop"
netcfg:enm:myenm>set stop="/bin/alt_stop"
netcfg:enm:myenm>list
ENM:myenm
        activation-mode manual
        enabled         true
        start           "/usr/local/bin/myenm start"
        stop            "/bin/alt_stop"
netcfg:enm:myenm>exit
Committed changes
```

EXAMPLE 6 Configuring a Static Address

The following command sequence configures a static address of 192.168.2.12/24 on interface `bge0` using the Home NCP. This interface is enabled by default when the Home NCP is activated.

```
# netcfg
netcfg> create ncp Home
```

First configure phys NCU:

```
netcfg:ncp:Home> create ncu phys bge0
Created ncu 'bge0'.  Walking properties ...
activation-mode (manual) [manual|prioritized]>
mac-address>
autopush>
mtu>
netcfg:ncp:Home:ncu:bge0> end
Committed changes
netcfg:ncp:Home>
```

Then, configure IP NCU:

```
netcfg:ncp:Home> create ncu ip bge0
Created ncu 'bge0'.  Walking properties ...
ip-version (ipv4,ipv6) [ipv4|ipv6]>
ipv4-addrsrc (dhcp) [dhcp|static]> static
ipv4-addr> 192.168.2.10/24
ipv4-default-route>
ipv6-addrsrc (dhcp,autoconf) [dhcp|autoconf|static]>
ipv6-default-route>
netcfg:ncp:Home:ncu:bge0> list
ncu:bge0
```

EXAMPLE 6 Configuring a Static Address *(Continued)*

```
    type                interface
    class               ip
    parent              "Home"
    enabled             true
    ip-version          ipv4,ipv6
    ipv4-addrsrc        static
    ipv4-addr           "192.168.2.10/24"
    ipv6-addrsrc        dhcp,autoconf
netcfg:ncp:Home:ncu:bge0> exit
Committed changes
#
```

Switch to the Home NCP using `netadm(1M)`.

EXAMPLE 7 Configuring Location Based on a Condition

The following command sequences configures a location based on a condition. The location is activated whenever the IP address is in the 10.0.8.0/24 subnet. Also, NIS is configured in this location.

```
netcfg> select loc office
netcfg:loc:office> list
loc:office
       activation-mode                conditional-any
       conditions                     "ip-address is 10.0.8.0/24"
       enabled                        false
       nameservices                   dns,nis
       nameservices-config-file       "/etc/nsswitch.nis"
       dns-nameservice-configsrc      dhcp
       nis-nameservice-configsrc      manual
       nis-nameservice-servers        "10.2.18.24"
       default-domain                 "labs.east.sun.com"
netcfg:loc:office>
```

EXAMPLE 8 Creating a Known WLAN

The following command sequence establishes a known WLAN named `coffeeshop` with a WEP key.

```
netcfg> select wlan coffeeshop
netcfg:wlan:coffeeshop> list
known wlan:coffeeshop
       priority               2
       keyname                "foo"
       security-mode          wep
netcfg:wlan:coffeeshop> set priority=1
netcfg:wlan:coffeeshop> end
Committed changes
```

EXAMPLE 9 Exporting Current Configuration to a File

The following command exports the current configuration to a file.

```
netcfg> export -f /tmp/nwam.config
```

Or, perform the same task from the Unix command line:

```
# netcfg export -f /tmp/nwam.config
```

EXAMPLE 10 Importing Current Configuration from a File

The following command imports the current configuration from a file.

```
# netcfg -f /tmp/nwam.config
```

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os
Interface Stability	Committed

See Also

`dladm(1M)`, `domainname(1M)`, `ipadm(1M)`, `netadm(1M)`, `netcfgd(1M)`, `nwamd(1M)`, `resolv.conf(4)`, `attributes(5)`, `ipfilter(5)`

See also `nwam-manager(1M)`, available in the JDS/GNOME man page collection.

Name netcfgd – network configuration management daemon

Synopsis `/lib/inet/netcfgd`

Description `netcfgd` is a system daemon to manage network configuration. This daemon is started automatically by svc:/network/netcfg:default and should not be invoked directly. It does not constitute a programming interface.

`netcfgd` manages access to a network configuration repository, insuring that readers and writers have the appropriate authorizations. The required authorizations vary, depending on the portion of the database being accessed.

Tool	Read/Write	Authorization (or Privilege)
dladm	write	sys_dl_config privilege
ipadm	write	solaris.network.interface.config
netcfg	read	solaris.network.autoconf.read
netcfg	write	solaris.network.autoconf.write
netcfg	r/w wlans	solaris.network.autoconf.wlan
netadm	enable/disable profiles	solaris.network.autoconf.read solaris.network.autoconf.select

The `solaris.network.interface.config` authorization and the `sys_dl_config` privilege are obtained by means of the Network Management security profile. The `solaris.network.autoconf.read` authorization is part of the Basic Solaris User profile. The `solaris.network.autoconf.wlan` and `.select` authorizations are obtained by means of the Network Autoconf User profile, and the `autoconf.write` authorization, plus those included in the Network Autoconf User profile, are obtained by means of the Network Autoconf Admin profile.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os
Interface Stability	Committed

See Also `svcs(1)`, `dladm(1M)`, `ipadm(1M)`, `netcfg(1M)`, `nwamd(1M)`, `svcadm(1M)`, `auth_attr(4)`, `exec_attr(4)`, `prof_attr(4)`, `attributes(5)`, `smf(5)`

See also `nwam-manager(1M)`, available in the JDS/GNOME man page collection.

Name netservices – enable or disable network services

Synopsis

```
netservices open

netservices limited
```

Description The `netservices` command uses the Solaris service management facility, `smf(5)`, to control services that accept over the network from remote clients.

When `netservices` is invoked with the `limited` command-line argument, all network services except the secure shell daemon, sshd(1M), are either disabled or constrained to respond to local requests only.

Invoking netservices with the `open` command-line argument enables a large set of network services, as in previous releases of Solaris.

To customize the configuration set enabled by `netservices`, use `svcadm(1M)` to enable or disable individual services. Use `svccfg(1M)` to set properties that determine whether a service accepts input from remote clients. See the man pages for individual services for the names of service instances and their properties.

Note that the `netservices` command has an interface stability of Obsolete.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os
Interface Stability	Obsolete

See Also `svcadm(1M)`, `svccfg(1M)`, `sshd(1M)`, `attributes(5)`, `smf(5)`

Name netstat – show network status

Synopsis

```
netstat [-anvR] [-f address_family] [-P protocol]

netstat -g [-nv] [-f address_family]

netstat -p [-n] [-f address_family]

netstat -s [-f address_family] [-P protocol]
     [-T u | d ] [interval [count]]

netstat -m [-T u | d ] [-v] [interval [count]]

netstat -i [-I interface] [-an] [-f address_family]
     [-T u | d ] [interval [count]]

netstat -r [-anvR] [-f address_family | filter]

netstat -M [-ns] [-f address_family]

netstat -D [-I interface] [-f address_family]

netstat -d [-f address_family]
```

Description The `netstat` command displays the contents of certain network-related data structures in various formats, depending on the options you select.

The `netstat` command has the several forms shown in the SYNOPSIS section, above, listed as follows:

- The first form of the command (with no required arguments) displays a list of active sockets for each protocol.
- The second, third, and fourth forms (`-g`, `-p`, and `-s` options) display information from various network data structures.
- The fifth form (`-m` option) displays STREAMS memory statistics.
- The sixth form (`-i` option) shows the state of the interfaces.
- The seventh form (`-r` option) displays the routing table.
- The eighth form (`-M` option) displays the multicast routing table.
- The ninth form (`-D` option) displays the state of DHCP on one or all interfaces.
- The tenth form (`-d` option) displays the table of destination cache entries.

These forms are described in greater detail below.

With no arguments (the first form), `netstat` displays connected sockets for `PF_INET`, `PF_INET6`, and `PF_UNIX`, unless modified otherwise by the `-f` option.

Options `-a`

Show the state of all sockets, all routing table entries, or all interfaces, both physical and logical. Normally, listener sockets used by server processes are not shown. Under most conditions, only interface, host, network, and default routes are shown and only the status of physical interfaces is shown.

`-d`

Show the destination cache entry table. See `DISPLAYS`, below.

`-f` *address_family*

Limit all displays to those of the specified *address_family*. The value of *address_family* can be one of the following:

`inet` For the `AF_INET` address family showing IPv4 information.

`inet6` For the `AF_INET6` address family showing IPv6 information.

`unix` For the `AF_UNIX` address family.

`sdp` For the Socket Description Protocol (SDP) protocol and address family. The address state displayed for an SDP socket are listed below. Flags displayed by `netstat` are followed by their meanings.

`LST` Listen
`EST` Established
`PL` Path Lookup
`HS` Hello Request Sent
`HR` Hello Request Received
`HAR` Hello Ack Recvd
`HAS` Hello Ack sent
`DR` Fin received
`DS` Fin sent
`DSA` Fin Ack recvd
`DRC` Simultaneous Disconnect
`DSC` Disconnect sent (peer already closed)
`TW1` Time Wait 1
`TW2` Time Wait 2
`CLD` Closed
`ERR` Error
`INV` Invalid
`UNK` Unknown

For the SDP protocol and address family, `netstat` displays the following column headings:

`Local Address` Local IP address

`Remote Address`	Remote IP address
`State`	Current state of the socket
`RxBPending`	Bytes unread
`TxBQueued`	Bytes queued for Tx (includes `TxBPosted`)
`TxBPosted`	Bytes sent to HW for transmission
`LAdvtSz`	Local advertised buffer size
`RAdvtSz`	Remote advertised buffer size
`LAdvtBuff`	Number of local advertised Rx buffers
`RAdvtBuff`	Number of remote advertised Rx buffers
`LPostBuff`	Number of Rx buffers currently posted

`-f` *filter*

With `-r` only, limit the display of routes to those matching the specified filter. A filter rule consists of a *keyword*:*value* pair. The known keywords and the value syntax are:

`af:`{`inet`|`inet6`|`unix`|`sdp`|*number*}	Selects an address family. This is identical to `-f` *address_family* and both syntaxes are supported.
`outif:`{*name*|*ifIndex*|`any`|`none`}	Selects an output interface. You can specify the interface by name (such as `hme0`) or by `ifIndex` number (for example, `2`). If `any` is used, the filter matches all routes having a specified interface (anything other than null). If `none` is used, the filter matches all routes having a null interface. Note that you can view the index number (*ifIndex*) for an interface with the `-a` option of `ifconfig(1M)`.
`dst:`{*ip-address*[/*mask*]|`any`|`none`}	Selects a destination IP address. If specified with a mask length, then any routes with matching or longer (more specific) masks are selected. If `any` is used, then all but addresses but 0 are selected. If `none` is used, then address 0 is selected.
`flags:`[`+ -`]?[`ABDGHLMSU`]+	Selects routes tagged with the specified flags. By default, the flags as specified must be set in order to match. With a leading `+`, the flags specified must be set but others are ignored. With a leading `-`, the flags specified must not be set and others are permitted.

You can specify multiple instances of `-f` to specify multiple filters. For example:

```
% netstat -nr -f outif:hme0 -f outif:hme1 -f dst:10.0.0.0/8
```

The preceding command displays routes within network 10.0.0.0/8, with mask length 8 or greater, and an output interface of either `hme0` or `hme1`, and excludes all other routes.

`-g`
Show the multicast group memberships for all interfaces. If the `-v` option is included, source-specific membership information is also displayed. See DISPLAYS, below.

`-i`
Show the state of the interfaces that are used for IP traffic. Normally this shows statistics for the physical interfaces. When combined with the `-a` option, this will also report information for the logical interfaces. See `ifconfig(1M)`.

`-m`
Show the STREAMS memory statistics.

`-n`
Show network addresses as numbers. `netstat` normally displays addresses as symbols. This option may be used with any of the display formats.

`-p`
Show the net to media tables. See DISPLAYS, below.

`-r`
Show the routing tables. Normally, only interface, host, network, and default routes are shown, but when this option is combined with the `-a` option, all routes will be displayed, including cache. If you have not set up a multicast route, `-ra` might not show any multicast routing entries, although the kernel will derive such an entry if needed.

`-s`
Show per-protocol statistics. When used with the `-M` option, show multicast routing statistics instead. When used with the `-a` option, per-interface statistics will be displayed, when available, in addition to statistics global to the system. See DISPLAYS, below.

`-T u | d`
Display a time stamp.

Specify `u` for a printed representation of the internal representation of time. See `time(2)`. Specify `d` for standard date format. See `date(1)`.

`-v`
Verbose. Show additional information for the sockets, STREAMS memory statistics, routing table, and multicast group memberships.

`-I` *interface*
Show the state of a particular interface. *interface* can be any valid interface such as `hme0` or `eri0`. Normally, the status and statistics for physical interfaces are displayed. When this option is combined with the `-a` option, information for the logical interfaces is also reported.

`-M`
Show the multicast routing tables. When used with the `-s` option, show multicast routing statistics instead.

`-P` *protocol*
Limit display of statistics or state of all sockets to those applicable to *protocol*. The protocol can be one of `ip`, `ipv6`, `icmp`, `icmpv6`, `icmp`, `icmpv6`, `igmp`, `udp`, `tcp`, `rawip`. `rawip` can also be specified as `raw`. The command accepts protocol options only as all lowercase.

`-D`
Show the status of DHCP configured interfaces.

`-R`
This modifier displays extended security attributes for sockets and routing table entries. The `-R` modifier is available only if the system is configured with the Solaris Trusted Extensions feature.

With `-r` only, this option displays the routing entries' gateway security attributes. See `route(1M)` for more information on security attributes.

When displaying socket information using the first form of the commmand, this option displays additional information for Multi-Level Port(MLP) sockets. This includes:

- The label for the peer if the socket is connected.
- The following flags can be appended to the socket's “State” output:

  `P` The socket is a MLP on zone-private IP addresses.

  `S` The socket is a MLP on IP addresses shared between zones.

Operands

interval Display statistics accumulated since last display every *interval* seconds, repeating forever, unless *count* is specified. When invoked with *interval*, the first row of netstat output shows statistics accumulated since last reboot.

The following options support *interval*: `-i`, `-m`, `-s` and `-Ms`. Some values are configuration parameters and are just redisplayed at each interval.

count Display interface statistics the number of times specified by *count*, at the interval specified by *interval*.

Displays

Active Sockets (First Form)

The display for each active socket shows the local and remote address, the send and receive queue sizes (in bytes), the send and receive windows (in bytes), and the internal state of the protocol.

The symbolic format normally used to display socket addresses is either:

`hostname.`*port*

when the name of the host is specified, or

network.*port*

if a socket address specifies a network but no specific host.

The numeric host address or network number associated with the socket is used to look up the corresponding symbolic hostname or network name in the *hosts* or *networks* database.

If the network or hostname for an address is not known, or if the `-n` option is specified, the numerical network address is shown. Unspecified, or "wildcard", addresses and ports appear as an asterisk (`*`). For more information regarding the Internet naming conventions, refer to `inet(7P)` and `inet6(7P)`.

For SCTP sockets, because an endpoint can be represented by multiple addresses, the verbose option (`-v`) displays the list of all the local and remote addresses.

TCP Sockets The possible state values for TCP sockets are as follows:

`BOUND`	Bound, ready to connect or listen.
`CLOSED`	Closed. The socket is not being used.
`CLOSING`	Closed, then remote shutdown; awaiting acknowledgment.
`CLOSE_WAIT`	Remote shutdown; waiting for the socket to close.
`ESTABLISHED`	Connection has been established.
`FIN_WAIT_1`	Socket closed; shutting down connection.
`FIN_WAIT_2`	Socket closed; waiting for shutdown from remote.
`IDLE`	Idle, opened but not bound.
`LAST_ACK`	Remote shutdown, then closed; awaiting acknowledgment.
`LISTEN`	Listening for incoming connections.
`SYN_RECEIVED`	Initial synchronization of the connection under way.
`SYN_SENT`	Actively trying to establish connection.
`TIME_WAIT`	Wait after close for remote shutdown retransmission.

SCTP Sockets The possible state values for SCTP sockets are as follows:

`CLOSED`	Closed. The socket is not being used.
`LISTEN`	Listening for incoming associations.
`ESTABLISHED`	Association has been established.
`COOKIE_WAIT`	`INIT` has been sent to the peer, awaiting acknowledgment.

COOKIE_ECHOED	State cookie from the INIT-ACK has been sent to the peer, awaiting acknowledgement.
SHUTDOWN_PENDING	SHUTDOWN has been received from the upper layer, awaiting acknowledgement of all outstanding DATA from the peer.
SHUTDOWN_SENT	All outstanding data has been acknowledged in the SHUTDOWN_SENT state. SHUTDOWN has been sent to the peer, awaiting acknowledgement.
SHUTDOWN_RECEIVED	SHUTDOWN has been received from the peer, awaiting acknowledgement of all outstanding DATA.
SHUTDOWN_ACK_SENT	All outstanding data has been acknowledged in the SHUTDOWN_RECEIVED state. SHUTDOWN_ACK has been sent to the peer.

Network Data Structures (Second Through Fifth Forms)

The form of the display depends upon which of the `-g`, `-m`, `-p`, or `-s` options you select.

`-g` Displays the list of multicast group membership.

`-m` Displays the memory usage, for example, STREAMS mblks.

`-p` Displays the net to media mapping table. For IPv4, the address resolution table is displayed. See `arp(1M)`. For IPv6, the neighbor cache is displayed.

`-s` Displays the statistics for the various protocol layers.

The statistics use the MIB specified variables. The defined values for `ipForwarding` are:

forwarding(1)	Acting as a gateway.
not-forwarding(2)	Not acting as a gateway.

The IPv6 and ICMPv6 protocol layers maintain per-interface statistics. If the `-a` option is specified with the `-s` option, then the per-interface statistics as well as the total sums are displayed. Otherwise, just the sum of the statistics are shown.

For the second, third, and fourth forms of the command, you must specify at least `-g`, `-p`, or `-s`. You can specify any combination of these options. You can also specify `-m` (the fifth form) with any set of the `-g`, `-p`, and `-s` options. If you specify more than one of these options, `netstat` displays the information for each one of them.

Interface Status (Sixth Form)

The interface status display lists information for all current interfaces, one interface per line. If an interface is specified using the `-I` option, it displays information for only the specified interface.

The list consists of the interface name, `mtu` (maximum transmission unit, or maximum packet size)(see `ifconfig(1M)`), the network to which the interface is attached, addresses for each interface, and counter associated with the interface. The counters show the number of input

packets, input errors, output packets, output errors, and collisions, respectively. For Point-to-Point interfaces, the Net/Dest field is the name or address on the other side of the link.

If the `-a` option is specified with either the `-i` option or the `-I` option, then the output includes names of the physical interface(s), counts for input packets and output packets for each logical interface, plus additional information.

If the `-n` option is specified, the list displays the IP address instead of the interface name.

If an optional *interval* is specified, the output will be continually displayed in *interval* seconds until interrupted by the user or until *count* is reached. See OPERANDS.

The physical interface is specified using the `-I` option. When used with the *interval* operand, output for the `-I` option has the following format:

```
input    eri0          output        input          (Total)   output
packets  errs  packets errs  colls   packets  errs  packets  errs   colls
227681   0     659471  1     502     261331   0     99597    1      502
10       0     0       0     0       10       0     0        0      0
8        0     0       0     0       8        0     0        0      0
10       0     2       0     0       10       0     2        0      0
```

If the input interface is not specified, the first interface of address family `inet` or `inet6` will be displayed.

Routing Table (Seventh Form)

The routing table display lists the available routes and the status of each. Each route consists of a destination host or network, and a gateway to use in forwarding packets. The *flags* column shows the status of the route. These flags are as follows:

`U` Indicates route is `up`.

`G` Route is to a gateway.

`H` Route is to a host and not a network.

`M` Redundant route established with the `-multirt` option.

`S` Route was established using the `-setsrc` option.

`D` Route was created dynamically by a redirect.

`B` Packets will be silently dropped (`RTF_BLACKHOLE` set).

`R` Packets will be dropped with ICMP error sent (`RTF_REJECT` set).

`I` Indirect routes (gateway not directly reachable) established with the `-indirect` option.

`Z` (non-global exclusive-IP zone only) The route was statically added on boot based on routing information configured using `zonecfg(1M)` in the global zone.

If the `-a` option is specified, there will be routing entries with the following flags:

`b` Broadcast addresses.

`C` Clones interface host route entries for on-link destinations.

`L` Local addresses for the host.

Interface routes are created for each interface attached to the local host; the gateway field for such entries shows the address of the outgoing interface.

The `use` column displays the number of packets sent or forwarded using the route in question.

The *interface* entry indicates the network interface utilized for the route.

Multicast Routing Tables (Eighth Form)

The multicast routing table consists of the virtual interface table and the actual routing table.

DHCP Interface Information (Ninth Form)

The DHCP interface information consists of the interface name, its current state, lease information, packet counts, and a list of flags.

The states correlate with the specifications set forth in *RFC 2131*.

Lease information includes:

- when the lease began;
- when lease renewal will begin; and
- when the lease will expire.

The flags currently defined include:

`BOOTP` The interface has a lease obtained through `BOOTP` (IPv4 only).

`BUSY` The interface is busy with a DHCP transaction.

`PRIMARY` The interface is the primary interface. See `dhcpinfo(1)` and `ifconfig(1M)`.

`FAILED` The interface is in failure state and must be manually restarted.

Packet counts are maintained for the number of packets sent, the number of packets received, and the number of lease offers declined by the DHCP client. All three counters are initialized to zero and then incremented while obtaining a lease. The counters are reset when the period of lease renewal begins for the interface. Thus, the counters represent either the number of packets sent, received, and declined while obtaining the current lease, or the number of packets sent, received, and declined while attempting to obtain a future lease.

Destination Cache Entry Table (Tenth Form)

The destination cache entry display shows the recorded path MTU, the age (in seconds) of the entry, and flags. The `P` flag indicates that a path MTU is recorded. The `S` flag indicates that the path MTU is smaller than the minumum that IP will allow. The `U` flag indicates that some transport metrics (round-trip time, and so forth) are cached in the destination cache entry.

Files `/etc/default/inet_type` `DEFAULT_IP` setting

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also `arp(1M)`, `dhcpinfo(1)`, `dhcpagent(1M)`, `ifconfig(1M)`, `iostat(1M)`, `kstat(1M)`, `mibiisa(1M)`, `savecore(1M)`, `vmstat(1M)`, `zonecfg(1M)`, `hosts(4)`, `inet_type(4)`, `networks(4)`, `protocols(4)`, `services(4)`, `attributes(5)`, `dhcp(5)`, `kstat(7D)`, `inet(7P)`, `inet6(7P)`

Droms, R., *RFC 2131, Dynamic Host Configuration Protocol*, Network Working Group, March 1997.

Droms, R. *RFC 3315, Dynamic Host Configuration Protocol for IPv6 (DHCPv6)*. Cisco Systems. July 2003.

Notes When displaying interface information, `netstat` honors the `DEFAULT_IP` setting in `/etc/default/inet_type`. If it is set to `IP_VERSION4`, then `netstat` will omit information relating to IPv6 interfaces, statistics, connections, routes and the like.

However, you can override the `DEFAULT_IP` setting in `/etc/default/inet_type` on the command-line. For example, if you have used the command-line to explicitly request IPv6 information by using the `inet6` address family or one of the IPv6 protocols, it will override the `DEFAULT_IP` setting.

If you need to examine network status information following a kernel crash, use the `mdb(1)` utility on the `savecore(1M)` output.

The `netstat` utility obtains TCP statistics from the system by opening `/dev/tcp` and issuing queries. Because of this, `netstat` might display an extra, unused connection in `IDLE` state when reporting connection status.

Previous versions of `netstat` had undocumented methods for reporting kernel statistics published using the `kstat(7D)` facility. This functionality has been removed. Use `kstat(1M)` instead.

`netstat` restricts its output to information that is relevant to the zone in which `netstat` runs. (This is true for both shared-IP and exclusive-IP zones.)

Name netstrategy – return network configuration information

Synopsis

```
/usr/sbin/netstrategy
```

Description The `netstrategy` command determines the network configuration strategy in use on a system and returns information in a form that is easily consumable by a script. The command returns three tokens:

<root filesystem type> <primary interface> <network config strategy>

These tokens are described as follows:

<root filesystem type>
Type of filesystem that contains the bootable kernel, as would be specified in the *fstype* column of the `mnttab(4)`.

<primary interface>
Name of the primary network interface. For a diskless machine, this is the interface used to load the kernel.

<network config strategy>
The means by which a system obtains its IP address for booting. This can be one of `rarp`, `dhcp`, or `none`.

The `netstrategy` command is not intended for use on a command line.

Options The `netstrategy` command has no options.

Exit Status

`0` Success.

`!=0` An error occurred.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os
Interface Stability	Committed

See Also `ifconfig(1M)`, `mnttab(4)`, `attributes(5)`

Name newaliases – rebuild the data base for the mail aliases file

Synopsis

```
newaliases
```

Description `newaliases` rebuilds the random access data base for the mail aliases file `/etc/mail/aliases`.

`newaliases` accepts all the flags that `sendmail(1M)` accepts. However, most of these flags have no effect, except for the `-C` option and three of the `Processing Options` that can be set from a configuration file with the `-o` option:

`-C` */path/to/alt/config/file*	Use alternate configuration file.
`-oA`*file*	Specify possible alias files.
`-oL`*n*	Set the default log level to *n*. Defaults to `9`.
`-on`	Validate the RHS of aliases when rebuilding the `aliases(4)` database.

`newaliases` runs in verbose mode (`-v` option) automatically.

Examples **EXAMPLE 1** Running the `newaliases` Command

The following command runs `newaliases` on an alias file different from the `/etc/mail/aliases` default in `sendmail(1M)`:

```
example% newaliases -oA/path/to/alternate/alias/file
```

Exit Status `newaliases` returns an exit status describing what it did. The codes are defined in `/usr/include/sysexits.h`.

`EX_OK`	Successful completion on all addresses.
`EX_NOUSER`	User name not recognized.
`EX_UNAVAILABLE`	Catchall. Necessary resources were not available.
`EX_SYNTAX`	Syntax error in address.
`EX_SOFTWARE`	Internal software error, including bad arguments.
`EX_OSERR`	Temporary operating system error, such as “cannot fork”.
`EX_NOHOST`	Host name not recognized.
`EX_TEMPFAIL`	Message could not be sent immediately, but was queued.

Files

`/etc/aliases`	Symbolic link to `/etc/mail/aliases`
`/etc/mail/aliases.pag`	
`/etc/mail/aliases.dir`	`ndbm` files maintained by `newaliases`
`/etc/mail/aliases.db`	Berkeley DataBase file maintained by newaliases

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	service/network/smtp/sendmail

See Also sendmail(1M), aliases(4), attributes(5)

Name newfs – construct a UFS file system

Synopsis `newfs [-NSBTv] [`*mkfs-options*`] `*raw-device*

Description `newfs` is a friendly front-end to the `mkfs(1M)` program for making UFS file systems on disk partitions. `newfs` calculates the appropriate parameters to use and calls `mkfs`.

If run interactively (that is, standard input is a tty), `newfs` prompts for confirmation before making the file system.

If the `-N` option is not specified and the inodes of the device are not randomized, `newfs` calls `fsirand(1M)`.

You must be super-user or have appropriate write privileges to use this command.

Creating a Multiterabyte UFS File System

Keep the following limitations in mind when creating a multiterabyte UFS file system:

- *nbpi* is set to 1 Mbyte unless you specifically set it higher. You cannot set *nbpi* lower than 1 Mbyte on a multiterabyte UFS file system.
- *fragsize* is set equal to *bsize*.

Options The following options are supported:

`-N`
Print out the file system parameters that would be used to create the file system without actually creating the file system. `fsirand(1M)` is not called here.

`-S`
Sends to stdout a human-readable version of the superblock that would be used to create a filesystem with the specified configuration parameters.

`-B`
Sends to stdout a binary (machine-readable) version of the superblock that would be used to create a filesystem with the specified configuration parameters.

`-T`
Set the parameters of the file system to allow eventual growth to over a terabyte in total file system size. This option sets *fragsize* to be the same as *bsize*, and sets *nbpi* to 1 Mbyte, unless the `-i` option is used to make it even larger. If you use the `-f` or `-i` options to specify a *fragsize* or *nbpi* that is incompatible with this option, the user-supplied value of *fragsize* or *nbpi* is ignored.

`-v`
Verbose. `newfs` prints out its actions, including the parameters passed to `mkfs`.

mkfs-options
Options that override the default parameters are:

`-a` *apc*
The number of alternate sectors per cylinder to reserve for bad block replacement for SCSI devices only. The default is `0`.

This option is not applicable for disks with EFI labels and is ignored.

`-b` *bsize*
The logical block size of the file system in bytes, either 4096 or 8192. The default is `8192`. The sun4u architecture does not support the `4096` block size.

`-c` *cgsize*
The number of cylinders per cylinder group, ranging from `16` to `256`. The default is calculated by dividing the number of sectors in the file system by the number of sectors in a gigabyte. Then, the result is multiplied by `32`. The default value is always between `16` and `256`.

`mkfs` can override this value. See `mkfs_ufs(1M)` for details.

This option is not applicable for disks with EFI labels and is ignored.

`-C` *maxcontig*
The maximum number of logical blocks, belonging to one file, that are allocated contiguously. The default is calculated as follows:

`maxcontig =` *disk drive maximum transfer size / disk block size*

If the disk drive's maximum transfer size cannot be determined, the default value for `maxcontig` is calculated from kernel parameters as follows:

If `maxphys` is less than `ufs_maxmaxphys`, which is typically 1 Mbyte, then `maxcontig` is set to `maxphys`. Otherwise, `maxcontig` is set to `ufs_maxmaxphys`.

You can set `maxcontig` to any positive integer value.

The actual value will be the lesser of what has been specified and what the hardware supports.

You can subsequently change this parameter by using `tunefs(1M)`.

`-d` *gap*
Rotational delay. This option is obsolete in the Solaris 10 release. The value is always set to `0`, regardless of the input value.

`-f` *fragsize*
The smallest amount of disk space in bytes that can be allocated to a file. *fragsize* must be a power of 2 divisor of *bsize*, where:

bsize / *fragsize* is 1, 2, 4, or 8.

This means that if the logical block size is `4096`, legal values for *fragsize* are `512`, `1024`, `2048`, and `4096`. When the logical block size is `8192`, legal values are `1024`, `2048`, `4096`, and `8192`. The default value is `1024`.

For file systems greater than 1 terabyte or for file systems created with the `-T` option, *fragsize* is forced to match block size (*bsize*).

`-i` *nbpi*
The number of bytes per inode, which specifies the density of inodes in the file system. The number is divided into the total size of the file system to determine the number of inodes to create.

This value should reflect the expected average size of files in the file system. If fewer inodes are desired, a larger number should be used. To create more inodes, a smaller number should be given. The default for *nbpi* is as follows:

```
Disk size                Density

Less than 1GB            2048
Less than 2GB            4096
Less than 3GB            6144
3GB to 1 Tbyte           8192
Greater than 1 Tbyte
   or created with -T    1048576
```

The number of inodes can increase if the file system is expanded with the `growfs` command.

`-m` *free*
The minimum percentage of free space to maintain in the file system, between 0% and 99%, inclusively. This space is off-limits to users. Once the file system is filled to this threshold, only the super-user can continue writing to the file system.

The default is ((64 Mbytes/partition size) * 100), rounded down to the nearest integer and limited between 1% and 10%, inclusively.

This parameter can be subsequently changed using the `tunefs`(1M) command.

`-n` *nrpos*
The number of different rotational positions in which to divide a cylinder group. The default is `8`.

This option is not applicable for disks with EFI labels and is ignored.

`-o` *space* | *time*
The file system can either be instructed to try to minimize the *time* spent allocating blocks, or to try to minimize the *space* fragmentation on the disk. The default is *time*.

This parameter can subsequently be changed with the `tunefs`(1M) command.

`-r` *rpm*
The rotational speed of the disk in revolutions per minute. The default is driver- or device-specific.

Note that you specify *rpm* for `newfs` and *rps* for `mkfs`.

This option is not applicable for disks with EFI labels and is ignored.

`-s` *size*
The size of the file system in sectors. The default is to use the entire partition.

`-t` *ntrack*
The number of tracks per cylinder on the disk. The default is taken from the disk label.

This option is not applicable for disks with EFI labels and is ignored.

Operands The following operands are supported:

raw-device
The name of a raw special device residing in the `/dev` directory (for example, `/dev/rdsk/c0t0d0s6`) on which to create the file system.

Usage See `largefile(5)` for the description of the behavior of `newfs` when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

Examples **EXAMPLE 1** Displaying the Parameters for the Raw Special Device

The following example verbosely displays the parameters for the raw special device, `c0t0d0s6`. It does not actually create a new file system:

```
example# newfs -Nv /dev/rdsk/c0t0d0s6
mkfs -F ufs -o N /dev/rdsk/c0t0d0s6 1112940 54 15 8192 1024 16 10 60
2048 t 0 –1 8 /dev/rdsk/c0t0d0s6: 1112940 sectors in
1374 cylinders of 15 tracks, 54 sectors 569.8MB in 86 cyl
groups (16 c/g, 6.64MB/g, 3072 i/g) super-block backups
(for fsck -b #) at:
32, 13056, 26080, 39104, 52128, 65152, 78176, 91200, 104224, . . .
```

EXAMPLE 2 Creating a UFS File System That Will Eventually Be Grown to a Multiterabyte UFS File System

The following example creates a UFS file system that will eventually be grown to a multiterabyte UFS file system.

This command creates a 800-Gbyte file system on the volume, `/dev/md/rdsk/d99`.

```
# newfs -T /dev/md/rdsk/d99
newfs: construct a new file system /dev/md/rdsk/d99: (y/n)? y
   /dev/md/rdsk/d99: 1677754368 sectors in 45512 cylinders of
   144 tracks, 256 sectors
   819216.0MB in 1821 cyl groups (25 c/g, 450.00MB/g, 448 i/g) . . .
```

Then, if you increase the volume size for this file system, you can use the `growfs` command to expand the file system. The file system is grown to 1.2 terabytes in this example:

```
# growfs -v /dev/md/rdsk/d99
/usr/lib/fs/ufs/mkfs -G /dev/md/rdsk/d99 2516631552 /dev/md/rdsk/d99:
   2516631552 sectors in 68268 cylinders of 144 tracks, 256  sectors
   1228824.0MB in 2731 cyl groups (25 c/g, 450.00MB/g, 448 i/g). . .
```

Exit Status The following exit values are returned:

`0`
The operation was successful.

`1`, `10`
Usage error or internal error. A message is output to `STDERR` explaining the error.

Other exit values may be returned by `mkfs(1M)`, which is called by `newfs`.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also `fsck(1M)`, `fsck_ufs(1M)`, `fsirand(1M)`, `mkfs(1M)`, `mkfs_ufs(1M)`, `tunefs(1M)`, `attributes(5)`, `largefile(5)`, `ufs(7FS)`

Diagnostics `newfs: No such file or directory`
The device specified does not exist, or a disk partition was not specified.

special`: cannot open`
You must write access to the device to use this command.

Name newkey – create a new Diffie-Hellman key pair in the publickey database

Synopsis

```
newkey -h hostname [-s nis | files | ldap]

newkey -u username [-s nis | files | ldap]
```

Description `newkey` establishes new public keys for users and machines on the network. These keys are needed when using secure RPC or secure NFS service.

`newkey` prompts for a password for the given *username* or `hostname` and then creates a new public/secret Diffie-Hellman 192 bit key pair for the user or host. The secret key is encrypted with the given password. The key pair can be stored in the `/etc/publickey` file or the NIS `publickey` map.

`newkey` consults the `publickey` entry in the name service switch configuration file (see `nsswitch.conf(4)`) to determine which naming service is used to store the secure RPC keys. If the `publickey` entry specifies a unique name service, `newkey` will add the key in the specified name service. However, if there are multiple name services listed, `newkey` cannot decide which source to update and will display an error message. The user is required to specify the source explicitly with the `-s` option.

In the case of NIS, `newkey` should be run by the superuser on the master NIS server for that domain.

In the case of LDAP, `newkey` should be run by the superuser on a machine that also recognizes the directory manager's bind distinguished name (DN) and password to perform an LDAP update for the host.

Options

`-h` *hostname* — Create a new public/secret key pair for the privileged user at the given `hostname`. Prompts for a password for the given `hostname`.

`-u` *username* — Create a new public/secret key pair for the given *username*. Prompts for a password for the given *username*.

`-s nis`
`-s files`
`-s ldap` — Update the database in the specified source: `nis` (for NIS), `files`, or `ldap` (LDAP). Other sources may be available in the future.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also `chkey(1)`, `keylogin(1)`, `nsswitch.conf(4)`, `publickey(4)`, `attributes(5)`

Name nfs4cbd – NFS Version 4 callback daemon

Synopsis

```
/usr/lib/nfs/nfs4cbd
```

Description The `nfs4cbd` daemon manages communication endpoints for the NFS Version 4 protocol callback program. `nfs4cbd` runs on the NFS Version 4 client and creates a listener port for each transport over which callbacks can be sent.

The `nfs4cbd` daemon is provided for the exclusive use of the NFS version 4 client.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/file-system/nfs

See Also `svcs(1)`, `mount_nfs(1M)`, `svcadm(1M)`, `attributes(5)`, `smf(5)`

Notes The `nfs4cbd` service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/network/nfs/cbd
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. The service's status can be queried using the `svcs(1)` command.

If it is disabled, it will be enabled by `mount_nfs(1M)` and `automountd(1M)` on the first NFSv4 mount, unless its `application/auto_enable` property is set to `false`.

This daemon might not exist in a future release of Solaris.

Name nfsd – NFS daemon

Synopsis

```
/usr/lib/nfs/nfsd [-a] [-c #_conn] [-l listen_backlog]
     [-p protocol] [-t device] [nservers]
```

Description `nfsd` is the daemon that handles client file system requests. Only users with `{PRIV_SYS_NFS}` and sufficient privileges to write to `/var/run` can run this daemon.

The `nfsd` daemon is automatically invoked using `share(1M)` with the `-a` option.

By default, `nfsd` starts over the TCP and UDP transports for versions 2 and 3. By default, it starts over the TCP for version 4. You can change this with the `-p` option.

A previously invoked `nfsd` daemon started with or without options must be stopped before invoking another `nfsd` command.

To change startup parameters for `nfsd`, use the `sharectl(1M)` command.

SMF Management The `nfsd` service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/network/nfs/server
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. The service's status can be queried using the `svcs(1)` command.

If `nfsd` is killed with SIGTERM, it will not be restarted by the service management facility. Instead, `nfsd` can be restarted by other signals, such as SIGINT.

The `sharectl(1M)` command can be used to manage all the parameters related to `nfsd`. The following are the parameters currently supported:

`server_versmax=`*num*
`server_versmin=`*num*
: The NFS server only uses NFS versions in the range specified by these variables. Valid values or versions are: `2`, `3`, and `4`. If one or both of these parameters are left unset, the default minimum version is `2`, while the default maximum version is `4`.

`server_delegation=on|off`
: By default, this variable is `on` and the NFS server provides delegations to clients. The user can turn off delegations for all exported file systems by setting this variable to `off` (case-sensitive). This variable applies only to NFS Version 4.

`max_connections=`*num*
: Sets the maximum number of concurrent, connection-oriented connections. The default is unlimited and is obtained by setting to `-1`. Equivalent to the `-c` option in `nfsd`.

`listen_backlog=`*num*
: Set connection queue length for the NFS over a connection-oriented transport. The default value is `32`, meaning 32 entries in the queue. Equivalent to the `-l` option in `nfsd`.

`protocol=ALL`
Start `nfsd` over the specified protocol only. Equivalent to the -p option in `nfsd`. `ALL` is equivalent to `-a` on the `nfsd` command line. Mutually exclusive of NFS SMF parameter `device`. One or the other of NFS SMF parameters `device` and `protocol` must not be set. If both are set, the `nfs/server` service goes into maintenance mode. For the UDP protocol, only version 2 and version 3 service is established. NFS Version 4 is not supported for the UDP protocol. Equivalent to the `-p` option.

`device=`*devname*
Start NFS daemon for the transport specified by the specified device only. Equivalent to the `-t` option in `nfsd`. Mutually exclusive of NFS SMF parameter `protocol`. One or the other of NFS SMF parameters `device` and `protocol` must not be set.

`servers=`*num*
Maximum number of concurrent NFS requests. The default is `1024`. Equivalent to the *nservers* operand.

See `EXAMPLES`, below.

Options The following options are supported:

`-a` Start an NFS daemon over all available connectionless and connection-oriented transports, including UDP and TCP. Equivalent of setting the `protocol` parameter to `ALL` in the SMF for NFS using the `sharectl(1M)` command.

`-c` *#_conn* This sets the maximum number of connections allowed to the NFS server over connection-oriented transports. By default, the number of connections is unlimited. Equivalent of the `max_connections` parameter in the SMF for NFS using the `sharectl(1M)` command.

`-l` Set connection queue length for the NFS TCP over a connection-oriented transport. The default value is 32 entries. Equivalent of the `listen_backlog` parameter in the SMF for NFS using the `sharectl(1M)` command.

`-p` *protocol* Start a NFS daemon over the specified protocol. Equivalent of the `protocol` parameter in the SMF for NFS using the `sharectl(1M)` command.

`-t` *device* Start a NFS daemon for the transport specified by the given device. Equivalent of the `device` parameter in the SMF for NFS using the `sharectl(1M)` command.

Operands The following operands are supported:

nservers This sets the maximum number of concurrent NFS requests that the server can handle. This concurrency is achieved by up to *nservers* threads created as needed in the kernel. *nservers* should be based on the load expected on this server. 16 is the usual number of *nservers*. If *nservers* is not specified, the maximum number of concurrent NFS requests will default to 1. Equivalent of the `servers` parameter in

the SMF for NFS using the sharectl(1M) command.

Usage If the nfs_portmon variable is set in /etc/system, then clients are required to use privileged ports (ports < IPPORT_RESERVED) to get NFS services. This variable is equal to zero by default. This variable has been moved from the "nfs" module to the "nfssrv" module. To set the variable, edit the /etc/system file and add this entry:

```
set nfssrv:nfs_portmon = 1
```

Examples **EXAMPLE 1** Turning Off Delegation

The nfsd properties specified in these examples are described under "SMF Management," above.

Delegation is an NFSv4 feature in which the server delegates the management of a file to a client. For example, the server could grant (or not grant) either a read delegation or a write delegation to a client. The following command does this, setting the server_delegation property to off.

```
# sharectl set -p server_delegation=off nfs
```

EXAMPLE 2 Determining Value of Delegation

The following command obtains the current value of the server_delegation property.

```
# sharectl get -p server_delegation nfs
server_delegation=on
```

EXAMPLE 3 Setting Maximum Number of Concurrent Requests

The following command sets the maximum number of concurrent NFS requests.

```
# sharectl set -p servers=32 nfs
```

EXAMPLE 4 Setting Connection Queue Length

The following command sets the maximum queue length for the NFS over a connection-oriented transport.

```
# sharectl set -p listen_backlog=48 nfs
```

Exit Status

0 Daemon started successfully.

1 Daemon failed to start.

Files

.nfs*XXX*	Client machine pointer to an open-but-unlinked file.
/etc/system	System configuration information file.
/var/nfs/v4_state /var/nfs/v4_oldstate	Directories used by the server to manage client state information. These directories should not be removed.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	service/file-system/nfs

See Also ps(1), svcs(1), mountd(1M), share(1M), sharectl(1M), svcadm(1M), sharetab(4), system(4), attributes(5), smf(5)

Oracle Solaris Administration: Network Services

Notes Manually starting and restarting `nfsd` is not recommended. If it is necessary to do so, use `svcadm` to enable or disable the `nfs` service (`svc:/network/nfs/server`). If it is disabled, it will be enabled by share_nfs(1M), unless its `application/auto_enable` property is set to `false`. See the *Oracle Solaris Administration: Network Services*, and svcadm(1M) for more information.

Name nfslogd – nfs logging daemon

Synopsis `/usr/lib/nfs/nfslogd`

Description The `nfslogd` daemon provides operational logging to the Solaris NFS server. It is the `nfslogd` daemon's job to generate the activity log by analyzing the RPC operations processed by the NFS server. The log will only be generated for file systems exported with logging enabled. This is specified at file system export time by means of the `share_nfs(1M)` command.

NFS server logging is not supported on Solaris machines that are using NFS Version 4.

Each record in the log file includes a time stamp, the IP address (or hostname if it can be resolved) of the client system, the file or directory name the operation was performed on, and the type of operation. In the basic format, the operation can either be an input (i) or output (o) operation. The basic format of the NFS server log is compatible with the log format generated by the Washington University FTPd daemon. The log format can be extended to include directory modification operations, such as `mkdir`, `rmdir`, and `remove`. The extended format is not compatible with the Washington University FTPd daemon format. See `nfslog.conf(4)` for details.

The NFS server logging mechanism is divided in two phases. The first phase is performed by the NFS kernel module, which records raw RPC requests and their results in work buffers backed by permanent storage. The location of the work buffers is specified in the `/etc/nfs/nfslog.conf` file. Refer to `nfslog.conf(4)` for more information. The second phase involves the `nfslogd` user-level daemon, which periodically reads the work buffers, interprets the raw RPC information, groups related RPC operations into single transaction records, and generates the output log. The `nfslogd` daemon then sleeps waiting for more information to be logged to the work buffers. The amount of time that the daemon sleeps can be configured by modifying the `IDLE_TIME` parameter in `/etc/default/nfslogd`. The work buffers are intended for internal consumption of the `nfslogd` daemon.

NFS operations use file handles as arguments instead of path names. For this reason the `nfslogd` daemon needs to maintain a database of file handle to path mappings in order to log the path name associated with an operation instead of the corresponding file handle. A file handle entry is added to the database when a client performs a lookup or other NFS operation that returns a file handle to the client.

Once an NFS client obtains a file handle from a server, it can hold on to it for an indefinite time, and later use it as an argument for an NFS operation on the file or directory. The NFS client can use the file handle even after the server reboots. Because the database needs to survive server reboots, it is backed by permanent storage. The location of the database is specified by the *fhtable* parameter in the `/etc/nfs/nfslog.conf` file. This database is intended for the internal use of the `nfslogd` daemon.

In order to keep the size of the file handle mapping database manageable, `nfslogd` prunes the database periodically. It removes file handle entries that have not been accessed in more than a specified amount of time. The `PRUNE_TIMEOUT` configurable parameter in

/etc/default/nfslogd specifies the interval length between successive runs of the pruning process. A file handle record will be removed if it has not been used since the last time the pruning process was executed. Pruning of the database can effectively be disabled by setting the PRUNE_TIMEOUT as high as INT_MAX.

When pruning is enabled, there is always a risk that a client may have held on to a file handle longer than the PRUNE_TIMEOUT and perform an NFS operation on the file handle after the matching record in the mapping database had been removed. In such case, the pathname for the file handle will not be resolved, and the log will include the file handle instead of the pathname.

There are various configurable parameters that affect the behavior of the nfslogd daemon. These parameters are found in /etc/default/nfslogd and are described below:

UMASK	Sets the file mode for the log files, work buffer files and file handle mapping database.
MIN_PROCESSING_SIZE	Specifies the minimum size, in bytes, that the buffer file must reach before processing the work information and writing to the log file. The value of MIN_PROCESSING_SIZE must be between 1 and ulimit.
IDLE_TIME	Specifies the amount of time, in seconds, the daemon should sleep while waiting for more information to be placed in the buffer file. IDLE_TIME also determines how often the configuration file will be reread. The value of IDLE_TIME must be between 1 and INT_MAX.
MAX_LOGS_PRESERVE	The nfslogd periodically cycles its logs. MAX_LOGS_PRESERVE specifies the maximum number of log files to save. When MAX_LOGS_PRESERVE is reached, the oldest files will be overwritten as new log files are created. These files will be saved with a numbered extension, beginning with filename.0. The oldest file will have the highest numbered extension up to the value configured for MAX_LOGS_PRESERVE. The value of MAX_LOGS_PRESERVE must be between 1 and INT_MAX.
CYCLE_FREQUENCY	Specifies how often, in hours, the log files are cycled. CYCLE_FREQUENCY is used to insure that the log files do not get too large. The value of CYCLE_FREQUENCY must be between 1 and INT_MAX.
MAPPING_UPDATE_INTERVAL	Specifies the time interval, in seconds, between updates of the records in the file handle to path mapping tables. Instead of updating the atime of a record each time that record is accessed, it is only updated if it has aged based on this

parameter. The record access time is used by the pruning routine to determine whether the record should be removed from the database. The value of this parameter must be between 1 and `INT_MAX`.

`PRUNE_TIMEOUT` Specifies when a database record times out, in hours. If the time that elapsed since the record was last accessed is greater than `PRUNE_TIMEOUT` then the record can be pruned from the database. The default value for `PRUNE_TIMEOUT` is 168 hours (7 days). The value of `PRUNE_TIMEOUT` must be between 1 and `INT_MAX`.

Exit Status The following exit values are returned:

`0` Daemon started successfully.

`1` Daemon failed to start.

Files `/etc/nfs/nfslogtab`

`/etc/nfs/nfslog.conf`

`/etc/default/nfslogd`

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	service/file-system/nfs

See Also `share_nfs(1M)`, `nfslog.conf(4)`, `attributes(5)`

Name nfsmapid – NFS user and group id mapping daemon

Synopsis `/usr/lib/nfs/nfsmapid`

Description The `nfsmapid` daemon maps to and from NFS version 4 `owner` and `owner_group` identification attributes and local UID and GID numbers used by both the NFS version 4 client and server.

`nfsmapid` uses the `passwd` and `group` entries in the `/etc/nsswitch.conf` file to direct how it performs the mappings.

The `nfsmapid` daemon has no external, customer-accessible interfaces. You can, however, administratively configure `nfsmapid` in one of the following ways:

- Specify the `nfsmapid_domain` parameter in the SMF for NFS using the `sharectl(1M)` command.
- Specify the `_nfsv4idmapdomain` DNS resource record.

The currently selected NFSv4 domain is available in the file `/var/run/nfs4_domain`.

Please refer to the *Oracle Solaris Administration: Network ServicesOracle Solaris Administration: Network Services* for further details.

SMF Management The `nfsmapid` service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/network/nfs/mapid
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. The service's status can be queried using the `svcs(1)` command.

If it is disabled, it will be enabled by `mount_nfs(1M)`, `share_nfs(1M)`, and `automountd(1M)`, unless its `application/auto_enable` property is set to `false`.

`nfsmapid` caches a user's UID and GID. If a user subsequently changes a UID or GID, using one of the utilities listed below, the `nfsmapid` cache becomes stale. At this point, any NFS operation that gets or set attributes will result in the exchange of this stale information. To resolve this situation, restart `nfsmapid`, as follows:

```
# svcadm restart svc:/network/nfs/mapid:default
```

The startup SMF parameter designating a domain name (`nfsmapid_domain`) can be manipulated with the `sharectl(1M)` command.

`nfsmapid_domain`
: The setting for the NFS SMF parameter `nfsmapid_domain` overrides the domain used by `nfsmapid` for building and comparing outbound and inbound attribute strings, respectively. Also, this setting overrides any other mechanism for setting the NFSv4 domain. In the absence of a `nfsmapid_domain` setting, the `nfsmapid` daemon determines the NFSv4 domain as follows:

- If a properly configured `/etc/resolv.conf` (see `resolv.conf(4)`) exists, `nfsmapid` queries specified nameserver(s) for the domain.
- If a properly configured `/etc/resolv.conf` (see `resolv.conf(4)`) exists, but the queried name server does not have a proper record of the domain name, `nfsmapid` attempts to obtain the domain name through the `BIND` interface (see `resolver(3RESOLV)`).
- If no `/etc/resolv.conf` exists, `nfsmapid` falls back on using the configured domain name (see `domainname(1M)`), which is returned with the leading domain suffix removed. For example, for `widgets.sales.acme.com`, `sales.acme.com` is returned.
- If `/etc/resolv.conf` does not exist, no domain name has been configured (or no `/etc/defaultdomain` exists), `nfsmapid` falls back on obtaining the domain name from the host name, if the host name contains a fully qualified domain name (FQDN).

If a domain name is still not obtained following all of the preceding steps, `nfsmapid` will have no domain configured. This results in the following behavior:

- Outbound `owner` and `owner_group` attribute strings are encoded as literal ID's. For example, the UID 12345 is encoded as `12345`.
- `nfsmapid` ignores the `domain` portion of the inbound attribute string and performs name service lookups only for the user or group. If the user/group exists in the local system name service databases, then the proper UID/GID will be mapped even when no domain has been configured.

This behavior implies that the same administrative user/group domain exists between NFSv4 client and server (that is, the same UID/GIDs for users/groups on both client and server). In the case of overlapping ID spaces, the inbound attribute string could potentially be mapped to the wrong id. However, this is not functionally different from mapping the inbound string to `nobody`, yet provides greater flexibility. See `EXAMPLES`, below.

The utilities that allow you to change UID and GID are:

- `usermod(1M)`
- `userdel(1M)`
- `groupmod(1M)`
- `groupdel(1M)`

Files `/var/run/nfs4_domain` Contains the domain name currently used by NFSv4.

Examples **EXAMPLE 1** Setting Domain Name

The following command uses `sharectl` to set the domain name.

```
# sharectl set -p nfsmapid_domain=oracle.com nfs
```

The `nfsmapid_domain` property is described under `NOTES`, below.

EXAMPLE 2 Obtaining Domain Name

The following command uses `sharectl` to obtain the current domain name.

```
# sharectl get -p nfsmapid_domain nfs nfsmapid_domain=oracle.com
```

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/file-system/nfs

See Also

`svcs(1)`, `automountd(1M)`, `domainname(1M)`, `groupdel(1M)`, `groupmod(1M)`, `mount_nfs(1M)`, `svcadm(1M)`, `share_nfs(1M)`, `sharectl(1M)`, `userdel(1M)`, `usermod(1M)`, `resolver(3RESOLV)`, `nfs(4)`, `resolv.conf(4)`, `attributes(5)`, `smf(5)`

Oracle Solaris Administration: Network Services

Notes

The `nfsmapid` daemon might not exist in a future release of Solaris.

Name nfsref – NFS referral utility

Synopsis

```
nfsref [-t svc_type] add path location [location ...]

nfsref [-t svc_type] remove path

nfsref [-t svc_type] lookup path
```

Description The `nfsref` command manages NFS Version 4 referrals, which are server-side pointers used to redirect clients to actual locations of file systems. Referrals are based on reparse points (see `reparsed(1M)` and `libreparse(3LIB)`). The path arguments in all forms of the synopsis refer to the path to the reparse point symbolic link. This command currently implements implements two service types: `nfs-basic`, which is the default, and `nfs-fedfs`. `nfs-basic` referrals embed location information within the reparse point, while `nfs-fedfs` referrals embed information to look up location information in LDAP (see `fedfs(5)`).

The first form of the command, `nfsref add`, creates a referral pointing to the specified locations. If a reparse point does not exist, one is created. If it does exist, NFS service data is added or replaces existing NFS service data. Each location has a *host*:*/path* format. The path can contain spaces, which must be escaped to ensure proper shell parsing.

The second form of the command, `nfsref delete`, removes an NFS referral. It removes NFS service data from the specified reparse point, and removes the reparse point if there are no other types of service data present.

The third form of the command, `nfsref lookup`, displays the locations to which the specified NFS referral points.

Creating FedFS referrals must always be done with LDAP simple authentication with the root distinguished name and matching password available. The root DN must be stored for the LDAP server in `nsdbparams`; the password may be stored in `nsdbparams`, or the admin may enter it when prompted. The default LDAP host and port will be those set with `nsdbparams` get unless overridden with the `FEDFS_NSDB_HOST` and `FEDFS_NSDB_PORT` environment variables.

Options The following options are supported:

`-t` *svc_type*
: Specify a service type. Currently, only `nfs-basic` and `nfs-fedfs` are supported.

Examples **EXAMPLE 1** Adding an NFS Referral

The following command creates an NFS referral at the server path `/pool/home/bob`, pointing to the resource `homeserver:/homepool/bob`.

```
# nfsref add /pool/home/bob homeserver:/homepool/bob
Created reparse point /pool/home/bob

# nfsref lookup /pool/home/bob
homeserver:/homepool/bob
```

EXAMPLE 2 Removing an NFS Referral

The following command removes an NFS referral at the server path `/pool/home/bob`.

```
# nfsref remove /pool/home/bob homeserver:/homepool/bob
Removed svc_type 'nfs-basic' from /pool/home/bob
```

Exit Status

`0`
Successful completion.

`>0`
An error occurred.

Files

`/usr/lib/reparsed/*.so.1`
Per-service plugins for `reparsed`.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	service/file-system/nfs
Interface Stability	Consolidation Private

See Also

`reparsed(1M)`, `libreparse(3LIB)`, `attributes(5)`, `fedfs(5)`

Name nfsstat – NFS statistics

Synopsis

```
nfsstat [-cnrsza] [-T u | d ] [-v version] [interval [count]]

nfsstat -m [pathname]...
```

Description `nfsstat` displays statistical information about the NFS and RPC (Remote Procedure Call), interfaces to the kernel. It can also be used to reinitialize this information. If no options are given the default is as follows:

```
nfsstat -csnra
```

The default displays everything, but reinitializes nothing.

Options

`-a`
Display `NFS_ACL` information.

`-c`
Display client information. Only the client side NFS, RPC, and NFS_ACL information is printed. Can be combined with the `-n`, `-r`, and `-a` options to print client side NFS, RPC, and NFS_ACL information only.

`-m` [*pathname...*]
Display statistics for each NFS mounted file system. If *pathname* is not specified, displays statistics for all NFS mounted file systems. If *pathname* is specified, displays statistics for the NFS mounted file systems indicated by *pathname*.

This includes the server name and address, mount flags, current read and write sizes, the retransmission count, the attribute cache timeout values, failover information, and the timers used for dynamic retransmission. The dynamic retransmission timers are displayed only where dynamic retransmission is in use. By default, NFS mounts over the TCP protocols and NFS Version 3 mounts over either TCP or UDP do not use dynamic retransmission.

If you specify the `-m` option, this is the only option that `nfsstat` uses. If you specify other options with `-m`, you receive an error message alerting that the `-m` flag cannot be combined with other options.

`-n`
Display NFS information. NFS information for both the client and server side are printed. Can be combined with the `-c` and `-s` options to print client or server NFS information only.

`-r`
Display RPC information.

`-s`
Display server information.

`-T u | d`
Display a time stamp.

Specify `u` for a printed representation of the internal representation of time. See `time(2)`. Specify `d` for standard date format. See `date(1)`.

`-v` *version*
Specify which NFS version for which to print statistics. When followed by the optional *version* argument, (`2|3|4`), specifies statistics for that version. By default, prints statistics for all versions.

`-z`
Zero (reinitialize) statistics. This option is for use by the super user only, and can be combined with any of the above options to zero particular sets of statistics after printing them.

Operands The following operands are supported:

count
Display only count reports

interval
Report once each interval seconds.

pathname
Specify the pathname of a file in an NFS mounted file system for which statistics are to be displayed.

Displays The server RPC display includes the following fields:

`badcalls`
The total number of calls rejected by the RPC layer (the sum of `badlen` and `xdrcall` as defined below).

`badlen`
The number of RPC calls with a length shorter than a minimum-sized RPC call.

`calls`
The total number of RPC calls received.

`dupchecks`
The number of RPC calls that looked up in the duplicate request cache.

`dupreqs`
The number of RPC calls that were found to be duplicates.

`nullrecv`
The number of times an RPC call was not available when it was thought to be received.

`xdrcall`
The number of RPC calls whose header could not be XDR decoded.

The server NFS display shows the number of NFS calls received (`calls`) and rejected (`badcalls`), and the counts and percentages for the various calls that were made.

The server NFS_ACL display shows the counts and percentages for the various calls that were made.

The client RPC display includes the following fields:

`calls`
The total number of RPC calls made.

`badcalls`
The total number of calls rejected by the RPC layer.

`badverfs`
The number of times the call failed due to a bad verifier in the response.

`badxids`
The number of times a reply from a server was received which did not correspond to any outstanding call.

`cantconn`
The number of times the call failed due to a failure to make a connection to the server.

`cantsend`
The number of times a client was unable to send an RPC request over a connectionless transport when it tried to do so.

`interrupts`
The number of times the call was interrupted by a signal before completing.

`newcreds`
The number of times authentication information had to be refreshed.

`nomem`
The number of times the call failed due to a failure to allocate memory.

`retrans`
The number of times a call had to be retransmitted due to a timeout while waiting for a reply from the server. Applicable only to RPC over connection-less transports.

`timeouts`
The number of times a call timed out while waiting for a reply from the server.

`timers`
The number of times the calculated time-out value was greater than or equal to the minimum specified time-out value for a call.

The client NFS display shows the number of calls sent and rejected, as well as the number of times a `CLIENT` handle was received (`clgets`), the number of times the `CLIENT` handle cache had no unused entries (`cltoomany`), as well as a count of the various calls and their respective percentages.

The client NFS_ACL display shows the counts and percentages for the various calls that were made.

The `-m` option includes information about mount flags set by mount options, mount flags internal to the system, and other mount information. See `mount_nfs(1M)`.

The following mount flags are set by mount options:

`forcedirectio`
Data transferred directly between client and server, with no buffering on client.

`grpid`
BSD group id inheritance. See description in `mount_nfs(1M)`.

`hard`
Hard mount.

`intr`
Interrupts allowed on hard mount.

`llock`
Local locking being used (no lock manager). Note that this is a private interface.

`noac`
Client is not caching attributes.

`nointr`
No interrupts allowed on hard mount.

`nocto`
No close-to-open consistency.

`retrans`
NFS retransmissions.

`rpctimesync`
RPC time sync.

`rsize`
Read buffer size in bytes.

`sec`
`sec` has one of the following values:

`dh`
`des`–style authentication (encrypted timestamps).

`krb5`
`kerberos v5`–style authentication.

`krb5i`
`kerberos v5`–style authentication with integrity.

`krb5p`
`kerberos v5`–style authentication with privacy.

`none`
No authentication.

`short`
Short hand UNIX–style authentication.

`sys`
UNIX-style authentication (UID, GID).

`soft`
Soft mount.

`timeo`
Initial NFS timeout, in tenths of a second.

`wsize`
Write buffer size in bytes.

The following mount flags are internal to the system:

`acl`
Server supports NFS_ACL.

`down`
Server is down.

`dynamic`
Dynamic transfer size adjustment.

`link`
Server supports links.

`mirrormount`
Mounted automatically by means of the `mirrormount` mechanism.

`printed`
"Not responding" message printed.

`readdironly`
Use `readdir` instead of `readdirplus`.

`referral`
Mounted automatically by means of the referral mechanism.

`symlink`
Server supports symbolic links.

The following flags relate to additional mount information:

`proto`
Protocol.

`vers`
NFS version.

The `-m` option also provides attribute cache timeout values. The following fields in `-m` output provide timeout values for attribute cache:

`acdirmax`
Maximum seconds to hold cached directory attributes.

`acdirmin`
Minimum seconds to hold cached directory attributes.

`acregmax`
Maximum seconds to hold cached file attributes.

`acregmin`
Minimum seconds to hold cached file attributes.

The following fields in `-m` output provide failover information:

`currserver`
Which server is currently providing NFS service. See the *System Administration Guide: IP Services* for additional details.

`failover`
How many times a new server has been selected.

`noresponse`
How many times servers have failed to respond.

`remap`
How many times files have been re-evaluated to the new server.

The fields in `-m` output shown below provide information on dynamic retransmissions. These items are displayed only where dynamic retransmission is in use.

`cur`
Current backed-off retransmission value, in milliseconds.

`dev`
Estimated deviation, in milliseconds.

`srtt`
The value for the smoothed round-trip time, in milliseconds.

Exit Status The following exit values are returned:

`0`
Successful completion.

`>0`
An error occurred.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/file-system/nfs

See Also `mount_nfs(1M)`, `attributes(5)`

Installing Oracle Solaris 11.1 Systems

System Administration Guide: IP Services

Name nscadm – network storage control utility

Synopsis

```
nscadm freeze device

nscadm unfreeze device

nscadm isfrozen device
```

Description The `nscadm` command performs several network storage control functions.

The `nscadm freeze` command closes existing references to the specified device, and blocks future accesses. This allows maintenance of virtual volume device drivers (for example, RAID 0, RAID 1, RAID 5) to be performed without shutting down the system.

The `nscadm unfreeze` command reverses the effects of `nscadm freeze` for the specified device.

The `nscadm isfrozen` command returns the current status of the specified device.

Options The `nscadm` command supports the following option.

`-h` Display the usage menu.

Operands The `nscadm` command line supports the following operand.

device
Specifies the storage device to be acted upon by `nscadm`.

Exit Status For the `freeze` and `unfreeze`, subcommands `nscadm` returns the following exit values:

`0` Success

`255` Error

For the `isfrozen` subcommand, `nscadm` returns the following exit values:

`0` Device is currently frozen.

`1` Device is not currently frozen.

`255` Error

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	storage/avs/avs-cache-management
Interface Stability	Committed

See Also `scmadm(1M)`, `attributes(5)`

Name nscd – name service cache daemon

Synopsis

```
/usr/sbin/nscd [-f configuration-file] [-g] [-e cachename, yes
 | no] [-i cachename]
```

Description The nscd daemon is a process that provides a cache for most name service requests. The service properties of the svc:/system/name-service/cache SMF service determine the behavior of the cache daemon. See nscd.conf(4).

nscd provides caching for the passwd(4), group(4), hosts(4), ipnodes(4), exec_attr(4), prof_attr(4), user_attr(4), ethers(4), rpc(4), protocols(4), networks(4), bootparams(4), auth_attr(4), services(4), netmasks(4), project(4) databases through standard libc interfaces, such as gethostbyname(3NSL), getipnodebyname(3SOCKET), gethostbyaddr(3NSL), and others. The shadow file is specifically not cached. getspnam(3C) calls remain uncached as a result.

Each cache has a separate time-to-live for its data. By default, modifying the local database (/etc/hosts, /etc/passwd, and so forth) causes that cache to become invalidated upon the next call to nscd.

The updating and refreshing of any of the services that svc:/system/name-service/cache is optionally dependent upon (listed below) causes nscd to restart, which effectively clears all caches.

- svc:/network/dns/client, see resolv.conf(4)
- svc:/network/nis/client, see ypbind(1M) and ypfiles(4)
- svc:/network/ldap/client, see ldapclient(1M)
- svc:/system/name-service/switch, see nsswitch.conf(4)

See nscd.conf(4).

nscd also acts as its own administration tool. If an instance of nscd is already running, commands are passed to the running version transparently.

When running with per-user lookups enabled (see nscd.conf(4)), nscd forks one and only one child process (that is, a per-user nscd) on behalf of the user making the request. The per-user nscd will use the credentials of the user to open a per-user connection to the name repository configured for the per-user style of lookups. The lookup will be performed in the child process. The results are cached in the process and are available only to the same user. The caches are managed exactly the same as the main nscd daemon manages its own caches. Subsequent requests from the user will be handled by that per-user nscd until it terminates. The per-user nscd uses a configurable inactivity time-to-live (TTL) value and terminates itself after the inactivity TTL expires.

The maximum number of per-user nscds that can be created by the main nscd is configurable (see nscd.conf(4)). After the maximum number of them are created, the main nscd will use an LRU algorithm to terminate less active child nscds as needed.

The main nscd daemon creates, monitors, and manages all the child nscds. It creates a user's own nscd upon receiving the user's first per-user lookup. When the nscd daemon is started, if per-user lookups are enabled, it checks to ensure all conditions are met before getting ready to create a per-user nscd. When the daemon is stopped, it terminates all the per-user nscds under its control.

Per-user nscds use the same configuration as the main nscd. They read and use the same default configuration file or the one specified with the -f command line option. Once the configuration is read, the per-user nscd will use it for its entire lifetime.

Options Several of the options described below require a *cachename* specification. Supported values for *cachename* are: passwd, group, hosts, ipnodes, exec_attr, prof_attr, user_attr, ethers, rpc, protocols, networks, bootparams, auth_attr, services, netmasks, printers, and project.

-f *configuration-file*
Causes nscd to read its configuration data from the specified file. This option is obsolete and will be removed in a future release.

-g
Prints current configuration and statistics to standard output. This is the only option executable by non-root users.

-e *cachename*, yes|no
Enables or disables the specified cache.

-i *cachename*
Invalidate the specified cache.

Examples **EXAMPLE 1** Stopping and Restarting the nscd Daemon.

```
example# svcadm disable system/name-service/cache

example# svcadm enable system/name-service/cache
```

Files /etc/nscd.conf Obsolete. Formerly determined the behavior of the cache daemon

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also svcs(1), svcadm(1M), ypbind(1M), getspnam(3C), gethostbyname(3NSL), getipnodebyname(3SOCKET), auth_attr(4), bootparams(4), ethers(4), exec_attr(4), group(4), hosts(4), netmasks(4), networks(4), nscd.conf(4), nsswitch.conf(4), passwd(4), prof_attr(4), project(4), protocols(4), resolv.conf(4), rpc(4), services(4), user_attr(4), attributes(5)

Notes The output from the `-g` option to `nscd` is subject to change. Do not rely upon it as a programming interface.

The `nscd` service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/system/name-service/cache
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. The service's status can be queried using the `svcs(1)` command.

The obsolete service `svc:/system/name-service-cache` has been retained for backwards compatibility with scripts that might reference it. Its only purpose is to serve as an `optional_all` dependency on the service `svc:/system/name-service/cache`. The obsolete service name will be removed in a future release.

Name nscfg – import, export name service configurations

Synopsis

```
/usr/sbin/nscfg command [cmd_options] [operands]

/usr/sbin/nscfg import [-fnvq] FMRI]

/usr/sbin/nscfg export [-nvq] FMRI]

/usr/sbin/nscfg unconfig [-nvq] FMRI]

/usr/sbin/nscfg validate [-vq] FMRI]

/usr/sbin/nscfg help
```

Description The `nscfg` utility imports or exports legacy name service configuration files into or out of the SMF repository. Given a valid SMF configuration and corresponding FMRI, `nscfg` will regenerate the legacy naming service configuration files, such as `nsswitch.conf`, `resolv.conf`, `nscd.conf`, and so forth, into their legacy locations. Alternatively, `nscfg` can import those same configuration files populating the SMF repository if it is currently unpopulated.

The following are SMF services that can be operands to `nscfg`. In many instances, the legacy configuration file that is replaced by the SMF service is listed.

`svc:/system/name-service/switch:default`
Legacy file: `/etc/nsswitch.conf`

Name service switch configuration (used by `nscd`).

`svc:/system/name-service/cache:default`
Legacy file: `/etc/nscd.conf`

Name service cache (`nscd`).

`svc:/network/dns/client:default`
Legacy file: `/etc/resolv.conf`

DNS naming service.

`svc:/network/nis/domain:default`
Legacy file: `/etc/defaultdomain`

Legacy file: `/var/yp/binding/$DOMAIN/*`

Shared NIS domain configuration. Used by all NIS services. Also (historical) shared use with LDAP naming services. Must be enabled when using `nis/client` or `ldap/client`.

`svc:/network/nis/client:default`
NIS client naming service (`ypbind` and related)

`svc:/network/ldap/client:default`
Legacy file: `/var/ldap/*`

LDAP client naming service (`ldap_cachemgr` and related).

`svc:/network/nis/server:default`
NIS server service (`ypserv`).

`svc:/network/nis/passwd:default`
NIS server `passwd` service (`rpc.yppasswd`).

`svc:/network/nis/xfr:default`
NIS server `xfr` service (`ypxfrd`)

`svc:/network/nis/update:default`
NIS server update service (`rpc.ypupdated`)

`svc:/system/name-service/upgrade:default`
Import legacy configuration files into SMF service.

Interaction with Location Profiles

Configuration for the following services is managed by location profiles:

```
svc:/system/name-service/switch:default
svc:/network/dns/client:default
svc:/network/nis/domain:default
svc:/network/nis/client:default
svc:/network/ldap/client:default
```

See `netcfg(1M)` for more information about location profiles.

These profiles are either fixed, meaning the network configuration is being managed in the traditional way, or reactive, meaning the network configuration is being managed automatically, reacting to changes in the network environment according to policy rules specified in the profiles.

When a fixed location (there can currently be only one, the `DefaultFixed` location) is active, changes made to the SMF repository, including those made by means of `nscfg`, will be applied to the location when it is disabled, and thus will be restored if that location is later re-enabled.

When a reactive location is active, changes should not be applied directly to the SMF repository; these changes will not be preserved in the location profile, and will thus be lost if the location is disabled, or if the system's network configuration, as managed by `svc:/network/physical:default` and `svc:/network/location:default`, is refreshed or restarted. Changes should instead be applied to the location itself, using the `netcfg(1M)` command; this will save the change to the location profile repository, and will also apply it to the SMF repository (if the change is made to the currently active location).

Sub-commands

The `nscfg` utility supports the subcommands described below. Options are described in the context of the subcommands.

`import` [`-fvq`] *FMRI*
If none of the SMF repository properties for the specified FMRI are currently populated, import the legacy configuration files associated with the specified FMRI into the SMF repository.

With `-f`, force the repopulation of the SMF repository with the legacy configuration, even if currently populated.

With `-v`, issue verbose progress messages during the requested operation. With `-q`, issue no error or other messages during the requested operation.

`export` [`-vq`] *FMRI*
Export the SMF configuration for the specified FMRI to legacy configuration files. This operation removes any existing affected legacy file(s) and generates new one(s) using the SMF configuration.

`unconfig` [`-vq`] *FMRI*
Unconfigure the SMF configuration for the specified FMRI. This operation resets the specified FMRI and any existing legacy file(s) to their initial unconfigured state.

With `-v`, issue verbose progress messages during the requested operation. With `-q`, issue no error or other messages during the requested operation.

`validate` [`-vq`] *FMRI*
Validate the SMF configuration for the specified FMRI. This operation checks the current configuration and verifies whether any errors exist.

With `-v`, issue verbose progress messages during the requested operation. With `-q`, issue no error or other messages during the requested operation.

Examples

EXAMPLE 1 Importing DNS Client Configuration

The following command imports the DNS client configuration, stored in `resolv.conf`, into the SMF repository.

```
# nscfg import svc:/network/dns/client:default
```

EXAMPLE 2 Exporting SMF LDAP Client Configuration

The following command exports the SMF LDAP client configuration to the legacy configuration files in `/var/ldap`.

```
# nscfg export svc:/network/ldap/client:default
```

EXAMPLE 3 Resetting Name Service Switch Configuration

The following command resets the name service switch configuration to its initial unconfigured state. The command generates no output and exits quietly with status only.

```
# nscfg unconfig -q svc:/system/name-service/switch:default
```

EXAMPLE 4 Validating LDAP Client Configuration

The following command validates the LDAP client configuration for errors or inconsistencies. The command generates no output and exits quietly with status only.

```
# nscfg validate -q svc:/network/ldap/client:default
```

Exit Status

`0`

Command successfully executed.

`1`

An error occurred.

`2`

Configuration unmodified, no change necessary.

`3`

No configuration to import.

Files

- `/etc/default/{nss, yppasswd}`
- `/etc/defaultdomain`
- `/etc/nscd.conf`
- `/etc/nsswitch.conf`
- `/etc/resolv.conf`
- `/var/ldap/ldap_client_cred`
- `/var/ldap/ldap_client_file`
- `/var/yp/binding/{`*nisdomainname*`}/ypservers`
- `/var/yp/NISLDAPmapping`
- `/var/yp/securenets`
- `/var/yp/updaters`

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os
Interface Stability	Uncommitted

See Also

`svcs(1)`, `ldapclient(1M)`, `netcfg(1M)`, `nscd(1M)`, `svcadm(1M)`, `svccfg(1M)`, `ypinit(1M)`, `defaultdomain(4)`, `nsswitch.conf(4)`, `resolv.conf(4)`, `ypfiles(4)`, `attributes(5)`

Name nsdb-list, nsdb-nces, nsdb-resolve-fsn – FedFS observability utilities

Synopsis

```
nsdb-nces [-l nsdb] [-r port]

nsdb-list [-l nsdb] [-r port] [-e nce]

nsdb-resolve-fsn [-l nsdb] [-r port] fsn
```

Description The tools described here permit observability into FedFS data stored in FedFS NSDBs (LDAP servers which store FedFS information).

`nsdb-nces` will list the naming contexts on the LDAP server that contains FedFS data, and the relative distinguished name of the FedFS container.

The `nsdb-list` command will list all FedFS data stored in the LDAP server, showing FedFS fileset names (FSNs) and all fileset locations (FSLs) for each.

`nsdb-resolve-fsn` will show the FSLs that correspond to the passed FSN.

Examples The following options are supported:

`-l` *nsdb*
: Specify the LDAP server implementing the NSDB.

`-r` *port*
: Specify the port on which the LDAP server implementing the NSDB is listening.

`-e` *nce*
: Specify the distinguished name of the container of FedFS information on the LDAP server implementing the NSDB; this will often be a single unambiguous location and need not be specified.

Examples **EXAMPLE 1** Using the NSDB Tools

The following sequence of commands illustrates the use of all of the NSDB tools.

```
# nsdb-nces
Host: nsdb.cthon.org:389
  namingContext 'dc=cthon,dc=org' is a FedFS NCE, DIT starts at ''

# nsdb-list
NSDB: nsdb.cthon.org:389, dc=cthon,dc=org
  FSN UUID: 7cc0bf04-5459-11e1-8083-80093d11d889
    FSL UUID: 7cc33c02-5459-11e1-8084-00093d11d889 = filer-a:/tmp
  FSN UUID: db48f160-5858-11e1-b459-80093d11d889
    FSL UUID: db4998c2-5858-11e1-b45a-00093d11d889 = filer-j:/tmp

# nsdb-resolve-fsn 7cc0bf04-5459-11e1-8083-80093d11d889
For FSN UUID 7cc0bf04-5459-11e1-8083-80093d11d889
  FSL UUID: 7cc33c02-5459-11e1-8084-00093d11d889
    Location: filer-a:/tmp
```

Exit Status

`0`
Successful completion.

`>0`
An error occurred.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	service/file-system/nfs
Interface Stability	Committed

See Also `nsdbparams(1M)`, `nfsref(1M)`, `attributes(5)`, `fedfs(5)`

Name nsdbparams – FedFS connection management utility

Synopsis

```
nsdbparams [-r port] [-e nce] [-t sectype] [-f certfile]
     [-D bind_DN] [-w bind_PW | -] nsdb

nsdbparams delete [-r port] nsdb

nsdbparams show [-r port] nsdb

nsdbparams list

nsdbparams get

nsdbparams set [-r port] nsdb
```

Description The `nsdbparams` command manages defaults and connection information for working with FedFS NSDBs (LDAP servers that store FedFS information).

The first form of the command, `nsdbparams update`, creates or updates a connection entry for the named NSDB. If the port number is not provided, the default LDAP port of 389 is used. If the NCE is not provided, the server will be queried to enumerate NCEs, and if only one is present it will be used. The LDAP bind DN must be provided if the entry is new. The LDAP bind password may be stored or not; if not stored, the password will be prompted for when needed. The `-w -` form may be used to force prompting for a password to be stored instead of placing it on the command line.

The second form of the command, `nsdbparams delete`, removes a connection entry for `nsdb:`*port* or `nsdb:389` if the port number is not provided.

The fourth form of the command, `nsdbparams list`, enumerates all connection entries.

The fifth form of the command, `nsdbparams get`, shows a system-wide default NSDB and port number.

The sixth form of the command, `nsdbparams set`, sets the system-wide default NSDB and port number, using the default LDAP port of 389 if one is not provided.

Options The following options are supported:

`-r` *port*
: Specify the port on which the LDAP server implementing the NSDB is listening.

`-e` *nce*
: Specify the distinguished name of the container of FedFS information on the LDAP server implementing the NSDB; this will often be a single unambiguous location and need not be specified.

`-D` *bind_DN*
: Specify the distinguished name of a user permitted to change the NSDB information.

`-w` *bind_PW* | –
: Specify the password for the bind DN user; use of hyphen (`-`) will force prompting.

`-t` *sectype*
Specify the security level used to contact the LDAP server. Value values are `FEDFS_SEC_NONE`, for no encryption, and `FEDFS_SEC_TLS` for a connection secured with `StartTLS` (RFC 4513). If a TLS connection is to be used, the `-f` *certfile* argument must be present or a certificate must already be stored.

`-f` *certfile*
Specify a TLS certificate to be used to secure a connection with RFC 4513 `StartTLS` when `FEDFS_SEC_TLS` is used.

Examples

EXAMPLE 1 Using `nsdbparams` to Set Up Communications

The following example sets up communications with an NSDB called `nsdb.cthon.org` and makes it the default NSDB:

```
# nsdbparams update -D cn=Manager,dc=cthon,dc=org -w cthon.org \
nsdb.cthon.org
# nsdbparams show nikon.us.example.com
nikon.us.example.com:389
     default bind DN: cn=Manager,dc=cthon,dc=org
     default bind PW: cthon.org
     default NCE: dc=cthon,dc=org
     sectype: FEDFS_SEC_NONE
# nsdbparams set nsdb.cthon.org
# nsdbparams get
default nsdb: nsdb.cthon.org
default port: 389
```

Exit Status

`0`
Successful completion.

`>0`
An error occurred.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/file-system/nfs
Interface Stability	Committed

See Also

`nfsref(1M)`, `nsdb-list(1M)`, `nsdb-nces(1M)`, `attributes(5)`, `fedfs(5)`

Name nsdb-update-nci – FedFS initialization utility

Synopsis `nsdb-update-nci [-l nsdb] [-r port] [-D bind_DN] [-w bind_PW] nce`

Description The `nsdb-update-nci` command marks a distinguished name on an LDAP server as a container for FedFS data by adding the `fedfsNsdbContainerInfo` object class to the root of the naming context and setting the `fedfsNcePrefixR` attribute to point to the relative DN from the root of the naming context.

Options The following options are supported:

`-l` *nsdb*
Specify the LDAP server implementing the NSDB.

`-r` *port*
Specify the port on which the LDAP server implementing the NSDB is listening.

`-D` *bind_DN*
Specify the distinguished name of a user permitted to change the NSDB information.

`-w` *bind_PW*
Specify the password for the bind DN user; use of a hyphen (`-`) will force prompting.

Operands The following operand is supported.

nce
The distinguished name of the container to be used to store FedFS information on the LDAP server implementing the NSDB.

Examples **EXAMPLE 1** Using the Utility

The following sequence illustrates the use of `nsdb-update-nci`.

```
# nsdb-update-nci -l  localhost  -r  389  -D  cn=Manager  -w \
     cthon.org dc=cthon,dc=org adding new entry "dc=cthon,dc=org"
NCE entry created

# nsdb-nces Host: localhost:389
namingContext 'dc=cthon,dc=org' is a FedFS NCE, DIT starts at ''
```

Exit Status `0`
Successful completion.

`1`
An error occurred.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/file-system/nfs

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Committed

See Also nsdb-list(1M), nsdbparams(1M), nfsref(1M), attributes(5), fedfs(5)

Name nslookup – query Internet name servers interactively

Synopsis `nslookup [-option] [name | -] [server]`

Description The `nslookup` utility is a program to query Internet domain name servers. It has two modes: interactive and non-interactive. Interactive mode allows the user to query name servers for information about various hosts and domains or to print a list of hosts in a domain. Non-interactive mode is used to print just the name and requested information for a host or domain.

Parameters Interactive mode is entered in the following cases:

1. No arguments are given (the default name server is used).
2. The first argument is a hyphen (-) and the second argument is the host name or Internet address of a name server.

Non-interactive mode is used when the name or Internet address of the host to be looked up is given as the first argument. The optional second argument specifies the host name or address of a name server.

Options can also be specified on the command line if they precede the arguments and are prefixed with a hyphen. For example, to change the default query type to host information, and the initial timeout to 10 seconds, type:

```
nslookup -query=hinfo  -timeout=10
```

Interactive Commands

`host` [*server*]
Look up information for host using the current default server or using server, if specified. If host is an Internet address and the query type is A or PTR, the name of the host is returned. If host is a name and does not have a trailing period, the search list is used to qualify the name. To look up a host not in the current domain, append a period to the name.

`server` *domain*
`lserver` *domain*
Change the default server to *domain*; `lserver` uses the initial server to look up information about *domain*, while `server` uses the current default server. If an authoritative answer can't be found, the names of servers that might have the answer are returned.

`root`
Not implemented.

`finger`
Not implemented.

`ls`
Not implemented.

`view`
Not implemented.

`help`
Not implemented.

`?`
Not implemented.

`exit`
Exits the program.

`set` *keyword*[=*value*]
This command is used to change state information that affects the lookups. Valid keywords are:

`all`
Prints the current values of the frequently used options to `set`. Information about the current default server and host is also printed.

`class=`*value*
Change the query class to one of:

`IN`
the Internet class

`CH`
the Chaos class

`HS`
the Hesiod class

`ANY`
wildcard

The class specifies the protocol group of the information.

(Default = `IN`; abbreviation = `cl`)

`[no]debug`
Turn on or off the display of the full response packet and any intermediate response packets when searching.

(Default = `nodebug`; abbreviation = `[no]deb`)

`[no]d2`
Turn debugging mode on or off. This displays more about what `nslookup` is doing.

(Default = `nod2`)

`domain=`*name*
Sets the search list to *name*.

`[no]search`
If the lookup request contains at least one period but doesn't end with a trailing period, append the domain names in the domain search list to the request until an answer is received.

(Default = `search`)

`port=`*value*
Change the default TCP/UDP name server port to *value*.

(Default = `53`; abbreviation = `po`)

`querytype=`*value*
`type=`*value*
Change the top of the information query.

(Default = `A`; abbreviations = `q`, `ty`)

`[no]recurse`
Tell the name server to query other servers if it does not have the information. (Default = `recurse`; abbreviation = `[no]rec`)

`retry=`*number*
Set the number of retries to number.

`timeout=`*number*
Change the initial timeout interval for waiting for a reply to number seconds.

`[no]vc`
Always use a virtual circuit when sending requests to the server.

(Default = `novc`)

`[no]fail`
Try the next nameserver if a nameserver responds with `SERVFAIL` or a referral (`nofail`) or terminate query (`fail`) on such a response.

(Default = `nofail`)

Files `/etc/resolv.conf`
resolver configuration file

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	network/dns/bind
Interface Stability	Volatile

See Also dig(1M), host(1M), named(1M), attributes(5)

See the BIND 9 *Administrator's Reference Manual*. As of the date of publication of this man page, this document is available at https://www.isc.org/software/bind/documentation.

Notes BIND 9 nslookup is deprecated and not as full featured as its BIND 8 version. For more features and functionality refer to dig(1M).

nslookup and dig(1M) now report "Not Implemented" as NOTIMP rather than NOTIMPL. This will have impact on scripts that are looking for NOTIMPL.

Name nsupdate – Dynamic DNS update utility

Synopsis

```
nsupdate [-dv] [-y keyname:secret | -k keyfile] [-t timeout]
     [-u udptimeout] [-r udpretries] [filename]
```

Description The `nsupdate` utility submits Dynamic DNS Update requests as defined in RFC 2136 to a name server. This utility allows resource records to be added or removed from a zone without manually editing the zone file. A single update request can contain requests to add or remove more than one resource record.

Zones that are under dynamic control with `nsupdate` or a DHCP server should not be edited by hand. Manual edits could conflict with dynamic updates and cause data to be lost.

The resource records that are dynamically added or removed with `nsupdate` must be in the same zone. Requests are sent to the zone's master servers identified by the `MNAME` field of the zone's SOA record.

Transaction signatures can be used to authenticate the Dynamic DNS updates using the TSIG resource record type described in RFC 2845. The signatures rely on a shared secret that should only be known to `nsupdate` and the name server. Currently, the only supported encryption algorithm for TSIG is HMAC-MD5, which is defined in RFC 2104. Once other algorithms are defined for TSIG, applications will need to ensure that they select the appropriate algorithm as well as the key when authenticating each other. For instance, suitable `key` and `server` statements would be added to `/etc/named.conf` so that the name server can associate the appropriate secret key and algorithm with the IP address of the client application that will be using TSIG authentication. The `nsupdate` utility does not read `/etc/named.conf`.

The `nsupdate` utility uses the `-y` or `-k` option to provide the shared secret needed to generate a TSIG record for authenticating Dynamic DNS update requests. These options are mutually exclusive. See OPTIONS.

Options The following options are supported:

`-d` Operate in debug mode. This provides tracing information about the update requests that are made and the replies received from the name server.

`-k` *keyfile* Read the shared secret from the file *keyfile*, whose name is of the form `K{`*name*`}.+157.+{`*random*`}.private`. For historical reasons, the file `K{`*name*`}.+157.+{`*random*`}.key` must also be present.

`-r` *udpretries* Set the number of UDP retries. The default is 3 retries. If *udpretries* is set to zero, only one update request is made.

`-t` *timeout* Set *timeout* interval in seconds before update is aborted. The default is 300 seconds. A setting of zero disables the timeout.

`-u` *udptimeout* Set interval in seconds between UDP retires, the default is 3 seconds. A setting of zero causes the interval to be calculated based on the timeout (`-t`) and the number of UDP retries (`-r`).

`-v` Use a TCP connection. Using a TCP connection could be preferable when a batch of update requests is made. By default, `nsupdate` uses UDP to send update requests to the name server.

`-y` *keyname*:*secret* Generate a signature from *keyname*:*secret*, where*keyname* is the name of the key and *secret* is the base64 encoded shared secret.

Use of the `-y` option is discouraged because the shared secret is supplied as a command line argument in clear text and could be visible in the output from `ps(1)` or in a history file maintained by the user's shell.

Input Format The `nsupdate` utility reads input from *filename* or the standard input. Each command is supplied on exactly one line of input. Some commands are for administrative purposes. The others are either update instructions or prerequisite checks on the contents of the zone. These checks set conditions that some name or set of resource records (RRset) either exists or is absent from the zone. These conditions must be met if the entire update request is to succeed. Updates will be rejected if the tests for the prerequisite conditions fail.

Every update request consists of zero or more prerequisites and zero or more updates. This condition allows a suitably authenticated update request to proceed if some specified resource records are present or missing from the zone. A blank input line (or the `send` command) causes the accumulated commands to be sent as one Dynamic DNS update request to the name server.

The command formats and their meaning are as follows:

`server` *servername* [*port*]
Send all dynamic update requests to the name server *servername*. When no `server` statement is provided, `nsupdate` sends updates to the master server of the correct zone. The `MNAME` field of that zone's SOA record identifies the master server for that zone. The *port* argument is the port number on *servername* where the dynamic update requests get sent. If no port number is specified, the default DNS port number of 53 is used.

`local` *address* [*port*]
Send all dynamic update requests using the local *address*. When no `local` statement is provided, `nsupdate` sends updates using an address and port chosen by the system. The *port* argument can also be used to make requests come from a specific port. If no port number is specified, the system assigns one.

`zone` *zonename*
Specify that all updates are to be made to the zone *zonename*. If no `zone` statement is provided, `nsupdate` attempts to determine the correct zone to update based on the rest of the input.

`class` *classname*
Specify the default class. If no class is specified the default class is IN.

`key` *name secret*
Specify that all updates are to be TSIG signed using the *name secret* pair. The `key` command overrides any key specified on the command line with `-y` or `-k`.

`prereq nxdomain` *domain-name*
Require that no resource record of any type exists withthe name *domain-name*.

`prereq yxdomain` *domain-name*
Require that *domain-name* exists (has as at least one resource record, of any type).

`prereq nxrrset` *domain-name* [*class*] *type*
Require that no resource record exists of the specified *type*, *class* and *domain-name*. If *class* is omitted, IN (internet) is assumed.

`prereq yxrrset` *domain-name* [*class*] *type*
Require that a resource record of the specified *type*, *class* and *domain-name* must exist. If *class* is omitted, IN (internet) is assumed.

`prereq yxrrset` *domain-name* [*class*] *type data...*
The *data* from each set of prerequisites of this form sharing a common *type*, *class*, and *domain-name* are combined to form a set of RRs. This set of RRs must exactly match the set of RRs existing in the zone at the given *type*, *class*, and *domain-name*. The *data* are written in the standard text representation of the resource record's RDATA.

`update delete` *domain-name* [*ttl*] [*class*] [*type* [*data...*]]
Delete any resource records named *domain-name*. If *type* and *data* are provided, only matching resource records are removed. The internet class is assumed if *class* is not supplied. The *ttl* is ignored, and is only provided for compatibility.

`update add` *domain-name ttl* [*class*] *type data...*
Add a new resource record with the specified *ttl*, *class* and *data*.

`show`
Display the current message, containing all of the prerequisites and updates specified since the last send.

`send`
Sends the current message. This is equivalent to entering a blank line.

`answer`
Displays the answer.

Lines beginning with a semicolon are comments and are ignored.

Examples

EXAMPLE 1 Inserting and Deleting Resource Records from the Zone

The examples below show how `nsupdate` could be used to insert and delete resource records from the `example.com` zone. Notice that the input in each example contains a trailing blank line so that a group of commands are sent as one dynamic update request to the master name server for `example.com`.

EXAMPLE 1 Inserting and Deleting Resource Records from the Zone *(Continued)*

```
# nsupdate
> update delete oldhost.example.com A
> update add newhost.example.com 86400 A 172.16.1.1
> send
```

Any A records for `oldhost.example.com` are deleted. An A record for `newhost.example.com` with IP address 172.16.1.1 is added. The newly-added record has a 1 day TTL (86400 seconds).

EXAMPLE 2 Adding CNAME Only If No Records Exist

The following command adds a CNAME only if no records already exist for it.

```
# nsupdate
> prereq nxdomain nickname.example.com
> update add nickname.example.com 86400 CNAME somehost.example.com
> send
```

The prerequisite condition gets the name server to check that there are no resource records of any type for `nickname.example.com`. If there are, the update request fails. If this name does not exist, a `CNAME` for it is added. This action ensures that when the `CNAME` is added, it cannot conflict with the long-standing rule in RFC 1034 that a name must not exist as any other record type if it exists as a `CNAME`. (The rule has been updated for DNSSEC in RFC 4035 to allow `CNAME`s to have `RSIG`, `DNSKEY`, and `NSEC` records.)

Files

`/etc/resolv.conf`	used to identify default name server
`K{`*name*`}.+157.+{`*random*`}.key`	base-64 encoding of HMAC-MD5 key created by `dnssec-keygen`(1M).
`K{`*name*`}.+157.+{`*random*`}.private`	base-64 encoding of HMAC-MD5 key created by `dnssec-keygen(1M)`

Bugs The TSIG key is redundantly stored in two separate files. This is a consequence of `nsupdate` using the DST library for its cryptographic operations and could change in future releases.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	service/network/dns/bind
Interface Stability	Volatile

See Also `named(1M)`, `dnssec-keygen(1M)`, `attributes(5)`

RFC 2136, *RFC 3007*, *RFC 2104*, *RFC 2845*, *RFC 1034*, *RFC 2535*, *RFC 2931*, *RFC 4035*

Name ntfscat – display NTFS files and streams on the standard output

Synopsis `ntfscat [`*options*`] `*device*` [`*file*`]`

Description The `ntfscat` command reads a file or stream from an NTFS volume and display the contents on the standard output.

The case of the filename passed to `ntfscat` is ignored.

Options Supported options are listed below. Most options have both single-letter and full-name forms. Multiple single-letter options that do not take an argument can be combined. For example, `-fv` is the equivalent of `-f -v`. A full-name option can be abbreviated to a unique prefix of its name.

`-a`, `--attribute` *type*
Display the contents of a particular attribute type. By default, the unnamed `$DATA` attribute will be shown. The attribute can be specified by a number in decimal or hexadecimal, or by name.

```
Hex     Decimal Name
0x10    16      "$STANDARD_INFORMATION"
0x20    32      "$ATTRIBUTE_LIST"
0x30    48      "$FILE_NAME"
0x40    64      "$OBJECT_ID"
0x50    80      "$SECURITY_DESCRIPTOR"
0x60    96      "$VOLUME_NAME"
0x70    112     "$VOLUME_INFORMATION"
0x80    128     "$DATA"
0x90    144     "$INDEX_ROOT"
0xA0    160     "$INDEX_ALLOCATION"
0xB0    176     "$BITMAP"
0xC0    192     "$REPARSE_POINT"
0xD0    208     "$EA_INFORMATION"
0xE0    224     "$EA"
0xF0    240     "$PROPERTY_SET"
0x100   256     "$LOGGED_UTILITY_STREAM"
```

The attribute names can be specified without the leading dollar sign (`$`) symbol. If you use the `$` symbol, you must quote the name to prevent the shell from interpreting the name.

`-f`, `--force`
Overrides some sensible defaults, such as not using a mounted volume. Use this option with caution.

`-h`, `--help`
Show a list of options with a brief description of each.

`-i`, `--inode` *num*
Specify a file by its inode number instead of its name.

`-n`, `--attribute-name` *name*
Display the attribute identified by *name*.

`-q`, `--quiet`
Suppress some debug, warning, and error messages.

`-V`, `--version`
Show the version number, copyright, and license information.

`-v`, `--verbose`
Display more debug, warning, and error messages.

Examples

EXAMPLE 1 Displaying Contents of File in Root

The following command displays the contents of a file in the root of an NTFS volume.

```
# ntfscat /dev/dsk/c0d0p1 boot.ini
```

EXAMPLE 2 Displaying Contents of File in Subdirectory

The following command displays the contents of a file in a subdirectory of an NTFS volume.

```
# ntfscat /dev/dsk/c0d0p1 /winnt/system32/drivers/etc/hosts
```

EXAMPLE 3 Display Contents of an Attribute

The following command displays the contents of the `$INDEX_ROOT` attribute of the root directory (inode 5).

```
# ntfscat /dev/dsk/c0d0p1 -a INDEX_ROOT -i 5
```

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/file-system/ntfsprogs
Interface Stability	Uncommitted

See Also

`ntfsls(1M)`, `ntfsprogs(1M)`, `parted(1M)`, `attributes(5)`

`http://wiki.linux-ntfs.org`

Authors

`ntfscat` was written by Richard Russon, Anton Altaparmakov and Szabolcs Szakacsits.

Name ntfsclone – clone, image, restore, or rescue an NTFS

Synopsis

```
ntfsclone [options] source

ntfsclone --save-image [options] source

ntfsclone --resotore-image [options] source

ntfsclone --metadata [options] source
```

Description The `ntfsclone` utility efficiently clones (which includes copy, save, backup, and restore operations) or rescues an NTFS filesystem to a sparse file, an image, a device (partition), or to standard output. It works at disk sector level and copies only the written data (that is, not empty space). Unused disk space becomes zero (cloning to sparse file), encoded with control codes (saving in special image format), left unchanged (cloning to a disk/partition) or filled with zeros (cloning to standard output).

`ntfsclone` can be useful in making backups—taking an exact snapshot of an NTFS filesystem—and restoring it later on. It also can be used to test NTFS read/write functionality and allows you to troubleshoot users' issues using the clone, without the risk of destroying the original file system.

If not using the special image format (see section of the same name below), the clone is an exact copy of the original NTFS file system, from sector to sector. Thus, it can also be mounted just like the original NTFS filesystem. For example, if you clone to a file and the kernel has a loopback device and NTFS support, then the file can be mounted using:

```
# mount -t ntfs -o loop ntfsclone.img
```

Windows Cloning You must exercise great care to copy, move or restore a system or boot partition to another computer, or to a different disk or partition (for example, `/dev/dsk/c0d0p1` to `/dev/dsk/c0d0p2`, `/dev/dsk/c0d0p1` to `/dev/dsk/c0d1p1` or to a different disk sector offset).

Under most circumstances, to enable Windows to boot you must copy, move, or restore NTFS to the same partition that has the following characteristics as the original partition and disk:

- starts at the same sector
- on the same type of disk
- having the same BIOS legacy cylinder setting

The `ntfsclone` utility guarantees an exact copy of NTFS but does not deal with booting issues. This is by design: `ntfsclone` is a file system, not a system, utility. Its goal is only NTFS cloning, not Windows cloning. Because of this, `ntfsclone` can be used as a very fast and reliable building block for Windows cloning, but is not a complete answer. You can find useful tips on NTFS cloning at the NTFS web site, `http://wiki.linux-ntfs.org`.

Sparse Files A file containing unallocated blocks (holes) is referred to as a “sparse file”. The reported size of such files is always higher than the disk space consumed by them. The `du(1)` command reports the real disk space used by a sparse file. The holes are always read as zeros. All major Linux file

systems, such as, `ext2`, `ext3`, `reiserfs`, `Reiser4`, JFS, and XFS support sparse files. However, the ISO 9600 CD-ROM file system, as one example, does not.

Special Image Format

It is recommended that you save an NTFS filesystem to a special image format. Instead of representing unallocated blocks as holes, they are encoded using control codes. Thus, the image saves space without requiring sparse file support. The image format is ideal for streaming file system images over the network. The disadvantage of the special image format is that you cannot mount the image directly; you must first restore it.

To save an image using the special image format, use the `-s` or the `--save-image` option. To restore an image, use the `-r` or the `--restore-image` option. Note that you can restore images from standard input by using a hyphen (`-`) as the source file.

Metadata-only Cloning

Using the `-m` or `-–metadata` option, `ntfsclone` can save only the NTFS metadata and the clone still will be mountable. In this usage, all non-metadata file content is lost; reading back the data results in all zeros.

The metadata-only image can be compressed very well, usually to a size in the range of 1 to 8 MB. It is convenient to transfer such an image for investigation and troubleshooting.

In metadata-only mode, `ntfsclone` saves none of the user's data, which includes the resident user's data embedded into metadata. All is filled with zeros. Moreover, all the file timestamps, and deleted and unused spaces inside the metadata are filled with zeros. Thus, this mode is inappropriate, for example, for forensic analyses.

Note that filenames are not removed. Because a filename might contain sensitive information, consider the possibities for breaches of security or privacy before sending out a metadata-only image.

Options

Supported options are listed below. Most options have both single-letter and full-name forms. Multiple single-letter options that do not take an argument can be combined. For example, `-fv` is the equivalent of `-f -v`. A full-name option can be abbreviated to a unique prefix of its name.

`-f`, `-–force`
Forces `ntfsclone` to proceed, even if the filesystem is marked “dirty” following a consistency check.

`-h`, `-–help`
Show a list of options with a brief description of each one.

`-i`, `-–ignore-fs-check`
Ignore the result of the file system consistency check. This option can be used only with the `--meta-data` option. Any clusters that cause an inconsistency are saved.

`-m`, `-–metadata`
Clone only metadata. With this option, you must clone only to a file.

`-o`, `-–output` *file*
Clone NTFS to the non-existent *file*. If *file* is a hyphen (`-`), clone to the standard output.

`-O`, `--overwrite` *file*
Clone NTFS to *file*, overwriting *file* if it already exists.

`--rescue`
Ignore disk read errors so that a disk having bad sectors, for example, a failing disk, can be rescued with minimal impact on the disk. `ntfsclone` works at the lowest, sector level in this mode, enabling more data to be rescued. The contents of the unreadable sectors are filled with the question mark (`?`) character; the beginning of such sectors are marked by the string: `BadSector`.

`-r`, `--restore-image` *source*
Restore from the special image format specified by *source*. If *source* is a hyphen (`-`), the image is read from the standard input.

`-s`, `--save-image`
Save to the special image format. In terms of space usage and speed, this is the most efficient option if imaging is done to the standard output. This option is useful for image compression, encryption, or streaming through a network.

Examples

EXAMPLE 1 Cloning with Overwrite Option

The following command clones with the `--overwrite` option.

```
# ntfsclone --overwrite /dev/dsk/c0d2p1 /dev/dsk/c0d0p1
```

EXAMPLE 2 Saving to Special Image Format

The following command clones to the special image format to its original partition.

```
# ntfsclone --save-image --output backup.img /dev/dsk/c0d0p1
```

EXAMPLE 3 Restoring from a Special Image File

The following command restores an NTFS from a special image file.

```
# ntfsclone --restore-image --overwrite /dev/dsk/c0d0p1 backup.img
```

EXAMPLE 4 Saving to a Compressed Image

The following command saves an NTFS to a compressed image file.

```
# ntfsclone --save-image -o - /dev/dsk/c0d0p1
```

EXAMPLE 5 Restoring from a Compressed Image

The following command restores an NTFS volume from a compressed image file.

```
# gunzip -c backup.img.gz | \
ntfsclone --restore-image --overwrite /dev/dsk/c0d0p1 -
```

EXAMPLE 6 Backing up to a Remote Host Using `ssh`

The following command backs up to a remote host, using `ssh(1)`. Note that `ssh` will probably require a password.

```
# ntfsclone --save-image --output - /dev/dsk/c0d0p1 | \
gzip -c | ssh host 'cat > backup.img.gz'
```

EXAMPLE 7 Restoring from a Remote Host Using `ssh`

The following command restores from a remote host, using `ssh(1)`. Note that `ssh` will probably require a password.

```
# ssh host 'cat backup.img.gz' | gunzip -c  | \
ntfsclone --restore-image --overwrite /dev/dsk/c0d0p1 -
```

EXAMPLE 8 Streaming an Image File from a Web Server

The following command streams an image file from a web server and restore it to a partition.

```
# wget -qO - http://server/backup.img | \
ntfsclone --restore-image --overwrite /dev/dsk/c0d0p1 -
```

EXAMPLE 9 Cloning to a New File

The following command clones an NTFS volume to a non-existent file.

```
# ntfsclone --output ntfs-clone.img /dev/dsk/c0d0p1
```

EXAMPLE 10 Packing NTFS Metadata

The following command packs NTFS metadata into an image file. Note that `bzip2` takes a much longer time than `gzip`, but produces an archive that is up to ten times smaller than the latter produces.

```
# ntfsclone --metadata --output ntfsmeta.img /dev/dsk/c0d0p1
bzip2 ntfsmeta.img
```

EXAMPLE 11 Unpacking NTFS Metadata

The following command unpacks NTFS metadata into a sparse file.

```
# bunzip2 -c ntfsmeta.img.bz2 | \
cp --sparse=always /proc/self/fd/0 ntfsmeta.img
```

Exit Status

The return code is zero on success, non-zero otherwise.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/file-system/ntfsprogs

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Uncommitted

See Also du(1), ssh(1), ntfsresize(1M), ntfsundelete(1M), parted(1M), attributes(5)

http://wiki.linux-ntfs.org

Authors ntfsclone was written by Szabolcs Szakacsits with contributions from Per Olofsson (special image format support) and Anton Altaparmakov.

Name ntfscluster – identify files in a specified region of an NTFS volume

Synopsis `ntfscluster [`*options*`] `*device*

Description The `ntfscluster` utility has three modes of operation: `info`, `sector`, and `cluster`, described as follows.

`Info`
The default mode, `info` is currently not implemented. It will display general information about the NTFS volume when it is working.

`Sector`
The `sector` mode displays a list of files that have data in the specified range of sectors. This mode is put in effect by the `--sector` option.

`Cluster`
The `cluster` mode displays a list of files that have data in the specified range of clusters. When the cluster size is one sector, this is equivalent to the `sector` mode of operation. This mode is put in effect by the `--cluster` option.

Options Supported options are listed below. Most options have both single-letter and full-name forms. Multiple single-letter options that do not take an argument can be combined. For example, `-fv` is the equivalent of `-f -v`. A full-name option can be abbreviated to a unique prefix of its name.

`-c`, `--cluster` *range*
Any files whose data is in this range of clusters will be displayed.

`-F`, `--filename` *filename*
Display information about *filename*.

`-f`, `--force`
Overrides some sensible defaults, such as not working with a mounted volume. Use this option with caution.

`-h`, `--help`
Show a list of options with a brief description of each.

`-I`, `--inode` *num*
Show information about this inode.

`-i`, `--info`
This option is not yet implemented.

`-q`, `--quiet`
Suppress some debug, warning, and error messages.

`-s`, `--sector` *range*
Any files whose data is in this range of sectors will be displayed.

`-V`, `--version`
Show the version number, copyright, and license information.

`-v`, `--verbose`
Display more debug, warning, and error messages.

Examples

EXAMPLE 1 Displaying Information About a Volume

The following command displays information about the volume `/dev/dsk/c0d0p1`.

```
# ntfscluster /dev/dsk/c0d0p1
```

EXAMPLE 2 Displaying List of Files in a Cluster Range

The following command looks for files in the first 500 clusters of `/dev/dsk/c0d0p1`.

```
# ntfscluster -c 0-500 /dev/dsk/c0d0p1
```

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/file-system/ntfsprogs
Interface Stability	Uncommitted

See Also

`ntfsinfo(1M)`, `ntfsprogs(1M)`, `parted(1M)`, `attributes(5)`

`http://wiki.linux-ntfs.org`

Authors

`ntfscluster` was written by Richard Russon, with contributions from Anton Altaparmakov.

Name ntfscmp – compare two NTFS file systems and report the differences

Synopsis `ntfscmp [options] device1 device2`

Description The `ntfscmp` utility compares all aspects of two NTFS file systems and reports all differences it finds. The file systems can be on block devices or in image files. `ntfscmp` can be used for volume verification. However, its primary purpose is to be an efficient development tool, used to quickly locate, identify, and check the correctness of the metadata changes made to NTFS.

If one is interested only in the NTFS metadata changes, it can be useful to compare the metadata images created by using the `--metadata` option of `ntfsclone(1M)` to eliminate the usually uninteresting timestamp changes.

The terse output of `ntfscmp` is intentional, because the provided information is sufficient to determine exact differences. More copious output can be obtained by using `diff(1)` to compare the verbose output of `ntfsinfo(1M)` for each reported inode.

Options Supported options are listed below. Options have both single-letter and full-name forms.

`-h`, `--help`
Display help and exit.

`-P`, `--no-progress-bar`
Do not show progress bars.

`-v`, `--verbose`
Display more debug, warning, and error messages.

Exit Status The exit code is `0` on success, non-zero otherwise.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/file-system/ntfsprogs
Interface Stability	Uncommitted

See Also `diff(1)`, `ntfscat(1M)`, `ntfsclone(1M)`, `ntfsinfo(1M)`, `ntfsprogs(1M)`, `parted(1M)`, `attributes(5)`

`http://wiki.linux-ntfs.org`

Authors `ntfscmp` was written by Szabolcs Szakacsits.

Name ntfscp – copy file to an NTFS volume

Synopsis `ntfscp [`*options*`] device source_file destination`

Description The `ntfscp` utility copies files to an NTFS volume. *destination* (see Synopis) can be either a file or a directory. If *destination* is a directory specified by name, *source_file* is copied into this directory. If *destination* is a directory specified by inode number, an unnamed data attribute is created for this inode and *source_file* is copied into it. Consider possible negative consequence before specifying a directory by inode number: it is unusual to have an unnamed data stream in a directory.

Data Streams All data on NTFS is stored in streams, which can have names. A file can have more than one data stream, but exactly one must have no name. The size of a file is the size of its unnamed data stream. Usually, when you do not specify a stream name, you are seeking access to the unnamed data stream. If you want access to a named data stream, you need to add `:`*stream_name* to the filename. For example, by opening `some.mp3:artist` you will open stream `artist` in `some.mp3`. In an operating system, such as Windows, that prevents you from accessing named data streams, you need to use some program like `FAR` or utilities from `cygwin` to access those streams.

Options Supported options are listed below. Most options have both single-letter and full-name forms. Multiple single-letter options that do not take an argument can be combined. For example, `-fv` is the equivalent of `-f -v`. A full-name option can be abbreviated to a unique prefix of its name.

`-a`, `--attribute` *num*
Write to attribute designated by *num*.

`-f`, `--force`
Overrides some sensible defaults, such as not working with a mounted volume. Use this option with caution.

`-h`, `--help`
Show a list of options with a brief description of each one.

`-i`, `--inode`
Treat *destination* (see Synopsis) as inode number.

`-N`, `--attr-name` *name*
Write to attribute with this name.

`-n`, `--no-action`
Use this option to make a test run before doing the actual copy operation. Volume will be opened read-only and no write will be done.

`-q`, `--quiet`
Suppress some debug, warning, and error messages.

`-V`, `--version`
Show the version number, copyright, and license information.

`-v`, `--verbose`
Display more debug, warning, and error messages.

Examples

EXAMPLE 1 Copying from Home to Root Directory

The following command copies `new_boot.ini` from `/home/user` as `boot.ini` to the root of an `/dev/dsk/c0d0p1` NTFS volume.

```
# ntfscp /dev/dsk/c0d0p1 /home/user/new_boot.ini boot.ini
```

EXAMPLE 2 Copying a Stream

The following command copies `myfile` to `C:\some\path\myfile:stream` (assume that `/dev/dsk/c0d0p1` drive designator is `C`).

```
# ntfscp -N stream /dev/dsk/c0d0p1 myfile /some/path
```

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/file-system/ntfsprogs
Interface Stability	Uncommitted

See Also

`ntfsresize(1M)`, `ntfsprogs(1M)`, `parted(1M)`, `attributes(5)`

`http://wiki.linux-ntfs.org`

Authors

`ntfscp` was written by Yura Pakhuchiy, with contributions from Anton Altaparmakov and Hil Liao.

Name ntfsfix – fix common errors and force operating system to check NTFS

Synopsis `ntfsfix [`*options*`] `*device*

Description The `ntfsfix` utility fixes some common NTFS problems. Note that it is not a version of `chkdsk`. It repairs some fundamental NTFS inconsistencies, resets the NTFS journal file, and schedules an NTFS consistency check for the next reboot of the operating system.

Run `ntfsfix` on an NTFS volume if you think it was damaged by the operating system or in some other way and it cannot be mounted.

Options Supported options are listed below.

`-h`, `--help`
Show a list of options with a brief description of each.

`-V`, `--version`
Show the version number, copyright, and license information.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/file-system/ntfsprogs
Interface Stability	Uncommitted

See Also `mkntfs(1M)`, `ntfsprogs(1M)`, `parted(1M)`, `attributes(5)`

`http://wiki.linux-ntfs.org`

Authors `ntfsfix` was written by Anton Altaparmakov, with contributions from Szabolcs Szakacsits.

Name ntfsinfo – dump a file's attributes

Synopsis `ntfsinfo [`*options*`] `*device*

Description The `ntfsinfo` utility dumps the attributes of inode *inode-number* or the file *path-filename* and/or information about the MFT (`-m` option). Run `ntfsinfo` without arguments for a full list of options.

Options Supported options are listed below. Most options have both single-letter and full-name forms. Multiple single-letter options that do not take an argument can be combined. For example, `-fv` is the equivalent of `-f -v`. A full-name option can be abbreviated to a unique prefix of its name.

`-F`, `--file` *file*
Show information about *file*.

`-f`, `--force`
Overrides some sensible defaults, such as not overwriting an existing file. Use this option with caution.

`-h`, `--help`
Show a list of options with a brief description of each.

`-i`, `--inode` *num*
Show information about inode identified by *num*.

`-m`, `--mft`
Show information about the volume.

`-q`, `--quiet`
Suppress some debug, warning, and error messages.

`-t`, `--notime`
Do not display timestamps in the output.

`-V`, `--version`
Show the version number, copyright, and license information.

`-v`, `--verbose`
Display more debug, warning, and error messages.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/file-system/ntfsprogs
Interface Stability	Uncommitted

See Also ntfsprogs(1M), ntfsundelete(1M), parted(1M), attributes(5)

http://wiki.linux-ntfs.org

Authors ntfsinfo was written by Matthew J. Fanto, Anton Altaparmakov, Richard Russon, Szabolcs Szakacsits, Yuval Fledel, Yura Pakhuchiy and Cristian Klein.

Name ntfslabel – display or change the label on an NTFS file system

Synopsis

```
ntfslabel [options] device [new_label]
```

Description The `ntfslabel` utility displays or changes the file system label on the NTFS file system located on *device* (see Synopsis).

If the optional argument *new_label* is not present, `ntfslabel` displays the current file system label.

If the optional argument *new_label* is present, `ntfslabel` sets the file system label to be *new_label*. NTFS file system labels can be at most 128 Unicode characters long; if *new_label* is longer than 128 Unicode characters, `ntfslabel` truncates it and displays a warning message.

It is also possible to set the file system label using the `-L` option of `mkntfs(1M)` during creation of the file system.

Options Supported options are listed below. Most options have both single-letter and full-name forms. Multiple single-letter options that do not take an argument can be combined. For example, `-fv` is the equivalent of `-f -v`. A full-name option can be abbreviated to a unique prefix of its name.

`-f`, `--force`
Overrides some sensible defaults, such as not working with a mounted volume. Use this option with caution.

`-h`, `--help`
Show a list of options with a brief description of each.

`-n`, `--no-action`
Do not actually write a new label to disk.

`-q`, `--quiet`
Suppress some debug, warning, and error messages.

`-V`, `--version`
Show the version number, copyright, and license information.

`-v`, `--verbose`
Display more debug, warning, and error messages.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/file-system/ntfsprogs
Interface Stability	Uncommitted

See Also mkntfs(1M), ntfsprogs(1M), parted(1M), attributes(5)

http://wiki.linux-ntfs.org

Authors ntfslabel was written by Matthew J. Fanto, with contributions from Anton Altaparmakov and Richard Russon.

Name ntfsls – list directory contents on an NTFS file system

Synopsis `ntfsls [`*options*`] `*device*

Description The `ntfsls` utility lists information about the files specified by the `PATH` option (the root directory by default). *device* (see Synopsis) is the special file corresponding to the device (for example, `/dev/dsk/c0d0p0`*num*) or an NTFS image file.

Options Supported options are listed below. Most options have both single-letter and full-name forms. Multiple single-letter options that do not take an argument can be combined. For example, `-fv` is the equivalent of `-f -v`. A full-name option can be abbreviated to a unique prefix of its name.

`-a`, `--all`
Display all files. If this option is not specified file names in the POSIX namespace are not displayed.

`-F`, `--classify`
Append one of the indicators shown below to entries.

`* / = @ |`

`-f`, `--force`
Force execution. For example, this option is necessary to run on an NTFS partition stored in a normal file.

`-h`, `--help`
Display usage information and exit.

`-i`, `--inode`
Display the inode number of each file. This is the MFT reference number, in NTFS terminology.

`-l`, `--long`
Use a long listing format.

`-p`, `--path` *path*
The directory whose contents to list or the file (including the path) about which to display information.

`-q`, `--quiet`
Suppress some debug, warning, and error messages.

`-R`, `--recursive`
Show the contents of all directories beneath the specified directory.

`-s`, `--system`
Unless this options is specified, all files beginning with a dollar sign character will not be listed, as these files are usually system files.

`-v`, `--verbose`
Display more debug, warning, and error messages.

`-V`, `--version`
Display the `ntfsls` version number and exit.

`-x`, `--dos`
Display short file names, that is, files in the DOS namespace, instead of long file names, that is, files in the WIN32 namespace.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/file-system/ntfsprogs
Interface Stability	Uncommitted

See Also , `ntfsprogs(1M)`, `parted(1M)`, `attributes(5)`

`http://wiki.linux-ntfs.org`

Authors This version of `ntfsls` was written by Lode Leroy, Anton Altaparmakov, Richard Russon, Carmelo Kintana and Giang Nguyen.

Name ntfsprogs – list of NTFS tools

Synopsis `ntfsprogs`

Description `ntfsprogs` is the name of a suite of NTFS utilities based around a shared library. The tools are available for free and come with full source code. The tools are listed below.

`mkntfs(1M)`
Create an NTFS filesystem.

`ntfscat(1M)`
Dump a file's content to the standard output.

`ntfsclone(1M)`
Efficiently clone, backup, restore, or rescue NTFS.

`ntfscluster(1M)`
Locate the files that use the specified sectors or clusters.

`ntfscmp(1M)`
Compare two NTFS file systems and report the differences.

`ntfscp(1M)`
Copy a file to an NTFS volume.

`ntfsfix(1M)`
Check and fix some common errors, clear the log file, and make the operating system perform a thorough check next time it boots.

`ntfsinfo(1M)`
Show information about NTFS or one of the files or directories within it.

`ntfslabel(1M)`
Show, or set, an NTFS filesystem's volume label.

`ntfsls(1M)`
List information about files in a directory residing on an NTFS.

`ntfsmount(1M)` (not a SunOS man page)
Read-write NTFS user space driver.

`ntfsresize(1M)`
Resize NTFS without losing data.

`ntfsundelete(1M)`
Recover deleted files from NTFS.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/file-system/ntfsprogs
Interface Stability	Uncommitted

See Also parted(1M), attributes(5)

http://wiki.linux-ntfs.org

Authors The tools were written by Anton Altaparmakov, Carmelo Kintana, Cristian Klein, Erik Sornes, Giang Nguyen, Holger Ohmacht, Lode Leroy, Matthew J. Fanto, Per Olofsson, Richard Russon, Szabolcs Szakacsits, Yura Pakhuchiy, and Yuval Fledel.

Name ntfsresize – resize an NTFS file system without data loss

Synopsis

```
ntfsresize [options] --info device

ntfsresize [options] [--size size[k|M|G]] device
```

Description The `ntfsresize` program safely resizes Windows XP, Windows Server 2003, Windows 2000, Windows NT4 and Longhorn NTFS filesystems without data loss. All NTFS versions used by 32-bit and 64-bit Windows "operating systems" are supported. Defragmentation is not required prior to resizing, because `ntfsresize` can relocate any data if needed, without risking data integrity.

`ntfsresize` can be used to shrink or enlarge any NTFS file system located on an unmounted device (usually a disk partition). The new file system will have a size that you specify. The size parameter can have one of the optional modifiers `k`, `M`, `G`, denoting, respectively, kilobytes, megabytes, or gigabytes. `ntfsresize` conforms to the SI, ATA, an IEEE standards and the disk manufacturers by supporting `k=10`^{3}, `M=10`^{6} and `G=10`^{9}.

If both `---info` and `---size` options are omitted then the NTFS file system will be enlarged to the underlying device size.

To resize a file system on a partition, you must resize both the file system and the partition, by editing the partition table on the disk. Similarly to other command-line file system resizers, `ntfsresize` does not manipulate the size of the partitions. To do that you must use a disk partitioning tool, such as `fdisk(1M)`. Alternatively, you could use one of the many user friendly partitioners that uses `ntfsresize` internally. Such partitioners include, among others, Mandriva's DiskDrake, QTParted, SUSE/Novell's YaST Partitioner, IBM's EVMS, GParted, or Debian/Ubuntu's Partman.

Back up your data and your partition table before using any partitioning tool. For an NTFS file system, you can use `ntfsclone(1M)` as a means of backup.

To shrink an NTFS partition, first use `ntfsresize` to shrink the size of the file system. Then use a utility such as `fdisk(1M)` to shrink the size of the partition by deleting the partition and recreating it with the smaller size. Do not make the partition smaller than the new size of NTFS; otherwise, you will not be able to boot from that partition. If you mistakenly made a too-small partition, you would have to recreate the partition to be as large as newly sized NTFS file system.

To enlarge an NTFS file system, you must first enlarge the size of the underlying partition. You can use `fdisk(1M)` to delete the partition and recreate it with a larger size. Make sure the newly sized partition does not overlap with any other partition. Then use `ntfsresize` to enlarge the file system.

When recreating a partition, make sure you create it at the same starting sector and with the same partition type as was used in the partition you are replacing. Otherwise, you will not be able to access your file system. Use the `fdisk u` command to switch from the default cylinder

unit to the reliable sector unit. Also, if the bootable flag was set in the old partition, make sure to set it in the recreated partiton. Otherwise, you might not be able to boot from the new partition.

Extended Description

There are a handful of very rarely met restrictions in the use of `ntfsresize`. An example of such a restriction occurs with a file system stored on a disk having unknown bad sectors. Relocation of the first `MFT` extent and resizing into the middle of a `$MFTMirr` extent are not supported. These cases are detected and resizing is restricted to a safe size or the closest safe size is displayed.

Upon completion of a resizing, `ntfsresize` schedules an NTFS consistency check. In Windows, this check is performed by `chkdsk`. Upon the first subsequent reboot into Windows, you will note `chkdsk` running in a blue background. This is normal. Windows might force a quick reboot after the consistency check. Depending on your hardware configuration, Windows might alert you to a systems setting change and recommend or require a reboot. Acknowledge the message and reboot a second time.

Options

Supported options are listed below. Most options have both single-letter and full-name forms. Multiple single-letter options that do not take an argument can be combined. For example, `-fv` is the equivalent of `-f -v`. A full-name option can be abbreviated to a unique prefix of its name.

`-b`, `--bad-sectors`
By default, `ntfsresize` exits upon encountering bad sectors. This option allows the utility to proceed in spite of such sectors.

Prior using this option, it is strongly recommended that you use `ntfsclone(1M)` with the `--rescue` option to make a backup, then, in Windows, run `chkdsk /f /r volume`: from the command line. If the disk guarantee displays as valid, then replace it, as it is defective. Note that no software can repair bad sector errors. The most that can be done is to work around these defects.

This option has no effect if a disk has no bad sectors.

`-f`, `--force`
`ntfsresize` always marks a file system for consistency check before a real (not using `-–no-action`) resize operation and it leaves that way for extra safety. Thus, if an NTFS file system was marked by `ntfsresize`, it is safe to use this option. You must use this option, if you need to resize several times without booting into Windows between each resizing step.

`-h`, `--help`
Display usage information and exit.

`-i`, `--info`
Used when you want to shrink a file system. Causes `ntfsresize` to determine the smallest shrunken file system size supported. Most of the time the smallest size is the space already used on the file system. `ntfsresize` does not shrink a file system to a smaller size than what is returned by this option. Depending on several factors, it might be unable to shrink to this

theoretical size. Although the integrity of your data should be never at risk, it is nevertheless strongly recommended to make a test run by using the `--no-action` option before actual resizing.

Based on testing, the smallest attainable size is approximately space used in the file system plus 20–200 MB. Note also that Windows might need an additional 50–100 MB to boot safely.

This option never causes any changes to the file system; the partition is opened read-only.

`-n`, `--no-action`
Use this option to make a test run before doing the resize operation. Volume will be opened read-only and `ntfsresize` displays what it would do if it were to resize the file system. Proceed with the actual resizing only if the test run passed.

`-P`, `--no-progress-bar`
Do not display progress bars during `ntfsresize` operation.

`-s`, `--size` *size*[k|M|G]]
Resize file system to *size* bytes. The new file system will have a size that you specify. The size parameter can have one of the optional modifiers `k`, `M`, `G`, denoting, respectively, kilobytes, megabytes, or gigabytes. `ntfsresize` conforms to the SI, ATA, an IEEE standards and the disk manufacturers by supporting `k=10`^{3}, `M=10`^{6} and `G=10`^{9}. Before performing an actual resizing, run `ntfsresize` with the `--no-action` option, along with this option, first.

`-v`, `--verbose`
Display copious output.

`-V`, `--version`
Display the version number of `ntfsresize`.

Exit Status Display zero on success, non-zero otherwise.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/file-system/ntfsprogs
Interface Stability	Uncommitted

See Also `fdisk(1M)`, `ntfsclone(1M)`, `parted(1M)`, `attributes(5)`

`http://wiki.linux-ntfs.org`

Notes In Linux version 2.6, with partitions that have been manipulated by `parted(1M)`, use of `ntfsresize` preceded corruption of partition tables, which resulted in unbootable Windows systems. This occurred even if the NTFS file system was consistent. This problem is independent of NTFS and, thus, `ntfsresize`. Moreover, `ntfsresize` never touches the

partition table. Under the conditions just described, you can, in the BIOS, change `Disk Access Mode` to `LBA` to regain the ability to boot. For further discussion of this condition see the `ntfsresize` FAQ at: `http://mlf.linux.rulez.org/mlf/ezaz/ntfsresize.html`.

Authors `ntfsresize` was written by Szabolcs Szakacsits, with contributions from Anton Altaparmakov and Richard Russon.

Name ntfsundelete – recover a deleted file from an NTFS volume

Synopsis `ntfsundelete [`*options*`] `*device*

Description The `ntfsundelete` utility can, under the right circumstances, recover a deleted file from an NTFS volume. The command has three modes of operation:

`Scan`
The default mode, `scan` simply reads an NTFS Volume and looks for files that have been deleted. It then displays a list, giving the inode number, name, and size of each deleted file.

`Undelete`
The undelete mode takes the files either matching the regular expression (option `-m`) or specified by the *inode-expressions* and recovers as much of the data as possible. It saves the result to another location.

`Copy`
The "wizard's" option. Saves a portion of the MFT to a file, which can be useful when debugging `ntfsundelete`.

There are many circumstances under which `ntfsundelete` is unable to recover a file. For example, consider the following scenario. When a file is deleted the MFT Record is marked as not in use and the bitmap representing the disk usage is updated. If the power is not turned off immediately, the free space, where the file used to reside might get overwritten. Worse, the MFT Record might be reused for another file. If this happens, it is impossible to tell where the file was on disk.

Even if all the clusters of a file are not in use, there is no guarantee that they have not been overwritten by some short-lived file.

`ntfsundelete` cannot recover compressed or encrypted files. During a scan, it will display such a file as being 0% recoverable.

Locale In NTFS, all filenames are stored as Unicode. A filename is converted into the current locale for display by `ntfsundelete`. The utility has successfully displayed Chinese pictogram filenames and then correctly recovered them.

Extended MFT Records In rare circumstances, a single MFT Record will not be large enough to hold the metadata describing a file (a file would have to be in hundreds of fragments for this to happen). In these cases, one MFT record might hold the filename, while another will hold the information about the data. `ntfsundelete` will not try and piece together such records. It will simply list unnamed files with data.

Recovered File's Size and Creation Date To recover a file, `ntfsundelete` has to read the file's metadata. Unfortunately, when a file is deleted, the metadata can be left in an inconsistent state. For example, the file size might be recorded as zero; the creation date of a file might be set to the time it was deleted or to a random time. In such situations, `ntfsundelete` picks the largest file size it finds and writes that to disk. It also tries to set the file's creation date to the last-modified date. This date might be the correct last modified date, or something unexpected.

Options Supported options are listed below. Most options have both single-letter and full-name forms. Multiple single-letter options that do not take an argument can be combined. For example, `-fv` is the equivalent of `-f -v`. A full-name option can be abbreviated to a unique prefix of its name.

`-b`, `--byte` *num*
Fill in the parts of unrecoverable file clusters with byte represented by *num*. The default is zeros.

`-C`, `--case`
Make filename search, when attempting a match with the `--match` option, case-sensitive. The default filename search is case-insensitive.

`-c`, `--copy` *range*
This "wizard" option writes a block of MFT FILE records to a file. The default file is mft which will be created in the current directory. This option can be combined with the `--output` and `--destination` options.

`-d`, `--destination` *dir*
Specify the location of the output file for the `--copy` and `--undelete` options.

`-f`, `--force`
Overrides some sensible defaults, such as not overwriting an existing file. Use this option with caution.

`-h`, `--help`
Show a list of options with a brief description of each one.

`-i`, `--inodes` *range*
Recover the files within the specified range of inode numbers. *range* can be a single inode number, several numbers separated by commas, or a range separated by a dash (`-`).

`-m`, `--match` *pattern*
Filter the output by looking only for filenames that match *pattern*. The pattern can include the wildcards `?`, matching exactly one character, or `*`, matching zero or more characters. By default, the matching is case-insensitive. To make the search case-sensitive, use the `--case` option.

`-O`, `--optimistic`
Recover parts of the file even if they are currently marked as in use.

`-o`, `--output` *file*
Set the name of the output file created by the `--copy` or `--undelete` options.

`-P`, `--parent`
Display the parent directory of a deleted file.

`-p`, `--percentage` *num*
Filter the output of the `--scan` option by matching only files with *num* percent of recoverable content.

`-q`, `--quiet`
Reduce the amount of output to a minimum. This option is not useful with the `--scan` option.

`-s`, `--scan`
Search through an NTFS volume and display a list of files that could be recovered. This is the default action of `ntfsundelete`. This list can be filtered by filename, size, percentage recoverable, or last modification time, using the `--match`, `--size`, `--percent`, and `--time` options, respectively.

In the output from this option, the `%age` (percentage) field displays how much of a file can potentially be recovered.

`-S`, `--size` *range*
Filter the output of the `--scan` option by looking for a particular range of file sizes. *range* can be specified as two numbers separated by a hyphen (`-`). A unit of size can be abbreviated using the suffixes `k`, `m`, `g`, and `t`, for kilobytes, megabytes, gigabytes, and terabytes respectively.

`-t`, `--time` *since*
Filter the output of the `--scan` option. Match only files that have been altered since this time. The time must be given as number and a suffix of `d`, `w`, `m`, or `y` for, respectively, days, weeks, months, or years.

`-T`, `--truncate`
The default behavior of `ntfsundelete` is to round *up* a file's size to the nearest cluster (which will be a multiple of 512 bytes). In cases where the utility has complete data about the size of a file, this option restores the file to exactly that size.

`-u`, `--undelete`
Specifies undelete mode. You can specify the files to be recovered using by using `--match` or `--inodes` options. This option can be combined with `--output`, `--destination`, and `--byte`.

When the file is recovered it will be given its original name, unless the `--output` option is used.

`-v`, `--verbose`
Increase the amount of output that `ntfsundelete` displays.

`-V`, `--version`
Display the version number, copyright, and license for `ntfsundelete`.

Examples

EXAMPLE 1 Searching for Deleted Files

The following command searches for deleted files on a specific device.

```
# ntfsundelete /dev/dsk/c0d0p1
```

EXAMPLE 2 Scanning for Files Matching a Wildcard

The following command searches for deleted files that match `*.doc`.

```
# ntfsundelete /dev/dsk/c0d0p1 -s -m '*.doc'
```

EXAMPLE 3 Searching for Files of a Certain Size

The following command looks for deleted files between 5000 and 6000000 bytes, with at least 90% of the data recoverable, on `/dev/dsk/c0d0p1`.

```
# ntfsundelete /dev/dsk/c0d0p1 -S 5k-6m -p 90
```

EXAMPLE 4 Searching for Recently Changed Files

The following command searches for deleted files altered in the last two days.

```
# ntfsundelete /dev/dsk/c0d0p1 -t 2d
```

EXAMPLE 5 Specifying an Inode Range

The following command undeletes inodes 2, 5 and 100 to 131 of device `/dev/sda1`.

```
# ntfsundelete /dev/sda1 -u -i 2,5,100-131
```

EXAMPLE 6 Specifying an Output File and Directory

The following command undeletes inode number 3689, names the file `work.doc`, and stores it in the user's home directory.

```
# ntfsundelete /dev/dsk/c0d0p1 -u -i 3689 -o work.doc -d ~
```

EXAMPLE 7 Saving MFT Records

The following command saves MFT records 3689 to 3690 to a file `debug`.

```
# ntfsundelete /dev/dsk/c0d0p1 -c 3689-3690 -o debug
```

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/file-system/ntfsprogs
Interface Stability	Uncommitted

See Also

`ntfsclone(1M)`, `ntfsresize(1M)`, `parted(1M)`, `attributes(5)`

`http://wiki.linux-ntfs.org`

Authors

`ntfsundelete` was written by Richard Russon and Holger Ohmacht, with contributions from Anton Altaparmakov.

Name nwamd – network auto-magic daemon

Synopsis

```
/lib/inet/nwamd
```

Description `nwamd` is a system daemon to manage network interfaces.

This daemon is started automatically by the network/physical:default service and should not be invoked directly. It does not constitute a programming interface.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os
Interface Stability	Volatile

See Also `svcs(1)`, `netcfgd(1M)`, `netadm(1M)`, `netcfg(1M)`, `svcadm(1M)`, `attributes(5)`, `nwam(5)`, `smf(5)`

See also `nwam-manager(1M)`, available in the JDS/GNOME man page collection.

Notes The networking service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/network/physical:default
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. The service's status can be queried using the `svcs(1)` command.

Name obpsym – Kernel Symbolic Debugging for OpenBoot Firmware

Synopsis

```
modload -p misc/obpsym
```

Description `obpsym` is a kernel module that installs OpenBoot callback handlers that provide kernel symbol information to OpenBoot. OpenBoot firmware user interface commands use the callbacks to convert numeric *addresses* to kernel symbol names for display purposes, and to convert kernel symbol names to numeric *literals* allowing symbolic names to be used as input arguments to user interface commands.

Once `obpsym` is installed, kernel symbolic names may be used anywhere at the OpenBoot firmware's user interface command prompt in place of a literal (numeric) string. For example, if `obpsym` is installed, the OpenBoot firmware commands `ctrace` and `dis` typically display symbolic names and offsets in the form *modname:symbolname + offset*. User interface Commands such as `dis` can be given a kernel symbolic name such as `ufs:ufs_mount` instead of a numeric address.

Placing the command

```
forceload: misc/obpsym
```

into the `system(4)` file forces the kernel module `misc/obpsym` to be loaded and activates the kernel callbacks during the kernel startup sequence.

`obpsym` may be useful as a kernel debugger in situations where other kernel debuggers are not useful. For example, on SPARC machines, if `obpsym` is loaded, you may be able to use the OpenBoot firmware's `ctrace` command to display symbolic names in the stack backtrace after a watchdog reset.

Kernel Symbolic Name Syntax The syntax for a kernel symbolic name is:

[*module-name* `:`] *symbol-name*

Where *module-name* is the name of the kernel module that the symbol *symbol-name* appears in. A `NULL` module name is taken as "all modules, in no particular order" by `obpsym`. The module name `unix` is equivalent to a `NULL` module name, so that conflicts with words defined in the firmware's vocabulary can be avoided.

Typically, OpenBoot firmware reads a word from the input stream and looks the word up in its internal *vocabulary* before checking if the word is a *literal*. Thus, kernel symbols, such as `reset` may be given as `unix:reset` to avoid the unexpected side effect of the firmware finding and executing a matching word in its vocabulary.

Files

`/etc/system`	System configuration information file.
`/platform/`*platform-name*`/kernel/misc/obpsym`	Platform-specific kernel symbol information.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/library/processor

See Also `kmdb(1)`, `uname(1)`, `kernel(1M)`, `modload(1M)`, `modunload(1M)`, `system(4)`, `attributes(5)`

OpenBoot 2.x Command Reference ManualOpenBoot 2.x Command Reference Manual

Warnings Some OpenBoot firmware user interface commands may use system resources incompatibly with the way they are used by the Unix kernel. These commands and the use of this feature as a kernel debugger may cause interactions that the Unix kernel is not prepared to deal with. If this occurs, the Unix kernel and/or the OpenBoot firmware user interface commands may react unpredictably and may panic the system, or may hang or may cause other unpredictable results. For these reasons, the use of this feature is only minimally supported and recommended to be used only as a kernel debugger of last resort.

If a breakpoint or watchpoint is triggered while the console frame buffer is powered off, the system can crash and be left in a state from which it is difficult to recover. If one of these is triggered while the monitor is powered off, you will not be able to see the debugger output.

Notes *platform-name* can be found using the `-i` option of `uname(1)`

`obpsym` is supported only on architectures that support OpenBoot firmware.

On some systems, OpenBoot must be completely RAM resident so the `obpsym` symbol callback support can be added to the firmware, if the firmware doesn't include support for the symbol callbacks. On these systems, `obpsym` may complain that it requires that “you must use `ramforth` to use this module”.

See the *OpenBoot 2.x Command Reference ManualOpenBoot 2.x Command Reference Manual* for details on how to use the *ramforth* command, how to place the command into *nvramrc*, and how to set *use-nvramrc?* to `true`. On systems with version 1.x OpenBoot firmware, *nvramrc* doesn't exist, and the *ramforth* command must be typed manually after each reset, in order to use this module.

Once installed, the symbol table callbacks can be disabled by using the following OpenBoot firmware command:

```
0 0 set-symbol-lookup
```

Name oplhpd – Hot plug daemon for SPARC Enterprise Server line

Synopsis

```
/usr/platform/SUNW,SPARC-Enterprise/lib/sparcv9/lib/oplhpd
```

Description The hot plug daemon for SPARC Enterprise Servers is a daemon process that runs on the SUNW,SPARC-Enterprise family of servers. The daemon is started by the service management facility (see `smf(5)`) and communicates with the service processor when hot plug PCI cassettes change their dynamic reconfiguration state.

The service FMRI for `oplhpd` is:

```
svc:/platform/sun4u/oplhpd:default
```

A domain supports only one running `oplhpd` process at a time.

Errors OPLHPD uses `syslog(3C)` to report status and error messages. All of the messages are logged with the LOG_DAEMON facility.

Error messages are logged with the LOG_ERR and LOG_NOTICE priorities, and informational messages are logged with the LOG_DEBUG priority. The default entries in the `/etc/syslog.conf` file log all of the OPLHPD error messages to the `/var/adm/messages log`.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/domain-configuration/sparc-enterprise, SUNWdcsr
Interface Stability	Committed

See Also `svcs(1)`, `inetadm(1M)`, `svcadm(1M)`, `syslog(3C)`, `syslog.conf(4)`, `attributes(5)`, `smf(5)`

Name parted – partition manipulation program

Synopsis `parted [`*options*`] [`*device* `[`*command* `[`*options*`...]...]]`

Description `parted` is a disk partitioning and partition resizing program. It allows you to create, destroy, resize, move, and copy `ext2`, `linux-swap`, `FAT`, `FAT32`, and `reiserfs` partitions. It can create, resize, and move Macintosh HFS partitions, as well as detect `jfs`, `ntfs`, `ufs`, and `xfs` partitions. It is useful for creating space for new operating systems, reorganizing disk usage, and copying data to new hard disks.

This manual page documents `parted` briefly. Complete `parted` documentation is distributed with the package in "GNU Info" format.

`parted` is implemented with a set of top-level options and a set of subcommands, most of which have their own options and operands. These subcommands are described below. `parted` has an optional operand:

device The block device to be used. When none is given, `parted` uses the first block device it finds.

If you invoke `parted` without an argument, the program displays a command prompt.

Options The following options are supported:

`-a` *alignment-type*, `--align` *alignment-type*
Sets alignment for newly created partitions. Valid alignment types are:

`none`
Use the minimum alignment allowed by the disk type.

`cylinder`
Align partitions to cylinders.

`minimal`
Use minimum alignment as given by the disk topology information. This and the *opt* value will use layout information provided by the disk to align the logical partition table addresses to actual physical blocks on the disks. The *min* value is the minimum alignment needed to align the partition properly to physical blocks, which avoids performance degradation.

`optimal`
Use optimum alignment as given by the disk topology information. This aligns to a multiple of the physical block size in a way that guarantees optimal performance.

`-h`, `--help`
Displays a help message.

`-l`, `--list`
Lists partition layout on all block devices.

`-m`, `--machine`
Displays machine-parseable output.

`-s`, `--script`
Never prompts for user intervention.

`-v`, `--version`
Displays the version number.

Sub-commands If you omit a subcommand in a `parted` command line, the utility issues a command prompt.

`check` *partition*
Do a simple check on *partition*.

`cp` [*source-device*] *source dest*
Copy the source partition's filesystem on *source-device* (or the current device if no other device was specified) to the *dest* partition on the current device.

`help` *command*
Display general help, or help on a command, if specified.

`mkfs` *partition fs-type*
Make a filesystem *fs-type* on partition. *fs-type* can be one of `fat16`, `fat32`, `ext2`, `linux-wap`, or `reiserfs`.

`mklabel` *label-type*
Create a new disk label (partition table) of *label-type*. *label-type* should be one of `bsd`, `dvh`, `gpt`, `loop`, `mac`, `msdos`, `pc98`, or `sun`.

`mkpart` *part-type* [*fs-type*] *start end*
Make a *part-type* partition with file system *fs-type* (if specified), beginning at *start* and ending at *end* (by default, in megabytes). *fs-type* can be one of `fat16`, `fat32`, `ext2`, `HFS`, `linux-swap`, `NTFS`, `reiserfs`, or `ufs`. *part-type* should be one of `primary`, `logical`, or `extended`.

`mkpartfs` *part-type fs-type start end*
Make a *part-type* partition with file system *fs-type*, beginning at *start* and ending at *end* (by default, in megabytes).

Use of this subcommand is discouraged. Instead use `mkpart` to create an empty partition, and then use external tools such as `mke2fs(8)` (part of Linux) to create the filesystem.

`move` *partition start end*
Move partition so that it begins at *start* and ends at *end*. Note that `move` never changes the minor number.

`name` *partition name*
Set the name of partition to *name*. This option works only on Mac, PC98, and GPT disk labels. The name can be placed in quotes, if necessary.

`print`
Display the partition table.

`quit`
Exit from `parted`.

`rescue` *start end*
Rescue a lost partition that was located somewhere between *start* and *end*. If a partition is found, `parted` will ask if you want to create an entry for it in the partition table.

`resize` *partition start end*
Resize the file system on *partition* so that it begins at *start* and ends at *end* (by default, in megabytes).

`rm` *partition*
Delete *partition*.

`select` *device*
Choose *device* as the current device to edit. *device* should usually be a Solaris or Linux hard disk device, but it can be a partition, software raid device, or an SVM or LVM logical volume if necessary.

`set` *partition flag state*
Change the state of the *flag* on *partition* to *state*. Supported flags are: `boot`, `root`, `swap`, `hidden`, `raid`, `lvm`, `lba`, and `palo`. *state* should be either `on` or `off`.

`unit` *unit*
Set *unit* as the unit to use when displaying locations and sizes, and for interpreting those given by the user when not suffixed with an explicit unit. *unit* can be one of `s` (sectors), `B` (bytes), `kB`, `MB`, `GB`, `TB`, `%` (percentage of device size), `cyl` (cylinders), `chs` (cylinders, heads, sectors), or `compact` (megabytes for input, and a human-friendly form for output).

`version`
Display version information and a copyright message.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/storage/parted
Interface Stability	Uncommitted

See Also

`fdisk(1M)`, `mkfs(1M)`, `attributes(5)`

The `parted` program is fully documented in the `info(1)` format GNU partitioning software manual.

Notes `ext3` filesystem functionality does not currently work. To manage `ext3` type filesystems use tools like `resize2fs(8)` or `mke2fs(8)` (both part of Linux). Note that the currently supported `ext2` filesystem will be deprecated once `ext3` support is finalized. Further note that `ext3` support will have limited functionality that is yet to be defined. Use tools like `resize2fs(8)` and `mke2fs(8)` to manage these types of filesystems.

To manually resize an `ext3` filesystem or a partition, use `resize2fs(8)`, `fdisk(8)`, or similar tools. For LVM situations, you will need to use the LVM commands to resize the LVM elements.

Author This manual page was written by Timshel Knoll for the Debian GNU/Linux system. It is here adapted for the Solaris operating system.

Name pbind – control and query bindings of processes or LWPs

Synopsis

```
pbind -b processor_id pid [/lwpid]...

pbind [-q] [pid [/lwpid]]...

pbind -Q [processor_id]...

pbind -u pid [/lwpid]...

pbind -U [processor_id]...
```

Description `pbind` controls and queries bindings of processes and LWPs (lightweight processes) to processors. `pbind` can also remove processor bindings that were previously established.

When an LWP is bound to a processor, it will be executed only by that processor except when the LWP requires a resource that is provided only by another processor. The binding is not exclusive, that is, the processor is free to execute other LWPs as well.

Bindings are inherited, so new LWPs and processes created by a bound LWP will have the same binding. Binding an interactive shell to a processor, for example, binds all commands executed by the shell.

Superusers may bind or unbind any process or LWP, while other users can bind or unbind any process or LWP for which they have permission to signal, that is, any process that has the same effective user ID as the user.

Options The following options are supported:

`-b` *processor_id*
: Binds all or a subset of the LWPs of the specified processes to the processor *processor_id*. Specify *processor_id* as the processor ID of the processor to be controlled or queried. *processor_id* must be present and on-line. Use the `psrinfo` command to determine whether or not *processor_id* is present and on-line. See `psrinfo(1M)`.

`-q`
: Displays the bindings of the specified processes or of all processes. If a process is composed of multiple LWPs which have different bindings and the LWPs are not explicitly specified, the bindings of only one of the bound LWPs will be displayed. The bindings of a subset of LWPs can be displayed by appending "/lwpids" to the process IDs. Multiple LWPs may be selected using "-" and "," delimiters. See EXAMPLES.

`-Q`
: Displays the LWPs bound to the specified list of processors, or all LWPs with processor bindings. For processes composed of multiple LWPs, the bindings of individual LWPs will be displayed.

`-u`
: Removes the bindings of all or a subset of the LWPs of the specified processes, allowing them to be executed on any on-line processor.

`-U`
Removes the bindings of all LWPs bound to the specified list of processors, or to any processor if no argument is specified.

Operands The following operands are supported:

pid
The process ID of the process to be controlled or queried.

lwpid
The set of LWP IDs of the specified process to be controlled or queried. The syntax for selecting LWP IDs is as follows:

```
2,3,4-8       LWP IDs 2, 3, and 4 through 8
-4            LWPs whose IDs are 4 or below
4-            LWPs whose IDs are 4 or above
```

processor_id
The processor ID of the processor to be controlled or queried.

Examples **EXAMPLE 1** Binding Processes

The following example binds processes 204 and 223 to processor 2:

```
example% pbind -b 2 204 223
process id 204: was 2, now 2
process id 223: was 3, now 2
```

EXAMPLE 2 Unbinding a Process

The following example unbinds process 204:

```
example% pbind -u 204
```

EXAMPLE 3 Querying Bindings

The following example queries bindings. It demonstrates that process 1 is bound to processor 0, process 149 has at least one LWP bound to CPU3, and process 101 has no bound LWPs.

```
example% pbind -q 1 149 101
process id 1: 0
process id 149: 3
process id 101: not bound
```

EXAMPLE 4 Querying LWP Bindings

The following example queries bindings of LWPs. It demonstrates that LWP 1 of process 149 is bound to CPU3, and LWP 2 of process 149 is not bound.

```
example% pbind -q 149/1-2
lwp id 149/1: 3
lwp id 149/2: not bound
```

EXAMPLE 5 Querying LWP Bindings for Processor 2:

The following example queries all LWPs bound to processor 2:

```
example% pbind -Q 2
lwp id 149/4: 2
lwp id 149/5: 2
```

Attributes

See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

Exit Status

The following exit values are returned:

`0`
Successful completion.

`>0`
An error occurred.

See Also

psradm(1M), psrinfo(1M), psrset(1M), processor_bind(2), processor_info(2), sysconf(3C), attributes(5)

Diagnostics

`pbind: cannot query pid 31: No such process`
The process specified did not exist or has exited.

`pbind: cannot bind pid 31: Not owner`
The user does not have permission to bind the process.

`pbind: cannot bind pid 31: Invalid argument`
The specified processor is not on-line.

Name pcitool – interrupt routing tool

Synopsis

```
/usr/sbin/pcitool -h

x86:

/usr/sbin/pcitool pci@unit-address -i cpu#,ino# | all [-r [-c] |
     -w cpu# [-g] ] [-v] [-q]

SPARC:

/usr/sbin/pcitool pci@unit-address -i ino# | all [-r [-c] |
     -w cpu# [-g] ] [-v] [-q]

/usr/sbin/pcitool pci@unit-address -m msi# | all [-r [-c] |
     -w cpu# [-g] ] [-v] [-q]
```

Description `PCItool` is a low-level tool that provides a facility for getting and setting interrupt routing information.

Interrupt Routing

The `pcitool -i` command displays device and CPU routing information for INOs on a given nexus, and allows rerouting of a given INO or INO group to a specific CPU.

On SPARC platforms, the INO is mapped to an interrupt mondo, where as one or more MSI/Xs are mapped to an INO. So, INO and MSI/Xs are individually retargetable. Use the `-i` option to retrieve or reroute a given INO; use the `-m` option for MSI/Xs.

Specifying *cpu#* is available on the x86 platform. In combination with *ino#*, this identifies an exclusive vector. The *cpu#* argument is not supported on the SPARC platform.

Required Privileges

A user must have all privileges in order to access interrupt information. A regular user can access interrupt information following an su(1M) to root or if he is granted the "Maintenance and Repair" rights profile in the `user_attr` file. See user_attr(4) and rbac(5).

Options The following options are supported:

`-h`
Display command usage.

`-q`
No errors are displayed as messages. However, `pcitool` still returns Unix error codes.

`-r` [`-c`]
Display device and CPU routing information for INOs on a given nexus. The device path and instance number of each device for each displayed INO is displayed. On some platforms, interrupts dedicated to the root complex are indicated by the string `(Internal)` appended to their pathnames.

With `-c`, dump interrupt controller information.

If neither `-r` nor `-w` are provided on the command line, `-r` is assumed. See `Examples`.

`-v`
Verbose output.

`-w` *cpu#* [`-g`]
Route the given INO or MSI/X to the given CPU. Display the new and original routing information. The INO or MSI/X must be specified.

On some platforms (such as x86) multiple MSI interrupts of a single function need to be rerouted together. Use `-g` to do this. The `-g` option works only on supported platforms and only for groups of MSI interrupts. (A "group" of 1 is accepted.) When `-g` is used, the vector provided must be the lowest-numbered vector of the group. The size of the group is determined internally. See `Examples`.

Examples

EXAMPLE 1 Displaying All INOs

The command for showing all INOs on `/pci@0,0` is:

```
# pcitool /pci@0,0 -i all
```

EXAMPLE 2 Displaying Output for Specific INO

The command for showing INO `0x0,0x21` on the root nexus `/pci@0,0` differs slightly between x86 and SPARC platforms.

On an x86 platform:

```
# pcitool /pci@0,0 -i 0,21
0x0,0x21:                   mpt                         0
/pci@7b,0/pci1022,7458@11/pci1000,3060@2
```

On a SPARC platform:

```
# pcitool /pci@0,0 -i 21
0x0,0x21:                   mpt                         0
/pci@7b,0/pci1022,7458@11/pci1000,3060@2
```

Output shown above is an example and might vary from your output.

EXAMPLE 3 Displaying Output for Specific MSI

The command for showing MSI `0x1` on the root nexus `/pci@0,0`, along with sample output, is shown below.

```
# pcitool /pci@0,0 -m 0x1
0x0,0x1: pcieb     0    /pci@7b,0/pci10de,5d@e
```

EXAMPLE 4 Rerouting an INO from One CPU to Another

Successful rerouting INO 21 from CPU 0 to CPU 1 produces the output shown below.

On an x86 platform:

EXAMPLE 4 Rerouting an INO from One CPU to Another *(Continued)*

```
# pcitool /pci@0,0 -i 0,21 -w 1
0x0,0x21 -> 0x1,0x20
```

On a SPARC platform:

```
# pcitool /pci@0,0 -i 21 -w 1
0x0,0x21 -> 0x1,0x20
```

EXAMPLE 5 Rerouting an MSI from One CPU to Another

Successful rerouting MSI 1 from CPU 1 to CPU 0 produces the output shown below.

```
# pcitool /pci@0,0 -m 1 -w 0
0x1,0x1 -> 0x0,0x1
```

EXAMPLE 6 Rerouting a Group of INOs

Successful rerouting of a group of INOs starting at 24 from CPU 0 to CPU 1 produces the output shown below.

On an x86 platform:

```
# pcitool /pci@0,0 -i 3,24 -w 1 -g
0x3,0x24 => 0x1,0x22
```

On a SPARC platform:

```
# pcitool /pci@0,0 -i 24 -w 1 -g
0x3,0x24 => 0x1,0x22
```

Exit Status

`0`
No error.

`EINVAL`
Out-of-range, misaligned, or otherwise invalid argument has been passed in.

`ETIME`
Timeout waiting for pending interrupt to settle before changing interrupts to a new CPU.

`EIO`
An I/O error occurred.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	PCI-based systems
Availability	system/management/pcitool

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Volatile

See Also su(1M), pci(4), user_attr(4), attributes(5), rbac(5)

PCI specification (available from www.pcisig.org)

Notes All values are entered in hex.

Not all commands are applicable to all platforms.

Name pfedit – per-file authorized edit of administrative files

Synopsis

```
pfedit [-r] file
```

Description The `pfedit` command allows authorized users to edit system configuration files. The *file* argument is a pathname of the file to be edited. If file is not an absolute pathname, the pathname of the current working directory is prepended, and all further processing proceeds as if that were the argument. The invoking user must have the authorization `solaris.admin.edit/`*path_to_file* or the blanket authorization `solaris.admin.edit`. The `pfedit` command allows use of symbolic links, by also checking for authorization for the `realpath(3C)` of file.

The `pfedit` command creates a copy of file owned by the invoking user, then invokes an editor on that file using the id and privileges of the invoking user. The default editor is `/usr/bin/vi`, but can be selected through the use of the `EDITOR` or `VISUAL` environment variable; if both are set, `VISUAL` has precedence. When the user exits the editor and if the copied file has been updated, the updated contents are applied atomically to file. All discretionary access attributes (owner, group, permissions and ACLs) of file are retained, together with any system or extended attributes on the original file. In any case, the user-owned file copy is removed before `pfedit` exits.

If file does not exist the file will be created with owner root, group root, and permissions 644 (`-rw-r--r--`), and then the previously described operations are applied that file. If `pfedit` has been used to create and modify file, the `-r` option can be used to remove file.

The `pfedit` command sets a discretionary lock on file, so that simultaneous updates by means of `pfedit` are prohibited.

The `pfedit` command is careful not to break hard links to other files. Since the atomic update requires replacement of the existing file with a new one with the updated contents, `pfedit` will refuse to operate on a file with a link count greater than one.

The `pfedit` command is restricted to editing text files, and will not accept updates which include non-text characters (NULs).

If configured, in the case of a successful update, an attempt to make unauthorized use, or if an error occurs, an audit record is generated to capture the subject, the file name, the authorization used, the file change if any, and the success or failure of the operation. The audit event type and default class is one of:

```
AUE_admin_edit:edit administrative file:as
AUE_admin_file_create:create administrative file:as
AUE_admin_file_remove:remove administrative file:as
```

Options The following option is supported:

`-r`
: Remove specified file (if file has been created by `pfedit`).

Examples

EXAMPLE 1 Creating a Profile

To create a profile with `solaris.admin.edit` authorization that can be assigned to users to modify `/etc/syslog.conf`, use the `profiles(1)` command.

```
% profiles -p "syslog Configure"
profiles: syslog Configure> set auths=solaris.admin.edit/etc/syslog.conf
profiles: syslog Configure> set desc="Edit syslog configuration"
profiles: syslog Configure> exit
```

EXAMPLE 2 Modifying /etc/syslog.conf

If a user has the "syslog Configure" profile as configured in the previous example then invoking:

```
# pfedit /etc/syslog.conf
```

...creates a copy of `/etc/syslog.conf` owned by that user, and by default invokes `/usr/bin/vi` running as that user on the copy. When the user exits the editor, `/etc/syslog.conf` is atomically updated with the contents saved by the user.

Exit Status

The `pfedit` command has an exit value of 0 if it completes successfully, and a non-zero value if any part of the operation fails.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os
Interface Stability	Uncommitted

See Also

`auths(1)`, `passwd(1)`, `profiles(1)`, `groupadd(1M)`, `groupdel(1M)`, `groupmod(1M)`, `useradd(1M)`, `userdel(1M)`, `usermod(1M)`, `fgetattr(3C)`, `realpath(3C)`, `attributes(5)`, `fsattr(5)`

Notes

Oracle Solaris includes administrative configuration files for which use of `pfedit` and the `solaris.admin.edit/`*path_to_file* authorization is not recommended. Alternate commands exist which are both domain-specific and safer. For example, for the `/etc/passwd`, `/etc/shadow`, or `/etc/user_attr` files, use instead `passwd(1)`, `useradd(1M)`, `userdel(1M)`, or `usermod(1M)`. For the `/etc/group` file, use instead `groupadd(1M)`, `groupdel(1M)`, or `groupmod(1M)`. For updating `/etc/security/auth_attr`, `/etc/security/exec_attr`, or `/etc/security/prof_attr`, the preferred command is `profiles(1)`.

The ability to modify the contents of some configuration files can be used to escalate the privileges assigned to the user. Assignment of an authorization to edit such a file, or of a profile containing such an authorization, should be considered equivalent to providing full privileged access.

Name pginfo – display information about processor groups

Synopsis

```
pginfo [-T] [-p] [-v] [-r string] [-R string]
     [pg ... | -c processor_id ...]

pginfo -s [-v] [-r string] [-R string] [pg ... | -c processor_id ...]

pginfo -c | -I [-r string] [-R string] [pg ... | -c processor_id ...]

pginfo -h
```

Description The `pginfo` displays information about the Processor Group (PG) hierarchy, its contents, and its characteristics. A PG is a set of CPUs that are grouped together by a common characteristic.

PGs are used by the operating system to represent the CPUs that share performance relevant hardware such as the execution pipelines, caches, and so forth. These PGs are organized into a hierarchy that models the processor topology of the machine. In this hierarchy, each CPU (strand) has a leaf PG that represents the CPUs that share the most hardware with it. Each successive ancestor of the leaf PG shares progressively less hardware with the CPU until the root PG is reached. The root PG contains all of the CPUs in the system and represents the group of CPUs sharing the least hardware with each other. (See `EXAMPLES` below for an example of PG hierarchy.)

If a machine does not have any performance-relevant hardware sharing relationships, then `pginfo` displays only a root PG that contains all of the CPUs in the system.

By default, `pginfo` displays information about each PG in the system, including its PG ID, sharing relationship, and online and offline CPUs. It displays the PGs in depth first order by default and uses indentation to help show how the PGs are related to each other (see `EXAMPLES` below).

You can specify options to:

- Display the PG hierarchy graphically
- List the PG sharing relationships that exist on the running system
- Give current PG utilization information, specifying PGs of interest by PG ID, CPU ID, or sharing relationship
- Specify that only CPU or PG IDs be displayed

In addition, there is a `-p` option to show which PGs contain the CPUs that correspond to the CPUs with a common physical relationship such as `system`, `chip`, and `core`. These physical relationships describe the physical characteristics of the CPUs and might or might not encapsulate performance-relevant processor sharing relationships.

If the system configuration repeatedly changes when `pginfo` is obtaining a snapshot of system data, `pginfo` displays an error message and terminates with exit status 1.

Options The following options are supported:

`-c` *processor_id ...*
Interpret arguments as processor IDs and display only information about PGs that contain the specified processors.

When used with the `-T` option, this option limits the PG hierarchy displayed to include only the lineage of each of the specified CPUs. This option cannot be used when specifying PGs of interest by PG ID.

`-C`
Display only CPU IDs for all CPUs belonging to the PGs. This option cannot be used at the same time as the `-I` option.

`-h`
Display short help message and exit with exit status 0.

`-I`
Display only PG IDs for the PGs. This option cannot be used at the same time as the `-C` option.

`-p`
Display the physical relationship that corresponds to a PGs. If a PG has the same CPUs as the whole system, a processor core, or a chip, `system`, `core`, or `chip` will be displayed, as appropriate, after the sharing relationship of the PG in square brackets ("`[]`").

`-r` *string1*,*string2*,...
Display only information about PGs with a sharing relationship name that matches any of the specified strings.

Each specified string can be a whole relationship name or a portion of one or more relationship names and the string matching is case-insensitive. The possible relationship names are in the list of sharing relationships that the `-s` option displays.

You can specify multiple `-r` options, which results in matching all PGs with a relationship name that contain any of the specified strings. When used with the `-T` option, this option limits the PG hierarchy displayed to include only the lineage of each of the PGs with the specified relationship.

`-R` *string1*,*string2*,...
Display only information about PGs with a sharing relationship name *other* than the one(s) specified.

String matching is the same as described above for the `-r` option. Multiple `-R` options can be entered.

`-s`
Display all sharing relationships supported on the running system for the specified PGs. The `-v` option can be used with this option to get the list of PGs for each sharing relationship.

`-T`

In the resulting hierarchy, the lineage of each CPU (hardware strand) is arranged from the PGs that share the most hardware in common with the CPU to the PGs that share the least with the CPU. If any CPUs, PGs, or relationships of interest are specified, the resulting PG hierarchy is limited to the lineages of the PGs with the specified CPUs, PGs, or relationships in the PG hierarchy.

`-v`

Verbose mode. Display additional information about PGs. When used without -s, -C, or -I, it is equivalent to giving the -T and -p options together at the same time. When used with the -s option, it gives the list of PGs for each sharing relationship.

Operands The following operands can be given on the command line by specifying one or more of their corresponding IDs or the keyword `all`. Multiple IDs can be specified as a space-separated list (for example, `1 3`), a range of numbers (for example, `5-8`), or both (for example, `1 3 5-8 13-16`). PGs and CPUs cannot be specified at the same time.

pg PGs of interest can be specified on the command line by PG ID.

processor_id When the `-c` option is entered, CPUs of interest can be specified on the command line by CPU ID.

If an invalid PG or CPU is specified, the `pginfo` command displays a message on standard error showing the invalid ID and continues processing other PGs or CPUs specified on the command line. When none of the specified PGs or CPUs are valid, `pginfo` exits with an exit status of 2.

Examples In the examples below, the system contains one UltraSPARC T1 processor chip with 8 cores and 32 strands.

EXAMPLE 1 Displaying Information About Every PG

The following command, using no arguments, displays information about every PG.

```
$ pginfo
PG  RELATIONSHIP           CPUs
0   System                 0-31
3    Data_Pipe_to_memory   0-31
2     Floating_Point_Unit  0-31
1      Integer_Pipeline    0-3
4      Integer_Pipeline    4-7
5      Integer_Pipeline    8-11
6      Integer_Pipeline    12-15
7      Integer_Pipeline    16-19
8      Integer_Pipeline    20-23
9      Integer_Pipeline    24-27
10     Integer_Pipeline    28-31
```

EXAMPLE 2 Displaying Information About All Sharing Relationships

The following command displays information about all sharing relationships.

```
$ pginfo -s -v
RELATIONSHIP         PGs
-------------------- -------
System               0
Data_Pipe_to_memory  3
Floating_Point_Unit  2
Integer_Pipeline     1 4-10
```

EXAMPLE 3 Displaying PG Hierarchy

The following command displays general information about all PGs in the system. The output shows which PGs belong to chips and cores.

```
$ pginfo -p -T
0 (System) CPUs: 0-31
‘-- 3 (Data_Pipe_to_memory [system,chip]) CPUs: 0-31
    ‘-- 2 (Floating_Point_Unit [system,chip]) CPUs: 0-31
        |-- 1 (Integer_Pipeline [core]) CPUs: 0-3
        |-- 4 (Integer_Pipeline [core]) CPUs: 4-7
        |-- 5 (Integer_Pipeline [core]) CPUs: 8-11
        |-- 6 (Integer_Pipeline [core]) CPUs: 12-15
        |-- 7 (Integer_Pipeline [core]) CPUs: 16-19
        |-- 8 (Integer_Pipeline [core]) CPUs: 20-23
        |-- 9 (Integer_Pipeline [core]) CPUs: 24-27
        ‘-- 10 (Integer_Pipeline [core]) CPUs: 28-31
```

EXAMPLE 4 Displaying List with Specific Criterion

The following command displays a list of CPUs sharing integer pipeline with CPU 0. This example also demonstrates the use of -r option to filter PGs by sharing relationship name.

```
$ pginfo -r integer_pipeline -C -c 0
0 1 2 3
```

EXAMPLE 5 Using Option to Exclude by Specific Criterion

The following command lists all PGs other than the ones that have `Integer_Pipeline` as their relationship.

```
$ pginfo -R Integer_Pipeline
PG RELATIONSHIP           CPUs
0  System                 0-31
3  Data_Pipe_to_memory    0-31
2  Floating_Point_Unit    0-31
```

Exit Status

`0`
Successful completion.

`1`
An error occurred.

`2`
Invalid syntax.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os
Interface Stability	Uncommitted

The command line options and output are Uncommitted.

See Also `pgstat(1M)`, `attributes(5)`

Name pgstat – report utilization statistics for Processor Groups

Synopsis

```
pgstat [-A] [-C] [-p] [-s key | -S key] [-t number] [-T u | d] [-v]
[-r string] [-R string] [-P pg ...] [-c processor_id... ]
[interval [count]]

pgstat -h
```

Description The `pgstat` displays utilization statistics about Processor Groups (PGs). A PG is a set of CPUs that are grouped together by a common characteristic.

PGs are used by the operating system to represent CPUs that share performance relevant hardware, such as execution pipelines, caches, and so forth. These PGs are organized into a hierarchy that models the processor topology of the machine. In this hierarchy, each CPU (strand) has a leaf PG that represents the CPUs that share the most hardware with it. Each successive ancestor of the leaf PG shares progressively less hardware with the CPU until the root PG is reached. The root PG contains all of the CPUs in the system and represents the group of CPUs sharing the least hardware with each other. (See "Examples" below for an example of PG hierarchy).

If a machine does not have any performance-relevant hardware sharing relationships, then `pgstat` displays only a root PG that contains all of the CPUs in the system.

By default, `pgstat` does the following:

- Measures the hardware and software utilization of all PGs in the PG hierarchy over a one second interval.
- Displays the utilization of the PGs in depth first order using indentation to help show how the PGs relate to each other.
- Displays the ID, sharing relationship, hardware load, software load, and online CPUs for each PG at the end of each interval.

The interval and count can be given as arguments to specify the number of seconds in the sampling interval and number of times to measure and display the utilization for the specified PGs.

You can specify options to further tailor the output, organize the output a certain way, and specify PGs of interest (see "Options" below for details).

A hyphen ("-") is displayed when the utilization for a given PG is not supported and a question mark (?) is displayed when the utilization is not available. On systems where the CPU hardware performance counters are needed to measure the hardware utilization, the hardware utilization might be unavailable because the counters are being used by a `cpc(3CPC)` consumer such as `attributes(5)`, `cputrack(1)`, `dtrace(1M)`, or another application that uses `libcpc(3LIB)`.

Options The following options are supported:

`-A`
Display summary of utilization data when `pgstat` is run over multiple intervals.

`-c` *processor_id...*
Display utilization about PGs that contain the specified CPUs. The CPUs can be specified as a comma separated list of CPU IDs. A hyphen ("-") can be used to specify contiguous ranges of CPU IDs (for example, `0-3`).

`-C`
Display utilization of each CPU in each PG.

`-h`
Display short help message and exit with exit status 0.

`-p`
Display the physical relationship that corresponds to a PG. If a PG has the same CPUs as the whole system, a processor core, or a chip, `system`, `core`, or `chip`, as appropriate, is displayed after the sharing relationship of the PG in square brackets ("`[]`").

`-P` *pg,...*
Display utilization for specified PGs. Multiple PGs can be specified as a comma-separated list of PG IDs. A hyphen ("-") can be used to specify a contiguous range of PG IDs (for example, `0-3`).

`-r` *string1*,*string2*,...
Display utilization only for PGs with a sharing relationship name that matches any of the specified strings. The string can be a whole relationship name or a portion of one or more relationship names. The string matching is case-insensitive.

Multiple `-r` options can be entered, which results in matching all PGs with a relationship name that matches any of the specified strings.

`-R` *string1*,*string2*,...
Display information only about PGs with a sharing relationship name *other* than the one(s) specified.

String matching is the same as described above for the `-r` option. Multiple `-R` options can be entered.

`-s` *key*
Sort output lines by the specified key in descending order. The specified key can be one of the following:

`pg`
Sort by PG ID.

`hwload`
Sort by hardware utilization.

`swload`
Sort by software utilization.

`user`
Sort by user time.

`sys`
Sort by system time.

`idle`
Sort by idle time.

`depth`
Sort by descending PG tree from root to leaves, depth-first (default).

`breadth`
Sort by descending PG tree from root to leaves, breadth-first.

`-S` *key*
Sort output lines by the specified key in ascending order. Possible key values are the same as for the `-s` option.

`-t` *number*
Show the top number of PGs for the specified integer number.

`-T u | d`
Display timestamp for each sampling interval in Unix time (see `time(2)`) or the standard date format used by `date(1)`.

`-v`
Display extra information about each PG including hardware utilization and capacity and software user, system, and idle times.

Output Headings pgstat displays the column headings, which are listed below, along with the meanings of those headings.

`PG`
Processor Group ID.

`RELATIONSHIP`
Sharing relationship for PG.

`HW`
Hardware load in percent (calculated as `UTIL/CAP` for interval).

`UTIL`
Hardware utilization of PG's shared hardware component over the interval. This can be a large number, so `K`, `M`, `B`, and `T` are used for denoting thousand, million, billion, and trillion, respectively.

CAP
: Approximate maximum possible utilization for PG's shared hardware component over the interval. This can be a large number, so K, M, B, and T are used for denoting thousand, million, billion, and trillion, respectively.

SW
: Software load in percent (calculated as (USR + SYS) / (USR + SYS + IDLE))

USR
: Percentage of time that software threads ran in user mode on CPUs in PG during interval.

SYS
: Percentage of time that software threads ran in system mode on CPUs in PG during interval.

IDLE
: Percentage of time that no software threads ran on CPUs in PG during interval.

CPUS
: CPU IDs for CPUs in PG.

Examples

In the following examples, the system contains one UltraSPARC T1 processor chip with 8 cores and 32 strands.

EXAMPLE 1 Displaying Utilization for Specified Period

The following command displays utilization for all PGs over the last two seconds.

```
$ pgstat 1 2
PG  RELATIONSHIP            HW    SW  CPUS
 0  System                   -  0.4%  0-31
 3   Data_Pipe_to_memory     -  0.4%  0-31
 2    Floating_Point_Unit   0%  0.4%  0-31
 1     Integer_Pipeline     0%    0%  0-3
 4     Integer_Pipeline     0%    0%  4-7
 5     Integer_Pipeline     0%    0%  8-11
 6     Integer_Pipeline     0%  0.2%  12-15
 7     Integer_Pipeline     0%    0%  16-19
 8     Integer_Pipeline   2.8%  2.7%  20-23
 9     Integer_Pipeline   0.1%  0.2%  24-27
10     Integer_Pipeline     0%    0%  28-31
PG  RELATIONSHIP            HW    SW  CPUS
 0  System                   -  0.4%  0-31
 3   Data_Pipe_to_memory     -  0.4%  0-31
 2    Floating_Point_Unit   0%  0.4%  0-31
 1     Integer_Pipeline     0%  0.2%  0-3
 4     Integer_Pipeline     0%    0%  4-7
 5     Integer_Pipeline     0%    0%  8-11
 6     Integer_Pipeline     0%    0%  12-15
 7     Integer_Pipeline     0%    0%  16-19
```

EXAMPLE 1 Displaying Utilization for Specified Period *(Continued)*

```
 8     Integer_Pipeline   3.1%  2.5%  20-23
 9     Integer_Pipeline     0%    0%  24-27
10     Integer_Pipeline     0%  0.2%  28-31
```

EXAMPLE 2 Displaying Information about Integer Pipeline

The following command displays detailed information about the two most utilized integer pipelines over the last two seconds.

```
$ pgstat -v -t 2 -r 'Integer_Pipeline' 1 2
PG  RELATIONSHIP         HW UTIL  CAP     SW    USR    SYS   IDLE CPUS
 1  Integer_Pipeline   0.2% 2.2M 1.4B   0.2%   0.0%   0.2%  99.8% 0-3
 4  Integer_Pipeline  13.1% 181M 1.4B  14.9%   0.0%  14.9%  85.1% 4-7
PG  RELATIONSHIP         HW UTIL  CAP     SW    USR    SYS   IDLE CPUS
 1  Integer_Pipeline   0.2% 1.9M 1.2B   0.2%   0.0%   0.2%  99.8% 0-3
 4  Integer_Pipeline  13.1% 163M 1.2B  14.9%   0.0%  14.9%  85.1% 4-7
```

EXAMPLE 3 Displaying Core Utilization over Specified Period

The following command displays information about core utilization over the last two minutes.

```
$ pgstat -A 60 2
PG  RELATIONSHIP              HW      SW  CPUS
 0  System                     -   56.9%  0-31
 3   Data_Pipe_to_memory       -   56.9%  0-31
 2    Floating_Point_Unit   0.0%   56.9%  0-31
 1     Integer_Pipeline    36.7%   58.7%  0-3
 4     Integer_Pipeline    41.9%   58.3%  4-7
 5     Integer_Pipeline    31.0%   58.0%  8-11
 6     Integer_Pipeline    30.7%   57.9%  12-15
 7     Integer_Pipeline    30.1%   55.8%  16-19
 8     Integer_Pipeline    40.2%   54.8%  20-23
 9     Integer_Pipeline    35.0%   56.0%  24-27
10     Integer_Pipeline    40.3%   55.8%  28-31
PG  RELATIONSHIP              HW      SW  CPUS
 0  System                     -   10.7%  0-31
 3   Data_Pipe_to_memory       -   10.7%  0-31
 2    Floating_Point_Unit   0.0%   10.7%  0-31
 1     Integer_Pipeline     9.0%   10.7%  0-3
 4     Integer_Pipeline     9.6%   10.8%  4-7
 5     Integer_Pipeline     8.6%    9.9%  8-11
 6     Integer_Pipeline    10.5%   11.9%  12-15
 7     Integer_Pipeline     9.1%   10.4%  16-19
 8     Integer_Pipeline     9.6%   10.9%  20-23
 9     Integer_Pipeline     8.9%   10.0%  24-27
10     Integer_Pipeline     9.5%   10.7%  28-31
```

EXAMPLE 3 Displaying Core Utilization over Specified Period *(Continued)*

```
       SUMMARY: UTILIZATION OVER 120 SECONDS

                                ------HARDWARE------ ------SOFTWARE------
PG  RELATIONSHIP              MIN    AVG    MAX    MIN    AVG    MAX CPUS
 0  System                     -      -      -   10.7%  10.7%  56.9% 0-31
 3   Data_Pipe_to_memory       -      -      -   10.7%  10.7%  56.9% 0-31
 2    Floating_Point_Unit   0.0%   0.0%   0.0%  10.7%  10.7%  56.9% 0-31
 1     Integer_Pipeline     9.0%   8.5%  36.7%  10.7%  10.7%  58.7% 0-3
 4     Integer_Pipeline     9.6%   9.1%  41.9%  10.8%  10.8%  58.3% 4-7
 5     Integer_Pipeline     8.6%   8.1%  31.0%   9.9%   9.9%  58.0% 8-11
 6     Integer_Pipeline    10.5%  10.0%  30.7%  11.9%  11.9%  57.9% 12-15
 7     Integer_Pipeline     9.1%   8.6%  30.1%  10.4%  10.4%  55.8% 16-19
 8     Integer_Pipeline     9.6%   9.1%  40.2%  10.9%  10.9%  54.8% 20-23
 9     Integer_Pipeline     8.9%   8.4%  35.0%  10.0%  10.0%  56.0% 24-27
10     Integer_Pipeline     9.5%   8.9%  40.3%  10.7%  10.7%  55.8% 28-31
```

Exit Status

The following exit values are returned:

`0`
: Successful completion.

`1`
: Unable to get PG information from the system.

`2`
: Specified interval, count, or all CPUs, PGs, and sharing relationships invalid.

`3`
: Invalid syntax.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os
Interface Stability	Uncommitted

The command line options and output are Uncommitted.

See Also

`cputrack(1)`, `attributes(5)`, `dtrace(1M)`, `pginfo(1M)`, `cpc(3CPC)`, `libcpc(3LIB)`, `attributes(5)`

Name picld – PICL daemon

Synopsis `/usr/lib/picl/picld`

Description The Platform Information and Control Library (PICL) provides a mechanism to publish platform-specific information for clients to access in a platform-independent way. `picld` maintains and controls access to the PICL information from clients and plug-in modules. The daemon is started in both single-user and multi-user boot mode.

Upon startup, the PICL daemon loads and initializes the plug-in modules. These modules use the `libpicltree(3PICLTREE)` interface to create nodes and properties in the PICL tree to publish platform configuration information. After the plug-in modules are initialized, the daemon opens the PICL daemon door to service client requests to access information in the PICL tree.

PICL Tree The PICL tree is the repository of all the nodes and properties created by the plug-in modules to represent the platform configuration. Every node in the PICL tree is an instance of a well-defined PICL class. The name of the base PICL class is `picl`, which defines a basic set of properties that all nodes in the tree must possess. Two of those properties are `name` and `_class`, where `name` contains the name of the node, and the `_class` contains the PICL class name of the node. Certain nodes in the PICL tree have well-known names. For example, the name of the root node of the PICL tree is `/` and the name of the root node of the sub-tree containing platform device nodes is `platform`.

PICL plug–in Modules The PICL plug-in modules are shared objects that publish platform-specific data in the PICL tree. They are located in well-known directories so that the daemon can locate and load them.

Plug-in modules are located in one of the following plug-in directories depending on the plaform-specific nature of the data that they collect and publish:

```
/usr/platform/‘uname -i‘/lib/picl/plugins
/usr/platform/‘uname -m‘/lib/picl/plugins
```

A plug-in module can specify its dependency on another plug-in module using the `-l` or `-R` linker option. The plug-ins are loaded by the daemon using `dlopen(3C)` according to the specified dependencies. Each plug-in module must define a `.init` section, which is executed when the plug-in module is loaded, to register themselves with the daemon. See `picld_plugin_register(3PICLTREE)` for additional information on plug-in registration.

The plug-in modules use the `libpicltree(3PICLTREE)` interface to publish nodes and properties in the PICL tree so that clients can access them.

When the PICL daemon invokes the initialization routine of the plug-in module, the plug-in collects the platform information and creates nodes and/or properties to represent the configuration in the PICL tree. A plug-in can create additional threads to monitor the platform configuration and update the PICL tree with any changes. This enables a PICL plug-in to operate as a daemon within the PICL framework.

An environmental monitor is an example of a plug-in module that uses a thread to monitor the temperatures and fan speeds of the platform, then publishes the environmental information in the PICL tree so clients can access them.

Clients use the libpicl(3PICL) interface to send requests to `picld` for accessing the PICL tree.

Exit Status `picld` does not return an exit status.

Files

`/var/run/picld_door`	`PICL` daemon door
`/usr/lib/picl/picld`	`PICL` daemon

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/picl

See Also svcs(1), svcadm(1M), dlopen(3C), libpicl(3PICL), libpicltree(3PICLTREE), picld_log(3PICLTREE), picld_plugin_register(3PICLTREE), attributes(5), smf(5)

Notes The `picld` service is managed by the service management facility, smf(5), under the service identifier:

```
svc:/system/picl
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using svcadm(1M). The service's status can be queried using the svcs(1) command.

Name ping – send ICMP (ICMP6) ECHO_REQUEST packets to network hosts

Synopsis

```
/usr/sbin/ping host [timeout]

/usr/sbin/ping -s [-l | -U] [-abdlLnrRv] [-A addr_family]
     [-c traffic_class] [-g gateway [-g gateway...]]
     [-N next_hop_router] [-F flow_label] [-I interval]
     [-i interface] [-P tos] [-p port] [-t ttl] host
     [data_size] [npackets]
```

Description The utility `ping` utilizes the ICMP (ICMP6 in IPv6) protocol's `ECHO_REQUEST` datagram to elicit an ICMP (ICMP6) `ECHO_RESPONSE` from the specified *host* or network *gateway*. If *host* responds, `ping` will display:

host `is alive`

...on the standard output and exit. Otherwise, after *timeout* seconds, it will write:

`no answer from` *host*

The default value of *timeout* is `20` seconds.

When you specify the `s` flag, sends one datagram per second (adjust with `-I`) and prints one line of output for every `ECHO_RESPONSE` that it receives. `ping` produces no output if there is no response. In this second form, `ping` computes round trip times and packet loss statistics; it displays a summary of this information upon termination or timeout. The default *data_size* is `56` bytes, or you can specify a size with the *data_size* command-line argument. If you specify the optional *npackets*, `ping` sends `ping` requests until it either sends *npackets* requests or receives *npackets* replies.

When using `ping` for fault isolation, first `ping` the local host to verify that the local network interface is running.

Options The following options are supported:

`-A` *addr_family*

Specify the address family of the target host. *addr_family* can be either `inet` or `inet6`. Address family determines which protocol to use. For an argument of `inet`, IPv4 is used. For `inet6`, IPv6 is used.

By default, if the name of a host is provided, not the literal IP address, and a valid IPv6 address exists in the name service database, `ping` will use this address. Otherwise, if the name service database contains an IPv4 address, it will try the IPv4 address.

Specify the address family `inet` or `inet6` to override the default behavior. If the argument specified is `inet`, `ping` will use the IPv4 address associated with the host name. If none exists, `ping` will state that the host is unknown and exit. It does not try to determine if an IPv6 address exists in the name service database.

If the specified argument is `inet6`, `ping` uses the IPv6 address that is associated with the host name. If none exists, `ping` states that the host is unknown and exits.

`-F` *flow_label*
Specify the flow label of probe packets. The value must be an integer in the range from 0 to 1048575. This option is valid only on IPv6.

`-D`
Turn off fragmentation. For IPv4, this means setting the Don't Fragment bit. For IPv4 and IPv6, this means do not allow fragmentation as the datagrams are sent. If the *data_size* exceeds the MTU, then `ping` might report that sending failed due to Message too long.

`-I` *interval*
Turn on the statistics mode and specify the interval between successive transmissions. The default is one second. See the discussion of the `-s` option.

`-L`
Turn off loopback of multicast packets. Normally, members are in the host group on the outgoing interface, a copy of the multicast packets will be delivered to the local machine.

`-N` *next_hop_router*
Specify a next-hop router so that the probe packet goes through the specified router along its path to the target host. This option essentially bypasses the system routing table and leaves the probe packet header unmodified. Only one next-hop router can be specified.

`-P` *tos*
Set the type of service (*tos*) in probe packets to the specified value. The default is zero. The value must be an integer in the range from 0 to 255. Gateways also in the path can route the probe packet differently, depending upon the value of *tos* that is set in the probe packet. This option is valid only on IPv4.

`-R`
Record route. Sets the IPv4 record route option, which stores the route of the packet inside the IPv4 header. The contents of the record route are only printed if the `-v` and `-s` options are given. They are only set on return packets if the target host preserves the record route option across echos, or the `-l` option is given. This option is valid only on IPv4.

`-U`
Send UDP packets instead of ICMP (ICMP6) packets. `ping` sends UDP packets to consecutive ports expecting to receive back ICMP (ICMP6) `PORT_UNREACHABLE` from the target host.

`-a`
`ping` all addresses, both IPv4 and IPv6, of the multihomed destination. The output appears as if `ping` has been run once for each IP address of the destination. If this option is used together with `-A`, `ping` probes only the addresses that are of the specified address family. When used with the `-s` option and *npackets* is not specified, `ping` continuously probes the destination addresses in a round robin fashion. If *npackets* is specified, `ping` sends *npackets* number of probes to each IP address of the destination and then exits.

`-b`
Bypass the global IPsec policy and send and receive packets in the clear for this connection only. This option can be used to troubleshoot network connectivity independent of IPsec. Because this option bypasses system-wide policy for this connection, it can only be used by superuser or a user granted the sys_net_config privilege.

`-c` *traffic_class*
Specify the traffic class of probe packets. The value must be an integer in the range from 0 to 255. Gateways along the path can route the probe packet differently, depending upon the value of *traffic_class* set in the probe packet. This option is valid only on IPv6.

`-d`
Set the `SO_DEBUG` socket option.

`-g` *gateway*
Specify a loose source route gateway so that the probe packet goes through the specified host along the path to the target host. The maximum number of gateways is 8 for IPv4 and 127 for IPv6. Note that some factors such as the link MTU can further limit the number of gateways for IPv6.

`-i` *interface_address*
Specify the outgoing interface address to use for multicast packets for IPv4 and both multicast and unicast packets for IPv6. The default interface address for multicast packets is determined from the (unicast) routing tables. *interface_address* can be a literal IP address, for example, `10.123.100.99`, or an interface name, for example, `eri0`, or an interface index, for example `2`.

`-l`
Use to send the probe packet to the given host and back again using loose source routing. Usually specified with the `-R` option. If any gateways are specified using `-g`, they are visited twice, both to and from the destination. This option is ignored if the `-U` option is used.

`-n`
Show network addresses as numbers. `ping` normally does a reverse name lookup on the IP addresses it extracts from the packets received. The `-n` option blocks the reverse lookup, so `ping` prints IP addresses instead of host names.

`-p` *port*
Set the base UDP *port* number used in probes. This option is used with the `-U` option. The default base *port* number is 33434. The `ping` utility starts setting the destination port number of UDP packets to this base and increments it by one at each probe.

`-r`
Bypass the normal routing tables and send directly to a host on an attached network. If the host is not on a directly attached network, an error is returned. This option can be used to `ping` a local host through an interface that has been dropped by the router daemon. See `in.routed(1M)`.

`-s`
Send one datagram per second and collect statistics.

`-t` *ttl*
Specify the IPv4 time to live, or IPv6 hop limit, for unicast and multicast packets. The default time to live (hop limit) for unicast packets can be set with the ipadm(1M) `set-prop` subcommand, using the `icmp_ipv4_ttl` property for IPv4 and the `icmp_ipv6_hoplimit` property for IPv6. The default time to live (hop limit) for multicast is one hop. See `EXAMPLES`. For further information, seeipadm(1M).

Note – You might observe property names that begin with "_" (underbar). These properties are private to a protocol and are subject to change or removal. See ipadm(1M).

`-v`
Verbose output. List any ICMP (ICMP6) packets, other than replies from the target host.

Operands

host
The network host

Examples

EXAMPLE 1 Using `ping` With IPv6

This example shows `ping` sending probe packets to all the IPv6 addresses of the host `xyz`, one at a time. It sends an ICMP6 `ECHO_REQUEST` every second until the user interrupts it.

```
istanbul% ping -s -A inet6 -a xyz
PING xyz: 56 data bytes
64 bytes from xyz (4::114:a00:20ff:ab3d:83ed): icmp_seq=0. time=0.479 ms
64 bytes from xyz (fec0::114:a00:20ff:ab3d:83ed): icmp_seq=1. time=0.843 ms
64 bytes from xyz (4::114:a00:20ff:ab3d:83ed): icmp_seq=2. time=0.516 ms
64 bytes from xyz (fec0::114:a00:20ff:ab3d:83ed): icmp_seq=3. time=4.943 ms
64 bytes from xyz (4::114:a00:20ff:ab3d:83ed): icmp_seq=4. time=0.485 ms
64 bytes from xyz (fec0::114:a00:20ff:ab3d:83ed): icmp_seq=5. time=2.201 ms
^C
----xyz PING Statistics----
6 packets transmitted, 6 packets received, 0% packet loss
round-trip (ms)  min/avg/stddev = 0.479/1.583/4.943/1.823
```

EXAMPLE 2 Using `ipadm` to Set Hop Limits

The following commands use ipadm(1M) to set IPv4 and IPv6 hop limits.

```
# ipadm set-prop -p _ipv6_hoplimit=100 icmp
# ipadm set-prop -p _ipv4_ttl=100 icmp
```

Exit Status

The following exit values are returned:

0
Successful operation; the machine is alive.

non-zero
An error has occurred. Either a malformed argument has been specified, or the machine was not alive.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	network/ping

See Also ifconfig(1M), in.routed(1M), ipadm(1M), ndd(1M), netstat(1M), rpcinfo(1M), traceroute(1M), attributes(5), icmp(7P), icmp6(7P)

Name pkg2du – convert driver packages to Driver Update format

Synopsis

```
/usr/bin/pkg2du [-f] [-v] [-d dir] [-o iso] [-l label]
     [-r release] pkg [pkg ...]
```

Description The `/usr/bin/pkg2du` utility takes one or more packages as input and converts them to Driver Update (DU) format. If the `-d` option is specified, the resulting DU directory tree is placed in the directory *dir*. If the `-o` option is specified, a Solaris ISO image of the DU directory tree is written in the file *iso*. The ISO image can be burned onto CD/DVD using `cdrw(1)` or `cdrecord(1)` (not a SunOS man page) and used during Solaris installation.

At least one of the `-d` and `-o` options must be specified. If both are specified, then both an ISO image and a directory tree are generated.

Options The following options are supported:

`-d` *dir*
Directory where the DU directory should be created.

`-o` *iso*
Create a Solaris ISO image of the DU directory.

`-f`
If *dir*`/DU` or *iso* exists, remove it without asking first.

`-l` *label*
Label/volume name of the ISO image (if `-o` option is specified).

`-r` *release*
Solaris release number to use. It takes the form of the return from `uname -r` command, for example, `5.10`. If unspecified, the release number of the currently running Solaris is used.

`-v`
Verbose. Multiple `-v` options increase verbosity.

Operands The following operands are supported:

pkg [*pkg*...]
One or more packages to be converted to DU format.

Examples **EXAMPLE 1** Creating a DU CD/DVD

The following commands create a DU CD or DVD containing packages `SUNWfoo` and `SUNWbar`.

```
# /usr/bin/pkg2du -r 5.10 -o my.iso SUNWfoo SUNWbar
# /usr/bin/cdrw -i my.iso
```

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os
Interface Stability	Committed

See Also cdrw(1), mkbootmedia(1M), attributes(5)

mkisofs(8) (not a SunOS man page)

Name pkgadd – transfer software packages to the system

Synopsis

```
pkgadd [-nv] [-a admin] [-G] [-x proxy]
     [ [-M] -R root_path] [-r response] [-k keystore]
     [-P passwd] [-V fs_file]
     [-d device | -d datastream pkginst | all]
     [pkginst | -Y category [,category]...]

pkgadd -s [-d device | -d datastream pkginst | all]
     [pkginst | -Y category [,category]...]
```

Description `pkgadd` transfers the contents of a software package from the distribution medium or directory to install it onto the system. Used without the `-d` *device* source specifier, `pkgadd` looks in the default spool directory (`/var/spool/pkg`) for the package. Used with the `-s` option, it writes the package to a spool directory instead of installing it.

The `pkgadd` utility requires an amount of temporary space the size of the package that is being installed. `pkgadd` determines which temporary directory to use by checking for the existance of the `$TMPDIR` environment variable. If `$TMPDIR` is not defined, `pkgadd` uses `P_tmpdir` from `stdio.h`. `P_tmpdir` has a default of `/var/tmp/`.

Certain unbundled and third-party packages are no longer entirely compatible with the latest version of `pkgadd`. These packages require user interaction throughout the installation and not just at the very beginning, or require that their request scripts be run as the root user.

To install these older packages (released prior to Solaris 2.4), set the following environment variable: `NONABI_SCRIPTS=TRUE`

As long as this environment variable is set, `pkgadd` permits keyboard interaction throughout the installation and package request scripts are run as `root`.

If you have package request scripts that require running as user `root` (instead of `noaccess` [the default] or user `install`), use the `rscript_alt` parameter in the `admin(4)` file to make an appropriate selection. See `admin(4)`.

Note that, in Solaris 8 and Solaris 9, the default user when running a request script was either `root` or `nobody`, depending on the operating system's patch level. In the current release, the default user is `noaccess`.

Package commands are `largefile(5)`-aware. They handle files larger than 2 GB in the same way they handle smaller files. In their current implementations, `pkgadd`, `pkgtrans(1)` and other package commands can process a datastream of up to 4 GB.

The `-d`, `-Y`, and *pkginst* arguments shown in the SYNOPSIS are described under OPERANDS, following OPTIONS.

Options The supported options are described as follows. The `-d` *device* source specifier is described under OPERANDS, below.

`-a` *admin*
Define an installation administration file, *admin*, to be used in place of the default administration file. The token `none` overrides the use of any *admin* file, and thus forces interaction with the user. Unless a full path name is given, `pkgadd` first looks in the current working directory for the administration file. If the specified administration file is not in the current working directory, `pkgadd` looks in the `/var/sadm/install/admin` directory for the administration file.

`-G`
This option is deprecated.

`-k` *keystore*
Use *keystore* as the location from which to get trusted certificate authority certificates when verifying digital signatures found in packages. If no keystore is specified, then the default keystore locations are searched for valid trusted certificates. See `KEYSTORE LOCATIONS` for more information.

`-M`
Instruct `pkgadd` not to use the `$`*root_path*`/etc/vfstab` file for determining the client's mount points. This option assumes the mount points are correct on the server and it behaves consistently with Solaris 2.5 and earlier releases.

`-n`
Installation occurs in non-interactive mode. Suppress output of the list of installed files. The default mode is interactive.

`-P` *passwd*
Password to use to decrypt keystore specified with `-k`, if required. See `PASS PHRASE ARGUMENTS` for more information about the format of this option's argument.

`-r` *response*
Identify a file or directory which contains output from a previous `pkgask(1M)` session. This file supplies the interaction responses that would be requested by the package in interactive mode. *response* must be a full pathname.

`-R` *root_path*
Define the full path name of a directory to use as the *root_path*. All files, including package system information files, are relocated to a directory tree starting in the specified *root_path*. The *root_path* may be specified when installing to a client from a server (for example, `/export/root/client1`).

Note – The root file system of any non-global zones must not be referenced with the `-R` option. Doing so might damage the global zone's file system, might compromise the security of the global zone, and might damage the non-global zone's file system. See `zones(5)`.

`-s` *spool*
Write the package into the directory *spool* instead of installing it.

`-v`
Trace all of the scripts that get executed by `pkgadd`, located in the *pkginst*`/install` directory. This option is used for debugging the procedural and non-procedural scripts.

`-V` *fs_file*
Specify an alternative *fs_file* to map the client's file systems. For example, used in situations where the $*root_path*`/etc/vfstab` file is non-existent or unreliable.

`-x` *proxy*
Specify a HTTP[S] proxy to use when downloading packages The format of proxy is *host*:*port*, where *host* is the hostname of the HTTP[S] proxy, and *port* is the port number associated with the proxy. This switch overrides all other methods of specifying a proxy. See ENVIRONMENT VARIABLES for more information on alternate methods of specifying a default proxy.

When executed without options or operands, `pkgadd` uses `/var/spool/pkg` (the default spool directory).

Operands The following operands are supported:

Sources By default, pkgadd looks in the `/var/spool/pkg` directory when searching for instances of a package to install or spool. Optionally, the source for the package instances to be installed or spooled can be specified using:

`-d` *device*
`-d` *datastream pkgname*,... | `all`
Install or copy a package from *device*. *device* can be any of the following:

- A full path name to a directory or the identifiers for tape or removable medium (for example, `/var/tmp`).
- A datastream created by `pkgtrans` (see `pkgtrans(1)`).
- A URL pointing to a datastream created by `pkgtrans`. The supported Universal Resource Identifiers (URIs) are `http:` and `https:`.

The second form of the `-d` specifier, above, indicates the syntax you use when specifying a datastream. In this case you must specify either a comma-separated list of package names or the keyword `all`.

Instances By default, `pkgadd` searches the specified source, and presents an interactive menu allowing the user to select which package instances found on the source are to be installed. As an alternative, the package instances to be installed can be specified using:

pkginst
The package instance or list of instances to be installed. The token `all` may be used to refer to all packages available on the source medium. The format *pkginst*`.*` can be used to indicate all instances of a package.

The asterisk character (*) is a special character to some shells and may need to be escaped. In the C-Shell, the asterisk must be surrounded by single quotes (') or preceded by a backslash (\).

-Y *category*[,*category*...]
Install packages based on the value of the CATEGORY parameter stored in the package's pkginfo(4) file. All packages on the source medium whose CATEGORY matches one of the specified categories will be selected for installation or spooling.

Keystore Locations

Package such as pkgadd use a set of trusted certificates to perform signature validation on any signatures found within the packages. If there are no signatures included in the packages then signature validation is skipped. The certificates can come from a variety of locations. If -k *keystore* is specified, and *keystore* is a directory, then *keystore* is assumed to be the base directory of the certificates to be used. If *keystore* is a file, then the file itself is assumed to have all required keys and certificates. When -k is not specified, then /var/sadm/security is used as the base directory.

Within the specified base directory, the store locations to be searched are different based on the application doing the searching and the type of store being searched for. The following directories are searched in the specified order:

1. <*store_dir*>/<*app_name*>/<*store_type*>
2. <*store_dir*>/<*store_type*>

Where <*store_dir*> is the directory specified by -k, <*app_name*> is the name of the application doing the searching, and <*store_type*> is one of keystore (for private keys), certstore (for untrusted public key certificates), or truststore (for trusted certificate authority certificates).

For example, when pkgadd is run with -k /export/certs, then the following locations are successively searched to find the trust store:

1. /export/certs/pkgadd/truststore
2. /export/certs/truststore

This searching order enables administrators to have a single location for most applications, and special certificate locations for certain applications.

Keystore And Certificate Formats

The packaging utilities, such as pkgtrans, require access to a set of keys and certificates in order to sign, and optionally verify, packages.

The keystore files found by following the search pattern specified in KEYSTORE LOCATIONS must each be a self-contained PKCS#12-format file.

When signing a package with pkgtrans, if a certstore has more than one public key certificate, then each public key must have a friendlyName attribute in order to be identifiable and selectable with the -a option when signing packages. In addition, the public key certificate selected with -a and found in the certstore must have an associated private key in the keystore.

Several browsers and utilities can be used to export and import certificates and keys into a PKCS#12 keystore. For example, a trusted certificate can be exported from Mozilla, and then imported into a PKCS#12 keystore for use with `pkgadd` with the OpenSSL Toolkit.

Pass Phrase Arguments

`pkgtrans` and `pkgadd` accept password arguments, typically using `-p` to specify the password. These allow the password to be obtained from a variety of sources. Both of these options take a single argument whose format is described below. If no password argument is given and a password is required then the user is prompted to enter one: this will typically be read from the current terminal with echoing turned off.

`pass:`*password*
: The actual password is *password*. Because the password is visible to utilities such as `ps` this form should only be used where security is not important.

`env:`*var*
: Obtain the password from the environment variable *var*. Because the environment of other processes is visible on certain platforms this option should be used with caution.

`file:`*pathname*
: The first line contained within *pathname* is the password. *pathname* need not refer to a regular file: it could, for example, refer to a device or named pipe. For example, to read the password from standard input, use `file:/dev/stdin`.

`console`
: Read the password from `/dev/tty`.

Examples

EXAMPLE 1 Installing a Package from a Solaris DVD

The following example installs a package from a Solaris DVD. You are prompted for the name of the package you want to install.

```
example# pkgadd -d /cdrom/cdrom0/s0/Solaris_10/Product
```

EXAMPLE 2 Installing a Set of Packages from a Datastream

The example command shown below installs all of the packages in the datastream specified by the `-d` source specifier. Prior to this command, this datastream must have been created with the `pkgtrans(1)` command.

```
example# pkgadd -d /var/tmp/datastream all
```

The keyword `all` specifies that all of the packages found in the designated datastream will be installed.

Exit Status

`0`
: Successful completion

`1`
: Fatal error.

2
Warning.

3
Interruption.

4
Administration.

5
Administration. Interaction is required. Do not use `pkgadd -n`.

10
Reboot after installation of all packages.

20
Reboot after installation of this package.

Environment Variables

HTTPPROXY
Specifies an HTTP proxy host. Overrides administration file setting, and `http_proxy` environment variable.

HTTPPROXYPORT
Specifies the port to use when contacting the host specified by HTTPPROXY. Ignored if HTTPPROXY is not set.

`http_proxy`
URL format for specifying proxy host and port. Overrides administration file setting.

Files

`/var/sadm/install/logs/`
Location where `pkgadd` logs an instance of software installation.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	package/svr4
Interface Stability	Committed

See Also

`pkginfo(1)`, `pkgmk(1)`, `pkgparam(1)`, `pkgproto(1)`, `pkgtrans(1)`, `installf(1M)`, `pkgadm(1M)`, `pkgask(1M)`, `pkgchk(1M)`, `pkgrm(1M)`, `removef(1M)`, `admin(4)`, `pkginfo(4)`, `attributes(5)`, `largefile(5)`, `zones(5)`

Application Packaging Developer's Guide

`http://www.openssl.org`

Notes When transferring a package to a spool directory, the `-r`, `-n`, and `-a` options cannot be used.

The `-r` option can be used to indicate a directory name as well as a filename. The directory can contain numerous response files, each sharing the name of the package with which it should be associated. This would be used, for example, when adding multiple interactive packages with one invocation of `pkgadd`. In this situation, each package would need a response file. If you create response files with the same name as the package (for example, `pkinst1` and `pkinst2`), then name the directory in which these files reside after the `-r`.

The `-n` option causes the installation to halt if any interaction is needed to complete it.

If the default *admin* file is too restrictive, the administration file may need to be modified to allow for total non-interaction during a package installation. See `admin(4)` for details.

If a package stream is specified with `-d`, and a digital signature is found in that stream, the default behavior is to attempt to validate the certificate and signature found. This behavior can be overridden with `admin` file settings. See `admin(4)` for more information.

Name pkgadm – manage packaging and patching system

Synopsis

```
pkgadm addcert [-ty] [-a app] [-k keystore] [-e keyfile]
     [-f format] [-n name] [-P passarg]
     [-p import_passarg] [-R rootpath] certfile

pkgadm removecert [-a app] [-k keystore] -n name
     [-P passarg] [-R rootpath]

pkgadm listcert [-a app] [-f format] [-k keystore] -n name
     [-P passarg] [-o outfile] [-R rootpath]

pkgadm dbstatus [-R rootpath]

pkgadm sync [-R rootpath] [-q]

pkgadm -V

pkgadm -?
```

Description The `pkgadm` utility is used for managing the packaging and patching system. It has several subcommands that perform various operations relating to packaging. The `pkgadm` command includes subcommands for managing certificates and keys used.

Managing Keys and Certificates

`pkgadm` maintains the packaging-system-wide keystore in `/var/sadm/security`, and individual user's certificates in `~/.pkg/security`. The following subcommands operate on the package keystore database:

`addcert`
: Add (import) a certificate into the database, with optional trust. Once added, trusted certificates can be used to verify signed packages and patches. Non-trusted user certificates and their associated keys can be used to sign packages and patches. Added user certificates are *not* used to build certificate chains during certificate verification.

`removecert`
: Removes a user certificate/private key pair, or a trusted certificate authority certificate from the keystore. Once removed, the certificate and keys cannot be used.

`listcert`
: Print details of one or more certificates in the keystore.

`sync`
: Writes the contents file and rolls the contents log file. With use of the `-q` option, forces the contents file server to quit.

Internal Install Database

The Solaris operating system relies upon enhanced System V revision 4 (SVr4) packages as the basis for its software installation and revision management. The package maintenance software stores information about installed packages in an internal database. The `pkgadm` subcomand `dbstatus` is used to determine how the package internal database is implemented. The `dbstatus` command returns a string that indicates the type of internal database in use. In

the current implementation, the `dbstatus` command always returns the string `text`, which indicates that the `contents(4)` package database is inuse. Future releases of Solaris might supply alternative database implementations.

Options The following options are supported:

`-a` *app*
If this option is used, then the command only affects the keystore associated with a particular application. Otherwise, the global keystore is affected.

`-e` *keyfile*
When adding a non-trusted certificate/key combination, this option can be used to specify the file that contains the private key. If this option is not used, the private key must be in the same file as the certificate being added.

`-f` *format*
When adding certificates, this specifies the format to expect certificates and private keys in. Possible values when adding are:

`pem`
Certificate and any private key uses PEM encoding.

`der`
Certificate and any private key uses DER encoding.

When printing certificates, this specifies the output format used when printing. Acceptable values for format are:

`pem`
Output each certificate using PEM encoding.

`der`
Output each certificate using DER encoding.

`text`
Output each certificate in human-readable format.

`-k` *keystore*
Overrides the default location used when accessing the keystore.

`-n` *name*
Identifies the entity in the store on which you want to operate. When adding a user certificate, or removing certificates, this name is required. The name is associated with the certificate/key combination, and when adding, can be used later to reference the entity. When printing certificates, if no alias is supplied, then all keystore entities are printed.

`-o` *outfile*
Output the result of the command to *outfile*. Only used when examining (printing) certificates from the key store. Standard out is the default.

`-P` *passarg*
Password retrieval method to use to decrypt keystore specified with `-k`, if required. See `PASS PHRASE ARGUMENTS` in `pkgadd(1M)` for more information about the format of this option's argument. `console` is the default.

`-p` *import_passarg*
This option's argument is identical to `-P`, but is used for supplying the password used to decrypt the certificate and/or private key being added. `console` is the default.

`-q`
(Applies to `sync` subcommand.) Shuts down the contents file cache daemon.

`-R` *rootpath*
Defines the full name of a directory to use as the root (`/`) path. The default user location of the certificate operations is `${HOME}/.pkg`. If the `-R` option is supplied, the certificates and keys will be stored under <*altroot*>`/var/sadm/security`. Note that this operation fails if the user does not have sufficient permissions to access this directory. The `listcert` command requires read permission, while `addcert` and `removecert` require both read and write permission.

Note – The root file system of any non-global zones must not be referenced with the `-R` option. Doing so might damage the global zone's file system, might compromise the security of the global zone, and might damage the non-global zone's file system. See `zones(5)`.

`-t`
Indicates the certificate being added is a trusted CA certificate. The details of the certificate (including the Subject Name, Validity Dates, and Fingerprints) are printed and the user is asked to verify the data. This verification step can be skipped with `-y`. When importing a trusted certificate, a private key should not be supplied, and will be rejected if supplied. Once a certificate is trusted, it can be used as a trust anchor when verifying future untrusted certificates.

`-V`
Print version associated with packaging tools.

`-y`
When adding a trusted certificate, the details of the certificate (Subject name, Issuer name, Validity dates, Fingerprints) are shown to the user and the user is asked to verify the correctness before proceeding. With `-y`, this additional verification step is skipped.

`-?`
Print help message.

Operands The following operand is supported:

`certfile`
File containing the certificate and optional private key, used when adding a trust anchor or certificate/key combination. Certificates must be encoded using PEM or binary DER.

Keystore Aliases

All keystore entries (user cert/key and trusted certificate entries) are accessed via unique aliases. Aliases are case-sensitive.

An alias is specified when you add an entity to a keystore using the `addcert` or `trustcert` subcommand. If an alias is not supplied for a trust anchor, the trust anchor's Common Name is used as the alias. An alias is required when adding a signing certificate or chain certificate. Subsequent `pkgcert` or other package tool commands must use this same alias to refer to the entity.

Keystore Passwords

See the `pkgadd(1M)` man page for a description of the passwords supplied to the `pkgadm` utility.

Examples

EXAMPLE 1 Adding a Trust Anchor

The following example adds a well-known and trusted certificate to be used when verifying signatures on packages.

```
example% pkgadm addcert -t /tmp/certfile.pem
```

EXAMPLE 2 Adding a Signing Certificate

The following example adds a signing certificate and associated private key, each of which is in a separate file, which can then be used to sign packages.

```
example% pkgadm addcert -a pkgtrans -e /tmp/keyfile.pem \
/tmp/certfile.pem
```

EXAMPLE 3 Printing Certificates

The following example prints all certificates in the root keystore.

```
example% pkgadm listcert
```

Exit Status

`0`
successful completion

`non-zero`
fatal error

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	package/svr4
Interface Stability	Committed

See Also

`pkginfo(1)`, `pkgmk(1)`, `pkgparam(1)`, `pkgproto(1)`, `pkgtrans(1)`, `svcs(1)`, `installf(1M)`, `pkgadd(1M)`, `pkgask(1M)`, `pkgrm(1M)`, `removef(1M)`, `svcadm(1M)`, `admin(4)`, `contents(4)`, `exec_attr(4)`, `pkginfo(4)`, `attributes(5)`, `rbac(5)`, `smf(5)`

Application Packaging Developer's Guide

Notes The service for `pkgadm` is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/system/pkgserv
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. The service's status can be queried using the `svcs(1)` command.

Name pkgask – stores answers to a request script

Synopsis

```
pkgask [-d device] [-R root_path] -r response pkginst...
```

Description `pkgask` allows the administrator to store answers to an interactive package (one with a `request` script, that is, a user-created file that must be named `request`). Invoking this command generates a response file that is then used as input at installation time. The use of this response file prevents any interaction from occurring during installation since the file already contains all of the information the package needs.

Options The following options are supported

`-d` *device* Run the request script for a package on *device*. *device* can be a directory pathname or the identifiers for tape or removable medium (for example, `/var/tmp` and `/dev/dsk/c1d0s0`). The default device is the installation spool directory.

`-R` *root_path* Define the full path name of a directory to use as the *root_path*. All files, including package system information files, are relocated to a directory tree starting in the specified *root_path*.

Note – The root file system of any non-global zones must not be referenced with the `-R` option. Doing so might damage the global zone's file system, might compromise the security of the global zone, and might damage the non-global zone's file system. See `zones(5)`.

`-r` *response* Identify a file or directory which should be created to contain the responses to interaction with the package. The name must be a full pathname. The file, or directory of files, can later be used as input to the `pkgadd(1M)` command.

Operands The following operands are supported:

pkginst Specify the package instance, or list of instances for which `request` scripts will be created. The token `all` may be used to refer to all packages available on the source medium.

Exit Status

`0` Successful completion.

`>0` An error occurred.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also `pkginfo(1)`, `pkgmk(1)`, `pkgparam(1)`, `pkgproto(1)`, `pkgtrans(1)`, `installf(1M)`, `pkgadd(1M)`, `pkgchk(1M)`, `pkgrm(1M)`, `removef(1M)`, `admin(4)`,`attributes(5)`

Application Packaging Developer's Guide

Notes The `-r` option can be used to indicate a directory name as well as a filename. The directory name is used to create numerous response files, each sharing the name of the package with which it should be associated. This would be used, for example, when you will be adding multiple interactive packages with one invocation of `pkgadd(1M)`. Each package would need a response file. To create multiple response files with the same name as the package instance, name the directory in which the files should be created and supply multiple instance names with the `pkgask` command. When installing the packages, you will be able to identify this directory to the `pkgadd(1M)` command.

If the default `admin` file is too restrictive, the administration file may need to be modified to allow for total non-interaction during a package installation. See`admin(4)` for details.

Name pkgchk – check package installation accuracy

Synopsis

```
pkgchk [-l | -acfnqvx] [-i file | -]
     [-p path... | -P partial-path...] [-R root_path]
     [ [-m pkgmap [-e envfile]] | pkginst... | -Y category,category...]

pkgchk -d device [-l | -fv] [-i file | -] [-M] [-p path]...
     [-V fs_file]
     [pkginst... | -Y category[,category...]]
```

Description `pkgchk` checks the accuracy of installed files or, by using the `-l` option, displays information about package files. `pkgchk` checks the integrity of directory structures and files. Discrepancies are written to standard error along with a detailed explanation of the problem.

The first synopsis defined above is used to list or check the contents and/or attributes of objects that are currently installed on the system, or in the indicated `pkgmap`. Package names may be listed on the command line, or by default, the entire contents of a machine will be checked.

The second synopsis is used to list or check the contents of a package which has been spooled on the specified device, but not installed. Note that attributes cannot be checked for spooled packages.

Options The following options are supported:

`-a`
Audit the file attributes only and do not check file contents. Default is to check both.

`-c`
Audit the file contents only and do not check file attributes. Default is to check both.

`-d` *device*
Specify the device on which a spooled package resides. *device* can be a directory path name or the identifiers for tape or removable medium (for example, `/var/tmp`.

`-e` *envfile*
Request that the package information file named as *envfile* be used to resolve parameters noted in the specified `pkgmap` file.

`-f`
Correct file attributes if possible. If used with the `-x` option, this option removes hidden files. When `pkgchk` is invoked with this option, it creates directories, named pipes, links, and special devices if they do not already exist. If the `-d` option calls out an uninstalled package, the `-f` option will only take effect if the package is in directory (not stream) format. All file attributes will be set to agree with the entries in the `pkgmap` file except that setuid, setgid, and sticky bits will not be set in the mode.

`-i` *file* | –
Read a list of path names from *file* or from stdin (–) and compare this list against the installation software database or the indicated `pkgmap` file. Path names that are not contained in *file* or stdin are not checked.

`-l`
List information on the selected files that make up a package. This option is not compatible with the `-a`, `-c`, `-f`, `-g`, and `-v` options.

`-m pkgmap`
Check the package against the package map file, `pkgmap`.

`-M`
Instruct `pkgchk` not to use the `$`*root_path*`/etc/vfstab` file for determining the client's mount points. This option assumes the mount points are correct on the server and it behaves consistently with Solaris 2.5 and earlier releases.

`-n`
Do not check volatile or editable files' contents. This should be used for most post-installation checking.

`-p` *path*
Check the accuracy only of the path name or path names listed. *path* can be one or more path names separated by commas (or by whitespace, if the list is quoted).

To specify a *path* that includes a comma, you must use the `-i` option, described above. See EXAMPLES.

`-P` *partial-path*
Check the accuracy of only the partial path name or path names listed. *partial-path* can be one or more partial path names separated by commas (or by whitespace, if the list is quoted). This option can be used instead of `-p` and is not compatible with the other option. This option matches any path name that contains the string contained in the partial path. See the note about paths that contain commas in the description of `-p`.

`-q`
Quiet mode. Do not give messages about missing files.

`-R` *root_path*
Define the full name of a directory to use as the *root_path*. All files, including package system information files, are relocated to a directory tree starting in the specified *root_path*. The *root_path* may be specified when installing to a client from a server (for example, `/export/root/client1`).

Note – The root file system of any non-global zones must not be referenced with the `-R` option. Doing so might damage the global zone's file system, might compromise the security of the global zone, and might damage the non-global zone's file system. See `zones(5)`.

`-v`
Verbose mode. Files are listed as processed.

`-V` *fs_file*
Specify an alternative *fs_file* to map the client's file systems. For example, used in situations where the `$`*root_path*`/etc/vfstab` file is non-existent or unreliable.

`-x`
Search exclusive directories, looking for files which exist that are not in the installation software database or the indicated `pkgmap` file.

`-Y` *category*
Check packages based on the value of the `CATEGORY` parameter stored in the installed or spooled package's `pkginfo(4)` file.

Operands *pkginst*
The package instance or instances to be checked. The format *pkginst*`.*` can be used to check all instances of a package. The default is to display all information about all installed packages.

The asterisk character (`*`) is a special character to some shells and may need to be escaped. In the C-Shell, an asterisk must be surrounded by single quotes (`'`) or preceded by a backslash (\);

partial-path
A portion of a path, such as a file or directory name.

Examples **EXAMPLE 1** Using `pkgchk` for Displaying Package Installation Information

The following example displays package installation information for `/usr/bin/ls`:

```
example% pkgchk -l -p /usr/bin/ls
```

EXAMPLE 2 Checking on Java Font Properties

The following example displays package installation information for all Java font properties installed on the system.

```
example% pkgchk -l -P font.properties
```

EXAMPLE 3 Specifying a Path That Contains a Comma

Assume you want to specify the path:

```
/platform/SUNW,Netra-T12/lib
```

List this path in a file. Here is one way in which you can do that:

```
example% echo "/platform/SUNW,Netra-T12/lib" > /tmp/p
```

You can then enter:

EXAMPLE 3 Specifying a Path That Contains a Comma *(Continued)*

```
example% pkgchk -i /tmp/p -l
Pathname: /platform/SUNW,Netra-T12/lib
Type: directory
Expected mode: 0755
Expected owner: root
Expected group: bin
Referenced by the following packages:
        system/core-osar
Current status: installed
```

Exit Status

`0`
Successful completion.

`>0`
An error occurred.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also

`pkginfo(1)`, `pkgtrans(1)`, `pkgadd(1M)`, `pkgask(1M)`, `pkgrm(1M)`, `pkginfo(4)`, `attributes(5)`, `largefile(5)`

Application Packaging Developer's Guide

Notes

Package commands are `largefile(5)`-aware. They handle files larger than 2 GB in the same way they handle smaller files. In their current implementations, `pkgadd(1M)`, `pkgtrans(1)` and other package commands can process a datastream of up to 4 GB.

Name pkgcond – determine type and capability of target

Synopsis `/usr/bin/pkgcond [-nv]` *condition*

Description The `pkgcond` command allows you to determine the type of target being operated on (global zone, non-global zone, diskless client, and so forth) and the capabilities available for that type of client (can add a driver, path is writable, and so forth). The `pkgcond` command is intended to be invoked from package and patch scripts, but can also be used in situations that mimic the context of these scripts. See `NOTES` for further guidance.

`pkgcond` has one mandatory argument, a condition. The command tests whether the condition is true for the specified path. The condition can be one of the following:

- `can_add_driver` [*path*]
- `can_remove_driver` [*path*]
- `can_update_driver` [*path*]
- `is_alternative_root` [*path*]
- `is_boot_environment` [*path*]
- `is_diskless_client` [*path*]
- `is_global_zone` [*path*]
- `is_mounted_miniroot` [*path*]
- `is_netinstall_image` [*path*]
- `is_nonglobal_zone` [*path*]
- `is_path_writable` *path*
- `is_running_system` [*path*]
- `is_sparse_root_nonglobal_zone` [*path*]
- `is_what` [*path*]
- `is_whole_root_nonglobal_zone` [*path*]

The *path* argument usually denotes the root of the global zone or non-global zone, or alternate root. If path is optional and not specified, the default is `/`.

The behavior of the `is_what` condition is somewhat special, because it displays results of all other conditions to standard output.

Options The following options are supported:

`-n`
Negate return status (`0` becomes `1` and `1` becomes `0`). It negates results in the case of `is_what` condition.

`-v`
Verbose mode. Displays detailed data about intermediate checks performed.

Examples **EXAMPLE 1** Listing All Available Information

The following command lists all available information about the current running system, in a user-friendly way.

```
example# pkgcond -n is_what
```

EXAMPLE 2 Determining if Target is an Alternate Root

The following command determines whether an alternate boot environment exists under `/altroot_mount`.

```
example# pkgcond is_alternative_root /altroot_mount
```

Exit Status

`0`
Condition is true unless `-n` was specified.

`1`
Condition is false unless `-n` was specified.

`2`
Command line usage error.

`3`
Command failed to perform the test due to a fatal error.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os
Interface Stability	Uncommitted

See Also

`pkgtrans(1)`, `pkgadd(1M)`, `pkgask(1M)`, `pkgchk(1M)`, `pkgrm(1M)`, `pkginfo(4)`, `attributes(5)`

Notes

Supported usage of `pkgcond` is subject to the following constraints:

1. Do not use `pkgcond` outside of the Solaris marketing release in which it is provided (for example, do not use Solaris 10 `pkgcond` against a Solaris 9 target).
2. Restrict use of the optional path argument according to the following rules:
   - The command `pkgcond` *condition* `$ROOTDIR` can be used in patch level scripts.
   - The command `pkgcond` *condition* `$PKG_INSTALL_ROOT` can be used in package level scripts.
   - A command of the form `pkgcond` *condition without* the optional path argument can be used in any context.

Use of `pkgcond` with an arbitrary path argument is *not* recommended or supported, as the results returned might not be accurate.

Name pkg.depotd – Image Packaging System depot server

Synopsis

```
/usr/lib/pkg.depotd [--cfg source] [-a address]
    [--content-root root_dir] [-d inst_root]
    [--debug feature_list] [--disable-ops=op[/1][,...]]
    [--image-root path] [--log-access dest]
    [--log-errors dest] [--mirror mode] [-p port]
    [--proxy-base url] [--readonly mode] [-s threads]
    [--sort-file-max-size bytes] [--ssl-cert-file source]
    [--ssl-dialog type] [--ssl-key-file source]
    [-t socket_timeout] [--writable-root path]
```

Description `pkg.depotd` is the depot server for the Image Packaging System. It provides network access to the data contained within a package repository. Clients that do not support direct access to a repository through the file system, or for which network access is the only available or preferred method of transport, typically use the package depot.

Clients such as `pkg`, the retrieval client, can retrieve a list of packages and package metadata from a repository directly or through the depot server. `pkgsend`, the publication client, can send new versions of packages to a repository directly or through the depot server. `pkgrepo` can be used to create repositories for use with the depot server, or to manage them both directly and through the depot server.

`pkg.depotd` is typically run as a service on the system. Package and software developers might want to run private copies for testing.

The depot does not provide any access control methods of its own. By default, all of the clients that are able to connect are able to read all package data and publish new package versions. The exception is that when running under Service Management Facility (SMF), the default is to run in read-only mode. The "Notes" section below describes some best practices for maintaining a public depot server with evolving content.

Smf Properties The `pkg.depot` server is generally configured via the SMF properties associated with its service. See the `smf`(5) man page for information about SMF properties. The following properties are recognized:

`pkg/address`
(`net_address`) The IP address on which to listen for connections. The default value is 0.0.0.0 (`INADDR_ANY`), which listens on all active interfaces. To listen on all active IPv6 interfaces, use `::`. Only the first value is used.

`pkg/content_root`
(`astring`) The file system path at which the instance should find its static and other web content. The default value is `/usr/share/lib/pkg`.

`pkg/debug`
(`astring`) A comma-separated list of debug features to enable. Possible values are:

`headers` Logs the headers of every request to the error log.

`pkg/disable_ops`
(`astring`) A comma-separated list of operations that should be disabled for the depot server. Operations are given as *operation*[/*version*] (`catalog` or `search_1`, for example).

`pkg/image_root`
(`astring`) The path to the image whose file information will be used as a cache for file data.

`pkg/inst_root`
(`astring`) The file system path at which the instance should find its repository data. Required unless `file_root` or `PKG_REPO` has been provided. The default value is `/var/pkgrepo`.

`pkg/log_access`
(`astring`) The destination for any access related information logged by the depot process. Possible values are: `stderr`, `stdout`, `none`, or an absolute path name. The default value is `stdout` if `stdout` is a `tty`. If `stdout` is not a `tty`, the default value is `none`. If you run `pkg` as a service, the default value for `log_access` is `none` and output is written to `/var/svc/log/application-pkg-server:*`. See the `logadm`(1M) man page for examples of managing large log files.

`pkg/log_errors`
(`astring`) The destination for any errors or other information logged by the depot process. Possible values are: `stderr`, `stdout`, `none`, or an absolute path name. The default value is `stderr`. See the `logadm`(1M) man page for examples of managing large log files.

`pkg/mirror`
(`boolean`) Sets whether package mirror mode is used. When true, publishing and metadata operations are disabled and only a limited browser user interface is provided. This property cannot be true when the `pkg/readonly` property is true. The default value is `false`.

`pkg/port`
(`count`) The port number on which the instance should listen for incoming package requests. If SSL certificate and key information has not been provided, the default value is 80; otherwise, the default value is 443.

`pkg/proxy_base`
(`uri`) This changes the base URL for the depot server and is most useful when running behind Apache or some other web server in a reverse proxy configuration.

`pkg/readonly`
(`boolean`) Sets whether modifying operations, such as those initiated by `pkgsend`, are disabled. Retrieval operations are still available. This property cannot be true when the `pkg/mirror` property is true. The default value is `true`.

`pkg/socket_timeout`
(`count`) The maximum number of seconds the server should wait for a response from a client before closing a connection. The default value is 60.

`pkg/sort_file_max_size`
(`count`) The maximum size of the indexer sort file. Used to limit the amount of RAM the depot uses for indexing, or increase it for speed.

`pkg/ssl_cert_file`
(`astring`) The absolute path name to a PEM-encoded Certificate file. The default value is `none`. This property must be used with `ssl_key_file`. The depot only responds to SSL requests if both `ssl_cert_file` and `/ssl_key_file` are provided.

`pkg/ssl_dialog`
(`astring`) Specifies what method should be used to obtain the passphrase used to decrypt the `ssl_key_file`. Possible values are:

`builtin`
Prompt for the passphrase. This is the default value.

`exec:`*/path/to/program*
Execute the specified external program to obtain the passphrase. The first argument to the program is `''`, and is reserved. The second argument to the program is the port number of the server. The passphrase is printed to `stdout`.

`smf:fmri`
Attempt to retrieve the value of the property `pkg_secure/ssl_key_passphrase` from the service instance related to the FMRI.

`pkg/ssl_key_file`
(`astring`) The absolute path name to a PEM-encoded Private Key file. This property must be used with the property `ssl_cert_file`. The depot only responds to SSL requests if both `/ssl_key_file` and `ssl_cert_file` are provided.

`pkg/threads`
(`count`) The number of threads started to serve requests. The default value is 60. Suitable only for small deployments. This value should be approximately 20 times the number of concurrent clients. The maximum value of `threads` is 5000.

`pkg/writable_root`
(`astring`) The file system path to a directory to which the program has write access. This is used with the `-readonly` option to enable the depot server to create files, such as search indexes, without needing write access to the package information.

`pkg_secure/ssl_key_passphrase`
(`astring`) The password to use to decrypt the `pkg/ssl_key_file`. This value is read-authorization protected using the attribute `solaris.smf.read.pkg-server`.

The presentation and behavior of the Browser User Interface (BUI) of the depot server is controlled using the following properties:

`pkg_bui/feed_description`
(`astring`) A descriptive paragraph for the RSS/Atom feed.

`pkg_bui/feed_icon`
(`astring`) The path name of a small image used to visually represent the RSS/Atom feed. The path name should be relative to the `content_root`. The default value is `web/_themes/pkg-block-icon.png`.

`pkg_bui/feed_logo`
(`astring`) The path name of a large image that will be used to visually brand or identify the RSS/Atom feed. This value should be relative to the `content_root`. The default value is `web/_themes/pkg-block-icon.png`.

`pkg_bui/feed_name`
(`astring`) A short, descriptive name for RSS/Atom feeds generated by the depot serving the repository. The default value is "package repository feed".

`pkg_bui/feed_window`
(`count`) The number of hours before the feed for the repository was last generated, to include when generating the feed.

The package depot is also able to act as a mirror server for local client images from `pkg`(5). This enables clients that share a subnet on a LAN to mirror their file caches. Clients can download files from one another, thereby reducing load on the package depot server. This functionality is available as an alternate depot service configured by SMF. It uses mDNS and `dns-sd` for service discovery.

The mDNS mirror is generally configured via the SMF properties associated with its service. The following properties are recognized:

`pkg/image_root`
(`astring`) The path to the image whose file information will be used as a cache for file data. The default value is `/`.

`pkg/port`
(`count`) The port number on which the instance should listen for incoming package requests. The default value is 80.

Options `pkg.depotd` can read its base configuration information from a file or from the property data of an existing SMF service instance.

`--cfg` *source*
Specify the path name of a file to use when reading and writing configuration data, or a string of the form `smf:`*fmri* where *fmri* is the service fault management resource identifier (FMRI) of the instance to read configuration data from. See "Depot Configuration" below for details on the format of the file specified.

If no preexisting configuration source is available, or to override values read from a configuration file provided using `--cfg`, the following options can be used to alter the default behavior of the depot server:

`-a` *address*
See `pkg/address` above.

`--content-root` *root_dir*
See `pkg/content_root` above.

`-d` *inst_root*
See `pkg/inst_root` above.

`--debug` *feature_list*
See `pkg/debug` above.

`--disable-ops=`*op*[`/1`][,...]
See `pkg/disable_ops` above.

`--image-root` *path*
See `pkg/image_root` above.

`--log-access` *dest*
See `pkg/log_access` above.

`--log-errors` *dest*
See `pkg/log_errors` above.

`--mirror` *mode*
See `pkg/mirror` above.

`-p` *port*
See `pkg/port` above.

`--proxy-base` *url*
See `pkg/proxy_base` above. This option is ignored if an empty value is provided.

`--readonly` *mode*
See `pkg/readonly` above.

`-s` *threads*
See `pkg/threads` above.

`--sort-file-max-size` *bytes*
See `pkg/sort_file_max_size` above.

`--ssl-cert-file` *source*
See `pkg/ssl_cert_file` above.

`--ssl-dialog` *type*
See `pkg/ssl_dialog` above.

`--ssl-key-file` *source*
See `pkg/ssl_key_file` above.

`-t` *socket_timeout*
See `pkg/socket_timeout` above.

`--writable-root` *path*
See `pkg/writable_root` above.

`-?`
`--help`
Display a usage message.

Additional administrative and management functionality for package repositories is provided by `pkgrepo`.

Depot Configuration

When a configuration file is provided (instead of an SMF FMRI) by using the `--cfg` option, the depot server reads and writes all configuration data in a simple text format. The configuration data is described in "SMF Properties" above. The configuration data consists of sections, lead by a `[`*section*`]` header, and followed by `name =` *value* entries. Continuations are in the style of RFC 822. Values can be split over multiple lines by beginning continuation lines with whitespace.

Any required values not provided in the configuration file must be provided using the option listed in "Options" above. A sample configuration file might look like this:

```
[pkg]
port = 80
inst_root = /export/repo

[pub_example_com]
feed_description = example.com's software
  update log
```

Examples

EXAMPLE 1 Enabling the Depot Server

```
# svcadm enable application/pkg/server
```

EXAMPLE 2 Changing the Listening Port of the Server.

```
# svccfg -s application/pkg/server setprop pkg/port = 10000
# svcadm refresh application/pkg/server
# svcadm restart application/pkg/server
```

EXAMPLE 3 Enabling the Mirror

```
# svcadm enable application/pkg/dynamic-mirror
```

Environment Variables

`PKG_REPO` — Specifies the directory that contains the repository to serve. This value is ignored if `-d` is specified.

`PKG_DEPOT_CONTENT` — Specifies the directory that contains static content served by the depot. The files listed below under "Files" should be present in this directory, although their content can differ from the supplied default content.

Exit Status The following exit values are returned:

`0` Successful operation.

`1` An error occurred.

`2` Invalid command line options were specified.

`99` An unanticipated exception occurred.

Files `/usr/share/lib/pkg`
Default presentation content location. Modify `pkg/content_root` to select an alternate location.

Attributes See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	`package/pkg`
Interface Stability	Uncommitted

See Also `dns-sd`(1M), `mdnsd`(1M), `pkg`(1), `pkgrepo`(1), `pkgsend`(1), `syslogd`(1M), `smf`(5)

`http://hub.opensolaris.org/bin/view/Project+pkg/`

Notes The `pkd.depotd` service is managed by SMF under the service identifier `svc:/application/pkg/server`.

The mDNS mirror service is managed by SMF under the service identifier `svc:/application/pkg/dynamic-mirror`.

To control read access to the depot, you can use an HTTP reverse proxy in combination with authentication methods such as client based SSL certificate access, which `pkg` natively supports.

Changes to configuration, or changes to package data using file system based operations, require a restart of the depot server process so that the changes can be reflected in operations and output. Use one of the following methods to restart the depot server process:

- Use `svcadm` to restart the `application/pkg/server` instance.
- Send a `SIGUSR1` signal to the depot server process using `kill`. This executes a "graceful restart" that leaves the process intact but reloads all configuration, package, and search data:

  `# kill -USR1` *pid*

Name pkgrm – remove a package from the system

Synopsis

```
pkgrm [-nv] [-a admin] [ [-A | -M] -R root_path]
     [-V fs_file]
     [pkginst... | -Y category[,category...]]

pkgrm -s spool
     [pkginst... | -Y category[,category...]]
```

Description `pkgrm` will remove a previously installed or partially installed package from the system. A check is made to determine if any other packages depend on the one being removed. If a dependency exists, the action taken is defined in the `admin` file.

The default state for the command is in interactive mode, meaning that prompt messages are given during processing to allow the administrator to confirm the actions being taken. Non-interactive mode can be requested with the `-n` option.

The `-s` option can be used to specify the directory from which spooled packages should be removed.

Certain unbundled and third-party packages are no longer entirely compatible with the latest version of `pkgrm`. These packages require user interaction throughout the removal and not just at the very beginning.

To remove these older packages (released prior to Solaris 2.4), set the following environment variable:`NONABI_SCRIPTS=TRUE` `pkgrm` permits keyboard interaction throughout the removal as long as this environment variable is set.

Options The following options are supported:

`-a` *admin*
: Use the installation administration file, `admin`, in place of the default `admin` file. `pkgrm` first looks in the current working directory for the administration file. If the specified administration file is not in the current working directory, `pkgrm` looks in the `/var/sadm/install/admin` directory for the administration file.

`-A`
: Remove the package files from the client's file system, absolutely. If a file is shared with other packages, the default behavior is to not remove the file from the client's file system.

`-M`
: Instruct `pkgrm` not to use the $*root_path*`/etc/vfstab` file for determining the client's mount points. This option assumes the mount points are correct on the server and it behaves consistently with Solaris 2.5 and earlier releases.

`-n`
: Non-interactive mode. If there is a need for interaction, the command will exit.

Use of this option requires that at least one package instance be named upon invocation of the command. Certain conditions must exist for a package to be removed non-interactively or a non-restrictive `admin` file needs to be used.

`-R` *root_path*
Defines the full path name of a directory to use as the *root_path*. All files, including package system information files, are relocated to a directory tree starting in the specified *root_path*.

Note – The root file system of any non-global zones must not be referenced with the `-R` option. Doing so might damage the global zone's file system, might compromise the security of the global zone, and might damage the non-global zone's file system. See `zones(5)`.

`-s` *spool*
Remove the specified package(s) from the directory *spool*. The default directory for spooled packages is `/var/sadm/pkg`.

`-v`
Trace all of the scripts that get executed by `pkgrm`, located in the *pkginst*`/install` directory. This option is used for debugging the procedural and non-procedural scripts.

`-V` *fs_file*
Specify an alternative *fs_file* to map the client's file systems. Used in situations where the $*root_path*`/etc/vfstab` file is non-existent or unreliable.

`-Y` *category*
Remove packages based on the value of the `CATEGORY` parameter stored in the installed or spooled package's `pkginfo(4)` file. No package with the `CATEGORY` value of `system` can removed from the file system with this option.

Operands The following operand is supported:

pkginst
Specifies the package to be removed. The format *pkginst*`.*` can be used to remove all instances of a package.

The asterisk character (`*`) is a special character to some shells and may need to be escaped. In the C-Shell, "*" must be surrounded by single quotes (`'`) or preceded by a backslash (\).

Examples **EXAMPLE 1** Removing All Instances of SUNWjunk from client1

The following example removes all instances of `SUNWjunk` from `client1`:

```
example% pkgrm -R /export/root/client1 SUNWjunk*
```

Note the caveat on the use of the `-R` option in the description of that option, above.

Exit Status The following exit values are returned:

`0`
Successful completion.

`1`
Fatal error.

`2`
Warning.

`3`
Interruption.

`4`
Administration.

`10`
Reboot after removal of all packages.

`20`
Reboot after removal of this package.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also `pkginfo(1)`, `pkgmk(1)`, `pkgparam(1)`, `pkgproto(1)`, `pkgtrans(1)`, `installf(1M)`, `pkgadd(1M)`, `pkgask(1M)`, `pkgchk(1M)`, `removef(1M)`, `admin(4)`, `pkginfo(4)`, `attributes(5)`, `largefile(5)`

Application Packaging Developer's Guide

Notes Package commands are `largefile(5)`-aware. They handle files larger than 2 GB in the same way they handle smaller files. In their current implementations, `pkgadd(1M)`, `pkgtrans(1)` and other package commands can process a datastream of up to 4 GB.

Name pkg.sysrepo – Image Packaging System system repository configuration

Synopsis

```
/usr/lib/pkg.sysrepo -p port [-c cache_dir] [-s cache_size]
    [-w http_proxy] [-W https_proxy]
```

Description `pkg.sysrepo` is used to generate the configuration files for the Image Packaging System (IPS) system repository. `pkg.sysrepo` is called by the `svc:/application/pkg/system-repository` Service Management Facility (SMF) service. Changes in configuration should be made to the properties in the SMF service.

The system repository is responsible for providing access to the package repositories configured in a reference image through a centralized proxy. Publisher configuration changes made to that reference image are seen immediately by any clients configured to use the system repository.

The system repository is primarily used in the global zone to allow non-global zones to access the repositories configured in the global zone. The SMF services `svc:/application/pkg/zones-proxyd` and `svc:/application/pkg/zones-proxy-client` are responsible for providing the transport between non-global zones and the global zone. This transport is only used by `pkg`(5).

Note that only `http`, `https`, and `v4` file repositories and `p5p` archives are supported. Older file repository formats are not supported. See `pkgrepo`(1) for more information about repository versions.

Options The following options are supported:

`-c` *cache_dir*
: Specify the absolute path to a directory that should be used by the system repository for caching responses from the publishers configured.

  By default, a file-cache is used. However, the special value `memory` can be used to indicate the an in-memory cache should be used. The special value `None` can be used to indicate that the system repository should not perform any caching. This setting should be configured using the `config/cache_dir` SMF property.

`-p` *port*
: Specify the port that the system repository should use to listen for requests. This setting should be configured using the `config/port` SMF property.

`-s` *cache_size*
: An integer value in megabytes that defines the maximum cache size of the system repository. This setting should be configured using the `config/cache_max` SMF property.

`-w` *http_proxy*
: A string of the form *scheme*`://`*hostname*`[:`*port*`]` that defines a web proxy that the system repository can use to access http-based package repositories. This setting can be configured using the `config/http_proxy` SMF property.

`-W` *https_proxy*
A string of the form *scheme*`://`*hostname*`[:`*port*`]` that defines a web proxy that the system repository can use to access https-based package repositories. This setting can be configured using the `config/https_proxy` SMF property.

Examples

EXAMPLE 1 Enabling the System Repository

```
$ svcadm enable svc:/application/pkg/system-repository
```

Exit Status

The following exit values are returned:

`0` Command succeeded.

`1` Command failed to write a valid configuration.

`2` Invalid command line options were specified.

`99` An unanticipated exception occurred.

Attributes

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	`package/pkg`
Interface Stability	Uncommitted

See Also

`pkg`(1), `pkg.depotd`(1M), `pkg`(5)

`http://hub.opensolaris.org/bin/view/Project+pkg/`

Name plockstat – report user-level lock statistics

Synopsis

```
plockstat [-vACHV] [-n count] [-s depth] [-e secs]
     [-x arg [=val]] command [arg]...

plockstat [-vACHV] [-n count] [-s depth] [-e secs]
     [-x arg [=val]] -p pid
```

Description The `plockstat` utility gathers and displays user-level locking statistics. By default, `plockstat` monitors all lock contention events, gathers frequency and timing data about those events, and displays the data in decreasing frequency order, so that the most common events appear first.

`plockstat` gathers data until the specified command completes or the process specified with the `-p` option completes.

`plockstat` relies on DTrace to instrument a running process or a command it invokes to trace events of interest. This imposes a small but measurable performance overhead on the processes being observed. Users must have the `dtrace_proc` privilege and have permission to observe a particular process with `plockstat`. Refer to the *Solaris Dynamic Tracing Guide* for more information about DTrace security features.

Options The following options are supported:

`-A`	Watch all lock events. This option is equivalent to `-CH`.
`-C`	Watch contention events.
`-H`	Watch hold events.
`-e` *secs*	Exit after the number of seconds specified have elapsed.
`-n` *count*	Display only the specified number of entries for each output category.
`-s` *depth*	Record a stack trace rather than just the calling function.
`-p` *pid*	Specify a process ID from which `plockstat` is to gather data.
`-v`	Print out a message to indicate that tracing has started.
`-x` *arg*[=*val*]	Enable or modify a DTrace runtime option or D compiler option. The list of options is found in the *Solaris Dynamic Tracing Guide*. Boolean options are enabled by specifying their name. Options with values are set by separating the option name and value with an equals sign (=).
`-V`	Print the Dtrace commands used to gather the data. The output can then be used directly with the `dtrace(1M)` command.

Operands The following operands are supported:

arg	A string to be passed as an argument to *command*.
command	The name of a utility to be invoked.

count A positive integer value.

pid A process identifier for a process to be monitored.

secs Duration specified as a positive integer number of seconds.

Display Headers The following headers appear over columns of data in `plockstat` output.

`Count`
Number of times an event occurred.

`nsec`
Average duration of an event, in nanoseconds.

`Lock`
Address of a lock, displayed symbolically if possible.

`Caller`
Address of a caller, displayed symbolically if possible.

Exit Status The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/dtrace
Interface Stability	See below.

The command-line syntax is Committed. The human-readable output is Uncommitted.

See Also `dtrace(1M)`, `lockstat(1M)`, `mutex_init(3C)`, `pthread_mutex_lock(3C)`, `pthread_rwlock_rdlock(3C)`, `pthread_rwlock_wrlock(3C)`, `pthread_rwlock_unlock(3C)`, `rwlock(3C)`, `attributes(5)`, `fasttrap(7D)`

Solaris Dynamic Tracing Guide

Name pntadm – DHCP network table management utility

Synopsis

```
pntadm -C [-r resource] [-p path] [-u uninterpreted] network

pntadm -A name_IP_address [-c comment] [-e mm/dd/yyyy]
     [-f num | keywords] [-h client_hostname]
     [-i [-a] client_ID] [-m [-y] macro] [-s server]
     [-r resource] [-p path] [-u uninterpreted] network

pntadm -M name_IP_address [-c comment] [-e mm/dd/yyyy]
     [-f num | keywords] [-h client_hostname]
     [-i [-a] client ID] [-m [-y] macro]
     [-n new_client_IP_address] [-s server] [-r resource]
     [-p path] [-u uninterpreted] network

pntadm -D name_IP_address [-y] [-r resource] [-p path]
     [-u uninterpreted] network

pntadm -P [-v] [-x] [-r resource] [-p path]
     [-u uninterpreted] network

pntadm -R [-r resource] [-p path] [-u uninterpreted] network

pntadm -L [-r resource] [-p path] [-u uninterpreted]

pntadm -B [-v] [batchfile]
```

Description The `pntadm` command is used to manage the Dynamic Host Configuration Protocol (DHCP) network tables. It is used to add and remove networks under DHCP management, and add, delete, or modify IP address records within network tables, or to view tables. For a description of the format of DHCP network tables, see `dhcp_network(4)`.

`pntadm` can be run as root or by other users assigned to the DHCP Management profile. See `rbac(5)` and `user_attr(4)`.

If the networks you want to add are subnetted, you need to update the `netmasks(4)` table.

One of the following options (function flags) must be specified with the `pntadm` command: `-A`, `-B`, `-C`, `-D`, `-L`, `-M`, `-P`, or `-R`.

The `pntadm` utility is obsolete and is subject to removal in a future release of Oracle Solaris.

Options The following options are supported:

`-A` *name_IP_address*
Add a client entry with hostname or client IP address, *name_IP_address*, to the named DHCP network table.

The following sub-options are optional:

`-c` *comment*
Comment text. The default is `NULL`.

`-e` *mm/dd/yyyy*
Absolute lease. The default is `0`.

`-f` *num* | *keywords*
Flag value. The default is `00`.

The flag (`-f`) option can be specified either as a single number denoting the intended flag value, or as a series of the following keywords, combined using the plus (+) symbol:

`DYNAMIC` or `00`
Server manager's assignment.

`PERMANENT` or `01`
Lease on entry is permanent.

`MANUAL` or `02`
Administrator managed assignment.

`UNUSABLE` or `04`
Entry is not valid.

`BOOTP` or `08`
Entry reserved for `BOOTP` clients.

For a more detailed description of the flag values, see `dhcp_network(4)`.

`-h` *client_hostname*
Client hostname. The default is NULL.

When the `-h` option is used in this mode, the *client_hostname* is added to the hosts table within the resource used for storing host names (files or DNS). The command will fail if this *client_hostname* is already present in the hosts table.

`-i` *client_ID* [`-a`]
Client identifier [`-a`]. The default is `00`.

The `-i` option modified with `-a` specifies that the client identifier is in ASCII format, and thus needs to be converted to hexadecimal format before insertion into the table.

`-m` *macro* [`-y`]
Macro name. Default is UNKNOWN.

The `-m` option modified with `-y` verifies the existence of the named macro in the `dhcptab` table before adding the entry.

`-s` *server*
Server IP or name. Default is system name (`uname -n`).

`-B`
Activate batch mode. `pntadm` will read from the specified file or from standard input a series of `pntadm` commands and execute them within the same process. Processing many

`pntadm` commands using this method is much faster than running an executable batchfile itself. Batch mode is recommended for using `pntadm` in scripts.

The following sub-option is optional:

`-v`
Display commands to standard output as they are processed.

`-C`
Create the DHCP network table for the network specified by *network*. See Operands. For details, see `dhcp_network(4)` and `networks(4)`.

`-D` *name_IP_address*
Delete the specified client entry with hostname or client IP address, *name_IP_address*, in the named DHCP network table. (See `dhcp_network(4)`.)

The following sub-option is optional:

`-y`
Remove associated host table entry. The `-y` option requests that all hostnames associated with the IP address in the hosts table in the resource be removed.

`-L`
List the DHCP network tables presently configured, one per line, on standard output. If none are found, no output is printed and an exit status of `0` is returned.

`-M` *name_IP_address*
Modify the specified client entry with hostname or client IP address, *name_IP_address*, in the named DHCP network table. See `dhcp_network(4)`. The default for the sub-options is what they currently are set to.

The following sub-options are optional.

`-c` *comment*
New comment text.

`-e` *mm/dd/yy*
New absolute lease expiration date. Time defaults to 12:00 AM of the day specified.

`-f` *num* | *keyboard*
New flag value, see explanation following the description of the `-A` option.

`-h` *host_name*
New client hostname.

The `-h` option allows you to change the current *hostname* associated with the IP address or to add a new *hostname* to the hosts table if an entry associated with this IP address does not exist.

`-i` *client_ID*
New client identifier [`-a`].

`-m` *macro* [`-y`]
Macro name defined in `dhcptab`.

`-n` *new_client_IP_address*
New IP address.

`-s` *server*
New server IP or name.

For more detailed description of the sub-options and flag values, see `dhcp_network(4)`.

`-P`
Display the named DHCP network table.

The following sub-options are optional:

`-v`
Display lease time in full verbose format and resolve IP addresses for the clients and server to hostnames.

`-x`
Display lease time in raw format.

These flag codes are used with the `-P` sub-options:

`-v`	`-x`	Description
D	00	DYNAMIC
P	01	PERMANENT
M	02	MANUAL
U	04	UNUSABLE
B	08	BOOTP

See `dhcp_network(4)` for information on these sub-options and associated flag codes.

`-p` *path*
Override the `dhcpsvc.conf(4)` configuration value for data store resource path, *path* See `dhcpsvc.conf(4)`

`-R`
Remove the named DHCP network table. See `dhcp_network(4)`.

`-r` *data_store_resource*
Override the `/etc/inet/dhcpsvc.conf` configuration value for `RESOURCE=` with the *data_store_resource* specified. See the `dhcpsvc.conf(4)` man page for more details on resource type, and the *Oracle Solaris DHCP Service Developer's Guide* for more information about adding support for other data stores.

`-u` uninterpreted
Data which will be ignored by `pntadm`, but passed to the currently configured public module to be interpreted by the data store. This might be used for a database account name or other authentication or authorization parameters required by a particular data store.

Operands The following operand is supported:

network
The network address or network name which corresponds to the `dhcp network` table. See `dhcp_network(4)`.

Examples **EXAMPLE 1** Creating a Table for the 10.0.0.0 DHCP Network

The following command creates a table for the `10.0.0.0` (subnetted to class C) DHCP network table. Note that if you have an alias for this network in your `networks(4)` table, you can use that value rather than the dotted Internet Address notation.

```
example# pntadm -C 10.0.0.0
```

EXAMPLE 2 Adding an Entry to the 10.0.0.0 Table

The following command adds an entry to the `10.0.0.0` table in the `files` resource in the `/var/mydhcp` directory:

```
example# pntadm -r SUNWfiles -p /var/mydhcp -A 10.0.0.1 10.0.0.0
```

EXAMPLE 3 Modifying the 10.0.0.1 Entry of the 10.0.0.0 Table

The following command modifies the `10.0.0.1` entry of the `10.0.0.0` table, changing the macro name to `Green`, setting the flags field to `MANUAL` and `PERMANENT`:

```
example# pntadm -M 10.0.0.1 -m Green -f 'PERMANENT+MANUAL' 10.0.0.0
```

EXAMPLE 4 Changing the 10.0.0.1 Entry to 10.0.0.2

The following command changes the `10.0.0.1` entry to `10.0.0.2`, making an entry in the `hosts(4)` table called `myclient`:

```
example# pntadm -M 10.0.0.1 -n 10.0.0.2 -h myclient 10.0.0.0
```

EXAMPLE 5 Setting the Client ID as ASCII

The following command sets the client ID as ASCII `aruba.foo.com` for the `myclient` entry:

```
example# pntadm -M myclient -i 'aruba.foo.com' -a 10.0.0.0
```

EXAMPLE 6 Deleting the myclientEntry from the 10.0.0.0 Table

The following command deletes the `myclient` (`10.0.0.2`) entry from the `10.0.0.0` table:

```
example# pntadm -D myclient 10.0.0.0
```

EXAMPLE 7 Listing the Configured DHCP Network Tables

The following command lists the configured DHCP network tables:

```
example# pntadm -L
192.168.0.0
10.0.0.0
```

EXAMPLE 8 Executing `pntadm` Commands in Batch Mode

The following command runs a series of `pntadm` commands contained in a batch file:

```
example# pntadm -B addclients
```

Exit Status

`0`
Successful completion.

`1`
Object already exists.

`2`
Object does not exist.

`3`
Non-critical error.

`4`
Critical error.

Files

- `/etc/inet/dhcpsvc.conf`
- `/etc/inet/hosts`

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	service/network/dhcp
Interface Stability	Obsolete

See Also

`dhcpconfig(1M)`, `dhcpmgr(1M)`, `dhcp_network(4)`, `dhcpsvc.conf(4)`, `dhcptab(4)`, `hosts(4)`, `netmasks(4)`, `networks(4)`, `user_attr(4)`, `attributes(5)`, `dhcp(5)`, `dhcp_modules(5)`, `rbac(5)`

Oracle Solaris DHCP Service Developer's Guide

System Administration Guide: IP Services

Alexander, S., and R. Droms, *DHCP Options and BOOTP Vendor Extensions*, RFC 1533, Lachman Technology, Inc., Bucknell University, October 1993.

Droms, R., *Interoperation Between DHCP and BOOTP*, RFC 1534, Bucknell University, October 1993.

Droms, R., *Dynamic Host Configuration Protocol*, RFC 1541, Bucknell University, October 1993.

Wimer, W., *Clarifications and Extensions for the Bootstrap Protocol*, RFC 1542, Carnegie Mellon University, October 1993.

Name polkit-is-privileged – check PolicyKit privileges

Synopsis

```
polkit-is-privileged  [-hvV] -u user -p privilege [-r resource]
```

Description The `polkit-is-privileged` command queries system policy to determine whether a user is allowed for a given privilege and resource. The resource name can be omitted. On the Solaris operating system, RBAC authorizations names should be used as privilege names.

Currently, the only consumer of PolicyKit is `hald(1M)`.

Options The following options are supported:

`-h`, `--help`	Display list of options and exit.
`-p` *privilege*, `--privilege` *privilege*	Name of privilege associated with user. Command tests for this privilege.
`-r` *resource*, `--resource` *resource*	Name of resource associated with user and privilege. Command tests for this resource.
`-u` *user*, `--user` *user*	User name or user id that is tested for.
`-v`, `--verbose`	Verbose mode.
`-V`, `--version`	Displays version number.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/library/policykit
Interface Stability	Volatile

See Also `auths(1)`, `profiles(1)`, `hald(1M)`, `getauthattr(3C)`, `auth_attr(4)`, `policy.conf(4)`, `prof_attr(4)`, `user_attr(4)`, `attributes(5)`

Name pooladm – activate and deactivate the resource pools facility

Synopsis

```
/usr/sbin/pooladm [-n] [-s] [-c] [filename] | -x

/usr/sbin/pooladm [-d | -e]
```

Description The `pooladm` command provides administrative operations on pools and sets. `pooladm` reads the specified filename and attempts to activate the pool configuration contained in it.

Before updating the current pool run-time configuration, `pooladm` validates the configuration for correctness.

Without options, `pooladm` prints out the current running pools configuration.

Options The following options are supported:

`-c` Instantiate the configuration at the given location. If a filename is not specified, it defaults to `/etc/pooladm.conf`.

`-d` Disable the pools facility so that pools can no longer be manipulated.

`-e` Enable the pools facility so that pools can be manipulated.

`-n` Validate the configuration without actually updating the current active configuration. Checks that there are no syntactic errors and that the configuration can be instantiated on the current system. No validation of application specific properties is performed.

`-s` Update the specified location with the details of the current dynamic configuration.

This option requires update permission for the configuration that you are going to update. If you use this option with the `-c` option, the dynamic configuration is updated before the static location.

`-x` Remove the currently active pool configuration. Destroy all defined resources, and return all formerly partitioned components to their default resources.

Operands The following operands are supported:

filename Use the configuration contained within this file.

Examples **EXAMPLE 1** Instantiating a Configuration

The following command instantiates the configuration contained at `/home/admin/newconfig`:

```
example# /usr/sbin/pooladm -c /home/admin/newconfig
```

EXAMPLE 2 Validating the Configuration Without Instantiating It

The following command attempts to instantiate the configuration contained at `/home/admin/newconfig`. It displays any error conditions that it encounters, but does not actually modify the active configuration.

EXAMPLE 2 Validating the Configuration Without Instantiating It *(Continued)*

```
example# /usr/sbin/pooladm -n -c /home/admin/newconfig
```

EXAMPLE 3 Removing the Current Configuration

The following command removes the current pool configuration:

```
example# /usr/sbin/pooladm -x
```

EXAMPLE 4 Enabling the Pools Facility

The following command enables the pool facility:

```
example# /usr/sbin/pooladm -e
```

EXAMPLE 5 Enabling the Pools Facility Using SMF

The following command enables the pool facility through use of the Service Management Facility. See `smf(5)`.

```
example# /usr/sbin/svcadm enable svc:/system/pools:default
```

EXAMPLE 6 Saving the Active Configuration to a Specified Location

The following command saves the active configuration to `/tmp/state.backup`:

```
example# /usr/sbin/pooladm -s /tmp/state.backup
```

Files `/etc/pooladm.conf` Configuration file for `pooladm`.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/resource-mgmt/resource-pools
Interface Stability	See below.

The invocation is Committed. The output is Uncommitted.

See Also `poolcfg(1M)`, `poolbind(1M)`, `psrset(1M)`, `svcadm(1M)`, `pset_destroy(2)`, `libpool(3LIB)`, `attributes(5)`, `smf(5)`

Oracle Solaris Administration: Oracle Solaris Zones, Oracle Solaris 10 Zones, and Resource Management

Notes Resource bindings that are not presented in the form of a binding to a partitionable resource, such as the scheduling class, are not necessarily modified in a `pooladm -x` operation.

The pools facility is not active by default when Solaris starts. `pooladm -e` explicitly activates the pools facility. The behavior of certain APIs related to processor partitioning and process binding are modified when pools is active. See `libpool(3LIB)`.

You cannot enable the pools facility on a system where processor sets have been created. Use the `psrset(1M)` command or `pset_destroy(2)` to destroy processor sets manually before you enable the pools facility.

Because the Resource Pools facility is an `smf(5)` service, it can also be enabled and disabled using the standard SMF interfaces.

Name poolbind – bind processes, tasks, or projects or query binding of processes to resource pools

Synopsis

```
/usr/sbin/poolbind -p poolname -e command [arguments]...

/usr/sbin/poolbind -p poolname [-i idtype] id...

/usr/sbin/poolbind -q pid...

/usr/sbin/poolbind -Q pid...
```

Description The `poolbind` command allows an authorized user to bind zones, projects, tasks, and processes to pools. With the `-e` option (see below), it can execute a command you specify, placing the executed command in a specified pool. It can also enable you to query a process to determine which pool a process is bound to.

Options The following options are supported:

`-e` *command* [*arguments*...] Executes *command*, bound to the pool you specify with `-p`.

`-i` *idtype* This option, together with the *idlist* arguments, specifies one or more processes to which the poolbind command is to apply. The interpretation of *idlist* depends on the value of *idtype*. The valid *idtype* arguments and corresponding interpretations of *idlist* are as follows:

- *pid* — *idlist* is a list of process IDs. Binds the specified processes to the specified pool. This is the default behavior if no idtype is specified.
- *taskid* — *idlist* is a list of task IDs. Bind all processes within the list of task IDs to the specified pool.
- *projid* — *idlist* is a list of project IDs. Bind all processes within the list of projects to the specified pool. Each project ID can be specified as either a project name or a numerical project ID. See `project(4)`.
- *zoneid* — *idlist* is a list of zone IDs. Bind all processes within the list of zones to the specified pool. Each zone ID can be specified as either a zone name or a numerical zone ID. See `zones(5)`.

`-p` *poolname* Specifies the name of a pool to which the specified zone, project, tasks, or processes are to be bound.

`-q` *pid* ... Queries the pool bindings for a given list of process IDs. If the collection of resources associated with the process does not correspond to any currently existing pool, or if there are multiple pools with the set of resources that the process is bound to, the query fails for that particular process ID.

`-Q` *pid* ... Queries the resource bindings for a given list of process IDs. The resource bindings are each reported on a separate line.

Examples

EXAMPLE 1 Binding All Processes

The following command binds all processes in projects `5` and `7` to the pool `web_app`:

```
example# /usr/sbin/poolbind -p web_app -i projid 5 7
```

EXAMPLE 2 Binding the Running Shell

The following command binds the running shell to the pool `web_app`:

```
 example# /usr/sbin/poolbind -p web_app $$
```

EXAMPLE 3 Querying the Pool Bindings

The following command queries the bindings to verify that the shell is bound to the given pool:

```
example# /usr/sbin/poolbind -q $$
```

EXAMPLE 4 Querying the Resource Bindings

The following command queries the bindings to verify that the shell is bound to the given resources:

```
example# /usr/sbin/poolbind -Q $$
```

Exit Status

The following exit values are returned:

`0` Successful completion.

`1` Requested operation could not be completed.

`2` Invalid command line options were specified.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/resource-mgmt/resource-pools
Interface Stability	See below.

The invocation is Committed. The output is Uncommitted.

See Also

`pooladm(1M)`, `poolcfg(1M)`, `libpool(3LIB)`, `project(4)`, `attributes(5)`, `zones(5)`

Oracle Solaris Administration: Oracle Solaris Zones, Oracle Solaris 10 Zones, and Resource Management

Name poolcfg – create and modify resource pool configuration files

Synopsis

```
/usr/sbin/poolcfg -c command [-d | [filename]]

/usr/sbin/poolcfg -f command_file [-d | [filename]]

/usr/sbin/poolcfg -h
```

Description The `poolcfg` utility provides configuration operations on pools and sets. These operations are performed upon an existing configuration and take the form of modifications to the specified configuration file. If you use the `-d` option, the modifications occur to the kernel state. Actual activation of the resulting configuration is achieved by way of the `pooladm(1M)` utility.

Pools configuration files are structured files that must have been constructed using `poolcfg` itself or `libpool(3LIB)` directly.

An invocation of `poolcfg` with the pool dynamic location and write permission will hang if the dynamic location has already been opened for writing.

The configurations which are created by this utility can be used by `pooladm` to instantiate the configuration upon a target host.

Options The following options are supported:

`-c` *command*	Specify *command* as an editing command. See `USAGE`.
`-d`	Operate directly on the kernel state. No *filename* is allowed.
`-f` *command_file*	Take the commands from *command_file*. *command_file* consists of editing commands, one per line.
`-h`	Display extended information about the syntax of editing commands.

Usage

Scripts A script consists of editing commands, one per line, of the following:

info [*entity-name*]
Display configuration (or specified portion) in human readable form to standard output. If no entity is specified, system information is displayed. Therefore, `poolcfg -c 'info' afile` is an equivalent invocation to `poolcfg -c 'info system name' afile`.

create *entity-name* [*property-list*]
Make an entity of the specified type and name.

destroy *entity-name*
Remove the specified entity.

modify *entity-name* [*property-list*]
Change the listed properties on the named entity.

associate *pool-name* [*resource-list*]
Connect one or more resources to a pool, or replace one or more existing connections.

transfer to [*resourcetype*] *name*[*component-list*]
Transfer one or more discrete components to a resource .

transfer [*quantity*] from [*resourcetype*] [*src*] to [*tgt*]
Transfer a resource quantity from *src* to *tgt*.

transfer [*quantity*] to [*resourcetype*] [*tgt*] from [*src*]
Transfer a resource quantity to *tgt* from *src*.

discover
Create a system entity, with one pool entity and resources to match current system configuration. All discovered resources of each resource type are recorded in the file, with the single pool referring to the default resource for each resource type.

This command is a NO-OP when `poolcfg` operates directly on the kernel. See the `-d` option.

You should avoid use of this command. The preferred method for creating a configuration is to export the dynamic configuration using `pooladm(1M)` with the `-s` option.

rename *entity-name* to *new-name*
Change the name of an entity on the system to its new name.

Property Lists The property list is specified by:

```
( proptype name = value [ ; proptype name = value ]* )
```

where the last definition in the sequence for a given proptype, name pair is the one that holds. For property deletion, use ~ `proptype name`.

Resource Lists A resource list is specified by:

```
( resourcetype name [ ; resourcetype name ]* )
```

where the last specification in the sequence for a resource is the one that holds. There is no deletion syntax for resource lists.

Component Lists A component list is specified by:

```
( componenttype name [ ; componenttype name ]* )
```

where the last specification in the sequence for a component is the one that holds. There is no deletion syntax for component lists.

Recognized Entities

system	Machine level entity
pool	Named collection of resource associations

Resource Types

pset	Processor set resource

Property Types

boolean	Takes one of two values true or false.
int	A 64–bit signed integer value.
uint	A 64–bit unsigned integer value.
string	Strings are delimited by quotes ("), and support the character escape sequences defined in `formats(5)`.
float	Scientific notation is not supported.

Examples

EXAMPLE 1 Writing a poolcfg Script

The following `poolcfg` script creates a pool named `Accounting`, and a processor set, `small-1`. The processor set is created first, then the pool is created and associated with the set.

```
create pset small-1 ( uint pset.min = 1 ; uint pset.max = 4)
create pool Accounting
associate pool Accounting ( pset small-1 )
```

EXAMPLE 2 Reporting on pool_0

The following command reports on `pool_0` to standard output in human readable form:

```
 # poolcfg -c 'info pool pool_0' /etc/pooladm.conf
```

EXAMPLE 3 Destroying pool_0 and Its Associations

The following command destroys `pool_0` and associations, but not the formerly associated resources:

```
# poolcfg -c 'destroy pool pool_0' /etc/pooladm.conf
```

EXAMPLE 4 Displaying the Current Configuration

The following command displays the current configuration:

```
$ poolcfg -c 'info' /etc/pooladm.conf
system example_system
                int system.version 1
                boolean system.bind-default true
                string system.comment Discovered by libpool

                pool pool_default
                     boolean pool.default true
                     boolean pool.active true
                     int pool.importance 5
                     string pool.comment
                     string.pool.scheduler FSS
                     pset pset_default

                pset pset_default
```

EXAMPLE 4 Displaying the Current Configuration *(Continued)*

```
int pset.sys_id -1
string pset.units population
boolean pset.default true
uint pset.max 4294967295
uint pset.min 1
string pset.comment
boolean pset.escapable false
uint pset.load 0
uint pset.size 2

cpu
     int cpu.sys_id 0
     string cpu.comment

cpu
     int cpu.sys_id 2
     string cpu.comment
```

EXAMPLE 5 Moving cpu with ID 2 to Processor Set pset1 in the Kernel

The following command moves `cpu` with ID `2` to processor set `pset1` in the kernel:

```
# poolcfg -dc 'transfer to pset pset1 ( cpu 2 )'
```

EXAMPLE 6 Moving 2 cpus from Processor Set pset1 to Processor Set pset2 in the Kernel

The following command moves 2 `cpus` from processor set `pset1` to processor set `pset2` in the kernel:

```
# poolcfg -dc 'transfer 2 from pset pset1 to pset2'
```

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/resource-mgmt/resource-pools
Interface Stability	See below.

The invocation is Committed. The output is Uncommitted.

See Also

`pooladm(1M)`, `poolbind(1M)`, `libpool(3LIB)`, `attributes(5)`, `formats(5)`

Oracle Solaris Administration: Oracle Solaris Zones, Oracle Solaris 10 Zones, and Resource Management

Name poold – automated resource pools partitioning daemon

Synopsis

```
poold [-l level]
```

Description `poold` provides automated resource partitioning facilities. `poold` can be enabled or disabled using the Solaris Service Management Facility, `smf(5)`. `poold` requires the Resource Pools facility to be active in order to operate.

The dynamic resource pools service's fault management resource identifier (FMRI) is:

```
svc:/system/pools/dynamic
```

The resource pools service's FMRI is:

```
svc:/system/pools
```

`poold`'s configuration details are held in a `libpool(3LIB)` configuration and you can access all customizable behavior from this configuration.

`poold` periodically examines the load on the system and decides whether intervention is required to maintain optimal system performance with respect to resource consumption. `poold` also responds to externally initiated (with respect to `poold`) changes of either resource configuration or objectives.

If intervention is required, `poold` attempts to reallocate the available resources to ensure that performance objectives are satisfied. If it is not possible for `poold` to meet performance objectives with the available resources, then a message is written to the log. `poold` allocates scarce resources according to the objectives configured by the administrator. The system administrator must determine which resource pools are most deserving of scarce resource and indicate this through the importance of resource pools and objectives.

Options The following options are supported:

`-l` *level* Specify the vebosity level for logging information.

Specify *level* as `ALERT`, `CRIT`, `ERR`, `WARNING`, `NOTICE`, `INFO`, and `DEBUG`. If *level* is not supplied, then the default logging level is `INFO`.

`ALERT`	A condition that should be corrected immediately, such as a corrupted system database.
`CRIT`	Critical conditions, such as hard device errors.
`ERR`	Errors.
`WARNING`	Warning messages.
`NOTICE`	Conditions that are not error conditions, but that may require special handling.
`INFO`	Informational messages.

DEBUG — Messages that contain information normally of use only when debugging a program.

When invoked manually, with the `-l` option, all log output is directed to standard error.

Examples

EXAMPLE 1 Modifying the Default Logging Level

The following command modifies the default logging level to ERR:

```
# /usr/lib/pool/poold -l ERR
```

EXAMPLE 2 Enabling Dynamic Resource Pools

The following command enables dynamic resource pools:

```
# /usr/sbin/svcadm enable svc:/system/pools/dynamic
```

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/resource-mgmt/resource-pools
Interface Stability	See below.

The invocation is Committed. The output is Uncommitted.

See Also

`pooladm(1M)`, `poolbind(1M)`, `poolcfg(1M)`, `poolstat(1M)`, `svcadm(1M)`, `pool_set_status(3POOL)`, `libpool(3LIB)`, `attributes(5)`, `smf(5)`

Oracle Solaris Administration: Oracle Solaris Zones, Oracle Solaris 10 Zones, and Resource Management

Name poolstat – report active pool statistics

Synopsis

```
poolstat [-p pool-list] [-r rset-list] [-T u | d ] [interval [count]]

poolstat [-p pool-list] [-o format -r rset-list]
     [-T u | d ] [interval [count]]
```

Description The `poolstat` utility iteratively examines all active pools on the system. It reports statistics based on the selected output mode. `poolstat` provides options to examine only specified pools and report resource set-specific statistics.

Without options, `poolstat` examines all pools, reports basic statistics for their resource sets, and exits.

DISPLAY FORMATS In default output format, `poolstat` outputs a header line and a line for each pool. The line begins with the pool ID and its name, followed by a column of statistical data for the processor set attached to the pool.

The columns are defined as follows:

`id` Pool ID.

`pool` Pool name.

`rid` Resource set id.

`rset` Resource set name.

`type` Resource set type.

`min` Minimum resource set size.

`max` Maximum resource set size.

`size` Current resource set size.

`used` The measure of how much of the resource set is currently is use. This is calculated as the percentage utilization of the resource set multiplied by its size. If resource set has been reconfigured during last sampling interval, this value might be not reported (`-`).

`load` The absolute representation of the load that is put on the resource set. For the definition of this property see `libpool(3LIB)`.

Options The following options are supported:

`-o` *format* Report statistics according to the format specification given in format. See `DISPLAY FORMATS`.

The `-o` option accepts lists as arguments. Items in a list can be either separated by commas or enclosed in quotes and separated by commas or spaces.

You can specify multiple `-o` options. The format specification is interpreted as the whitespace separated concatenation of all the format option arguments.

The `-o` option must be used in conjunction with the `-r` option.

`-p` *pool-list* Report only pools whose names are in the given list. If the -r option is also used, this option selects only resource sets which belong to pools in the given list. Statistics for pools or resource sets are reported in the same order in which pool names are listed on the pool-list. Pool can be specified by name or by ID.

The `-p` option accepts lists as arguments. Items in a *pool-list* can only be separated by spaces.

`-r` *rset-list* Report resource set statistics. If the *rset-list* argument is "`all`", then all possible resource set types are selected.

The `-r` option accepts lists as arguments. Items in a list can be either separated by commas or enclosed in quotes and separated by commas or spaces.

The following resource set types are supported:

`all` All resource set types

`pset` Processor set

`-T u | d` Display a time stamp.

Specify `u` for a printed representation of the internal representation of time. See `time(2)`. Specify `d` for standard date format. See `date(1)`.

Operands The following operands are supported:

count The number of times that the statistics are repeated. By default, poolstat reports statistics only once.

If neither interval nor count are specified, statistics are reported once. If interval is specified and count is not, statistics are reported indefinitely.

interval The sampling interval in seconds.

If neither interval nor count are specified, statistics are reported once. If interval is specified and count is not, statistics are reported indefinitely.

Examples **EXAMPLE 1** Using `poolstat`

The following example shows the default output from the `poolstat` utility:

```
% poolstat
                          pset
  id pool           size used load
   0 pool_default      4  3.6  6.2
   1 pool_admin        4  3.3  8.4
```

EXAMPLE 2 Reporting Resource Set Statistics

The following example reports resource set statistics.

```
% poolstat -r pset
  id pool           type rid rset          min  max size used load
  0 pool_default    pset  -1 pset_default    1  65K    2  1.2  8.3
  1 pool_admin      pset   1 pset_admin      1    1    1  0.4  5.2
  2 pool_other      pset  -1 pset_default    1  65K    2  1.2  8.3
```

Resource sets attached to multiple pools, as `pset_default` in the example above, are listed multiple times, once for each pool.

EXAMPLE 3 Restricting the Output to the List of Pools

The following example restricts the output to the list of pools

```
% poolstat -p pool_default
                    pset
 id pool         size used load
 0 pool_default    8  5.3 10.3

% poolstat -p 'pool_admin pool_default'
                      pset
  id pool         size used load
  1 pool_admin      6  4.3  5.3
  0 pool_default    2  1.9  2.0

% poolstat -r all -p 'pool_admin pool_default'
  id pool            type rid rset          min  max size used load
  1 pool_admin      pset   1 pset_admin      1    1    1  0.9  2.3
  2 pool_default    pset  -1 pset_default    1  65K    2  2.0  2.0
```

EXAMPLE 4 Customizing Output

The following example customizes output:

```
% poolstat -r -o pool,rset,size,load
  pool            rset          size load
  pool_default    pset_default     4  4.5
  pool_admin      pset_admin       4  2.1
```

Exit Status The following exit values are returned:

`0` Successful completion.

1 An error occurred.

2 Invalid command line options were specified.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/resource-mgmt/resource-pools
Interface Stability	Committed

See Also `libpool(3LIB)`, `attributes(5)`

Oracle Solaris Administration: Oracle Solaris Zones, Oracle Solaris 10 Zones, and Resource Management

Notes The system ids associated with resources can change after the system reboots or the resource configuration is altered.

Name ports – creates /dev entries and inittab entries for serial lines

Synopsis `/usr/sbin/ports [-r` *rootdir*`]`

Description `devfsadm(1M)` is now the preferred command for `/dev` and `/devices` and should be used instead of `ports`.

The `ports` command creates symbolic links in the `/dev/term` and `/dev/cua` directories to the serial-port character device files in `/devices` and adds new entries in `/etc/inittab` for non-system ports found. System-board ports are given single lower-case letters for names (such as `a` and `b`) while other ports are named numerically.

`ports` searches the kernel device tree to find the serial devices attached to the system. It also checks `/dev/term` and `/dev/cua` to see what symbolic links to serial devices already exist. `ports` then performs the following:

1. Assigns new numbers (or letters for system-board ports) to ports that are attached to the system but do not have `/dev/term` and `/dev/cua` entries. The numbers or letters assigned are the lowest-unused numbers or letters.
2. Removes dangling links: links from `/dev/term` and `/dev/cua` pointing to no-longer-existing ports.
3. Creates new `/dev/term` and `/dev/cua` links for new serial devices.

If the configuration has not changed, ports exits without doing anything.

Notice to Driver Writers `ports` considers devices with a node type of `DDI_NT_SERIAL`, `DDI_NT_SERIAL_MB`, `DDI_NT_SERIAL_DO`, or `DDI_NT_SERIAL_MB_DO` to be serial port devices. Devices with one of these node types must create minor device names that obey the following conventions when calling `ddi_create_minor_node(9F)`.

- The minor name for non-system port devices (`DDI_NT_SERIAL`) consists of an ASCII numeric string, where the first port on the device is named `0`, the second named `1`, the third named `2`, up to the number of ports provided by the device.
- The minor name for non-system dialout devices (`DDI_NT_SERIAL_DO`) is the ASCII numeric port name, concatenated with `,cu`. For example, the minor name for the first dialout port on the serial board is `0,cu`.
- The minor name for system-board port devices (`DDI_NT_SERIAL_MB`) consists of a string containing a single ASCII lowercase character, where the first port on the device is named `a`, the second is named `b`, the third is named `c`, for all ports on the device (or up through port `z`).
- The minor name for system-board dialout devices (`DDI_NT_SERIAL_MB_DO`) consists of the lowercase character port name, concatenated with `,cu`. For example, the minor name for the first dialout port on the on-board serial device is `a,cu`.

To prevent `disks` from attempting to automatically generate links for a device, drivers must specify a private node type and refrain from using one of the above node types when calling `ddi_create_minor_node(9F)`.

Options The following options are supported:

`-r` *rootdir* Causes `ports` to presume that the `/dev/term`, `/dev/cua`, and `/devices` directories are found under *rootdir*, not directly under `/`.

Examples **EXAMPLE 1** Creating the Serial and Dialout Minor Device Nodes

The following example creates the serial and dialout minor device nodes from the `xkserial` driver's `attach(9E)` function:

```
/*
* Create the minor number by combining the instance number
* with the port number.
*/ #define XKNUMPORTS           8
#define XKMINORNUM(i, p)     ((i) << 4 | (p))
#define XKMINORNUM_DO(i, p)  ((i) << 4 | (p) | 0x80)
int
xkserialattach(dev_info_t *dip, ddi_attach_cmd_t cmd)
{
   int instance, portnum;
   char name[8];
       /* other stuff in attach... */
   instance = ddi_get_instance(dip);
   for (portnum = 0; portnum < XKNUMPORTS; portnum++) {
       /*
        * create the serial port device
        */
        sprintf(name, "%d", portnum);
        ddi_create_minor_node(dip, name, S_IFCHR,
           XKMINORNUM(instance, portnum), DDI_NT_SERIAL, 0);

        /*
         * create the dialout device
         */
         sprintf(name,"%d,cu", portnum);
         ddi_create_minor_node(dip, name, S_IFCHR,
             XKMINORNUM_DO(instance, portnum), DDI_NT_SERIAL_DO, 0);
     }
}
```

EXAMPLE 2 Installing the xkserial Port Driver on a Sun Fire 4800

The following example installs the `xkserial` port driver on a Sun Fire 4800 (with the driver controlling the fictional XKSerial 8 port serial board), with these special files in `/devices`:

```
# ls -l /devices/ssm@0,0/pci@18,700000/pci@1/xkserial@f,800000/
crw-r-----   1 root sys   32,  16 Aug 29 00:02 xkserial@2000:0
crw-r-----   1 root sys   32, 144 Aug 29 00:02 xkserial@2000:0,cu
crw-r-----   1 root sys   32,  17 Aug 29 00:02 xkserial@2000:1
```

EXAMPLE 2 Installing the xkserial Port Driver on a Sun Fire 4800 *(Continued)*

```
crw-r-----   1 root sys   32, 145 Aug 29 00:02 xkserial@2000:1,cu
crw-r-----   1 root sys   32,  18 Aug 29 00:02 xkserial@2000:2
crw-r-----   1 root sys   32, 146 Aug 29 00:02 xkserial@2000:2,cu
crw-r-----   1 root sys   32,  19 Aug 29 00:02 xkserial@2000:3
crw-r-----   1 root sys   32, 147 Aug 29 00:02 xkserial@2000:3,cu
crw-r-----   1 root sys   32,  20 Aug 29 00:02 xkserial@2000:4
crw-r-----   1 root sys   32, 148 Aug 29 00:02 xkserial@2000:4,cu
crw-r-----   1 root sys   32,  21 Aug 29 00:02 xkserial@2000:5
crw-r-----   1 root sys   32, 149 Aug 29 00:02 xkserial@2000:5,cu
crw-r-----   1 root sys   32,  22 Aug 29 00:02 xkserial@2000:6
crw-r-----   1 root sys   32, 150 Aug 29 00:02 xkserial@2000:6,cu
crw-r-----   1 root sys   32,  23 Aug 29 00:02 xkserial@2000:7
crw-r-----   1 root sys   32, 151 Aug 29 00:02 xkserial@2000:7,cu
```

`/dev/term` contain symbolic links to the serial port device nodes in `/devices`

```
# ls -l /dev/term
/dev/term/0 -> ../../devices/[....]/xkserial@2000:0
/dev/term/1 -> ../../devices/[....]/xkserial@2000:1
/dev/term/2 -> ../../devices/[....]/xkserial@2000:2
/dev/term/3 -> ../../devices/[....]/xkserial@2000:3
/dev/term/4 -> ../../devices/[....]/xkserial@2000:4
/dev/term/5 -> ../../devices/[....]/xkserial@2000:5
/dev/term/6 -> ../../devices/[....]/xkserial@2000:6
/dev/term/7 -> ../../devices/[....]/xkserial@2000:7
```

and `/dev/cua` contain symbolic links to the dialout port device nodes in `/devices`

```
# ls -l /dev/cua

/dev/cua/0 -> ../../devices/[....]/xkserial@2000:0,cu
/dev/cua/1 -> ../../devices/[....]/xkserial@2000:1,cu
/dev/cua/2 -> ../../devices/[....]/xkserial@2000:2,cu
/dev/cua/3 -> ../../devices/[....]/xkserial@2000:3,cu
/dev/cua/4 -> ../../devices/[....]/xkserial@2000:4,cu
/dev/cua/5 -> ../../devices/[....]/xkserial@2000:5,cu
/dev/cua/6 -> ../../devices/[....]/xkserial@2000:6,cu
/dev/cua/7 -> ../../devices/[....]/xkserial@2000:7,cu
```

Files

`/dev/term/`*n*	Logical serial port devices
`/dev/cua/`*n*	Logical dialout port devices
`/etc/inittab`	Controls dispatching by `init`.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also add_drv(1M), devfsadm(1M), drvconfig(1M), attributes(5), devfs(7FS), attach(9E), ddi_create_minor_node(9F)

Writing Device Drivers

Name poweradm – manage power management properties

Synopsis

```
poweradm [-v] get [-a all|smf|platform|current] property ...

poweradm [-v] set property=value ...

poweradm [-v] list

poweradm show

poweradm -?
```

Description The `poweradm` program is used to display and manage the Power Management settings within a Solaris instance.

The Power Management properties and their values are stored in the service management facility (see `smf(5)`).

All users can run the `list`, `get`, and `show` forms given in the `SYNOPSIS`. These commands allow all users to view the current Power Management settings within a Solaris instance.

Only users and roles that belong to the Maintenance and Repair RBAC profile can execute the `set` form of `poweradm` given in the `SYNOPSIS`. See also the `NOTES` section below.

`poweradm` supports the following service property:

```
administrative-authority
```

The value for this property is specified as a string and can have the values:

- `smf`
- `platform`
- `none`

...with `platform` as the default value. The significance of this property is the source of administrative control for power management within the Solaris kernel. That is, this property indicates the origin of `time-to-full-capacity` and `time-to-minimum-responsiveness` settings within the Solaris kernel. Only a Solaris user with appropriate privileges can set this property.

When the `administrative-authority` is set to `platform` the values of `time-to-full-capacity` and `time-to-minimum-responsiveness` will be taken from the platform code. Setting these values in SMF, using the `poweradm` command will have no effect upon the values in the kernel. The `poweradm list` command will indicate that the kernel is currently using the platform values. As these values are changed by the platform administrator, so they will be modified in the kernel. Also, commands to enable or disable the power management within the Solaris kernel will come from the platform code. On systems that run virtual machines, the hypervisor or virtual machine manager can be the source of this property.

When the `administrative-authority` is set to `smf`, the values of `time-to-full-capacity` and `time-to-minimum-responsiveness` will be taken from SMF. As these values in SMF are changed by the Solaris administrator, so they will be applied to the kernel. Setting these values in the platform will have no effect upon the values in the kernel while administrative-authority is set to `smf`. Under this condition, the `poweradm` command will indicate that the kernel is currently using the values from SMF. As these values are changed by the Solaris administrator, so they will be modified in kernel.

When the `administrative-authority` is set to `none`, power management within the Solaris kernel will be turned off, although the power service will continue to run. Any power management instructions from the platform will be ignored, as will the settings in SMF. Only when `administrative-authority` is set to one of the other values will power management within the Solaris kernel restart, using the settings from the specified source.

`time-to-full-capacity`
Specified in microseconds.

This parameter constrains the dynamic capacity adjustment allowed while the system is in an active state.

This parameter defines the maximum time the system is allowed to reach (re-provision and make available) its full capacity, returning from any lower-capacity/less-responsive state, while it has been using any or all of the PM features falling within this bound.

By default, this value is taken from the platform (for example, `i86pc`), because the default setting for `administrative-authority` is set to `platform`.

Alternatively, if `administrative-authority` is set to `smf`, this value is taken from the definition provided by the power service (that is, SMF). At install time this value is set to be `undefined`. If the Solaris administrator chooses to modify this property, a value appropriate to the needs of the workload or applications must be picked.

`time-to-minimum-responsiveness`
Specified in milliseconds.

This parameter constrains the dynamic capacity adjustment allowed while the system is in an inactive state.

This parameter defines how long the system is allowed to return to its active state—that is, to provide the minimum capacity required to meet the above time-to-full-capacity constraint.

Moderate values (seconds) allow hardware components or subsystems on the platform to be placed in slower-response inactive states; larger values still (for example, 30 seconds to minutes) allow for such as whole system suspension, using techniques such as suspend-to-RAM.

By default, this value is taken from the platform (for example, `i86pc`), because the default setting for `administrative-authority` is set to `platform`.

Alternatively, if `administrative-authority` is set to `smf`, this value is taken from the definition provided by the power service (that is, SMF). At install time this value is set to be `undefined`. If the Solaris administrator chooses to modify this property, a value appropriate to the needs of the workload or applications must be picked.

`suspend-enable`
By default no machine running Solaris is permitted to attempt a suspend operation. Setting this property to `true` will permit a suspend operation to be attempted. The value of `administrative-authority` has no effect upon this property.

`platform-disabled`
This property cannot be changed by the `poweradm` command. The value of `platform-disabled` can be viewed by running the `list` subcommand. If set to `true` and `administrative-authority` is set to `platform`, power management has been disabled by the platform. If set to `false`, control of power management will be through the values of the other properties, described above. The output of the `show` subcommand will display the values of these properties. On systems that run virtual machines, the hypervisor or virtual machine manager can be the source of `platform-disabled`. The default value for `platform-disabled` is `false`.

Options The following options are supported.

`-?`
Display a synopsis of available subcommands and options.

`-v`
Provide verbose output. Can be used with any of the subcommands listed below.

Sub-commands The following subcommands are supported:

`get` [`-a all`|`smf`|`platform`|`current`]
Retrieves the current value of the named property. The `-a` option can be used to indicate the origin of the value either: SMF (`smf`), the platform (`platform`), the current value used by the kernel (`current`) or all of the preceeding (`all`). By default, if no origin is specified then `current` is assumed.

The `administrative-authority` and `suspend-enable` properties do not have a platform value

`set` *property=value...*
Changes the named *property* to the given *value*. The properties `administrative-authority` and `suspend-enable` are automatically synchronized to the new value in the kernel. The properties `time-to-full-capacity` and `time-to-minimum-responsiveness` are synchronized to the kernel if and only if `administrative-authority` is set to `smf`. If the `-v` option is used and the kernel cannot be updated immediately because `administrative-authority` is not set to `smf`, a warning message will be issued.

Only users and roles that belong to the Maintenance and Repair RBAC profile can execute the `set` subcommand.

`list`
Lists all the available Power Management properties values and indicate whether power management is active.

`show`
Output human readable text that indicates whether the platform or the Solaris instance is controlling power management, whether power management is enabled, and, if it is enabled, the values of `time-to-full-capacity` and `time-to-minimum-responsiveness`.

Examples

EXAMPLE 1 Setting Platform to Control Power Management

The following command sets the platform to control power management.

```
# poweradm set administrative-authority=platform
```

EXAMPLE 2 Disabling Power Management

The following command disables power management.

```
# poweradm set administrative-authority=none
```

EXAMPLE 3 Setting Useful Parameters

The following sequence of commands sets `time-to-full-capacity` to 300 microseconds, sets`time-to-minimum-responsiveness` to 500 milliseconds, and informs the Solaris instance of the new values.

```
# poweradm set time-to-full-capacity=300
# poweradm set time-to-minimum-responsiveness=500
# poweradm set administrative-authority=smf
```

EXAMPLE 4 Disabling Suspend and Resume

The following command disables suspend and resume.

```
# poweradm set suspend-enable=false
```

EXAMPLE 5 Listing Power Management Properties

The following command lists all available power management properties.

```
# poweradm list
```

EXAMPLE 6 Obtaining Value of a Property

The following command shows the current value of `time-to-full-capacity`.

```
# poweradm get time-to-full-capacity
```

EXAMPLE 7 Showing Value of a Property as Set by Platform

The following command retrieves the value of `time-to-full-capacity` set by the platform.

```
# poweradm get -a platform time-to-full-capacity
```

Note that this will only be the same as current value if `administrative-authority` has been set to platform. See the explanation of the `administrative-authority` property, above.

EXAMPLE 8 Showing Value of a Property as Set by Solaris Instance

The following command retrieves the value of `time-to-full-capacity` set by the Solaris instance.

```
# poweradm get -a smf time-to-full-capacity
```

Note that this will only be the same as current value if `administrative-authority` has been set to `smf`. See the explanation of the `administrative-authority` property, above.

EXAMPLE 9 Invoking `show` Subcommand

The following example commands illustrate the four possible contexts in which `poweradm show` can be invoked.

The following command is invoked when power management has been disabled by the platform.

```
# poweradm show
Power management is disabled with the hardware platform as the authority
```

The following command is invoked when power management has been disabled by the Solaris administrator.

```
# poweradm show
Power management is disabled with the Solaris instance as the authority
```

The following command is invoked when power management has been enabled by the platform.

```
# poweradm show
Power management is enabled with the hardware platform as the authority
time-to-full-capacity 300 microseconds
time-to-minimum-responsiveness 500 milliseconds
```

The following command is invoked when power management has been enabled by the Solaris instance.

```
# poweradm show
Power management is enabled with the Solaris instance as the authority
time-to-full-capacity 300 microseconds
time-to-minimum-responsiveness 500 milliseconds
```

Exit Status

`0`

Successful completion.

`1`

An error occurred.

`2`

Invalid command line options were specified.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/kernel/power
Interface Stability	Committed

See Also

`attributes(5)`, `smf(5)`, `smf_security(5)`

Notes

The power service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/system/power:default
```

The properties that can be set by `poweradm` are defined in that service as:

- `active_control/administrative-authority`, which is described under `administrative-authority` above
- `active_config/time-to-full-capacity`, which is described under `time-to-full-capacity` above
- `active_config/time-to-minimum-responsiveness`, which is described under `time-to-minimum-responsiveness` above
- `suspend/suspend-enable`, which is described under `suspend-enable` above

If the service is disabled, no power management settings will be uploaded to the kernel in the future. Existing settings will not be undone until the next reboot. Disabling all power management is best accomplished by setting `administrative-authority` to `none`.

If `administrative-authority` is set to the value `smf` before both `time-to-full-capacity` and `time-to-minimum-responsiveness` have been set, the service will go into maintenance mode. In such a situation, set `administrative-authority` to the value `none` then set both `time-to-full-capacity` and `time-to-minimum-responsiveness` to the values you want, clear the service and then set administrative-authority to `smf`.

To set properties in the `active_config` and `suspend` property groups, the `solaris.smf.value.power_config` authorization is required. To set properties in the

`active_control` property group requires the `solaris.smf.value.power_control` authorization is required. Both of these authorizations are part of the Maintenance and Repair profile.

Name powertop – report and analyze events that affect power management

Synopsis

```
powertop [-c processor_id] [-d count] [-t interval] [-v] [-h]
```

Description `PowerTOP` is an observability tool that shows how effectively the system is taking advantage of the CPU's power management features. By running the tool on an otherwise idle system, the user can see for how long the CPU is running at dif- ferent power states. Ideally, an unutilized (idle) system spends 100% of its time running at the lowest power state, but because of background user and kernel activity (random software periodically waking to poll status), idle systems can consume more power than they should.

The tool analyzes system activity periodically and displays a summary of how long the processor is executing at each supported power state. It also displays the top activities responsible for causing the CPU to wake up and use more energy. This report allows the user to identify and diagnose problematic areas of the system and optimize its power efficiency.

`PowerTOP` averages the amount of activity that is preventing the CPU from entering a lower power state and presents it on the "Wakeups-from-idle per second" field. This value represents the total number of wake-ups divided by the current interval. Notice that not all events are displayed on the screen at all times.

During execution, a user can force a refresh of the analysis by pressing the `R` key. The interval time is restored to the default or to a specified value. To quit the application, the user must press the `Q` key.

`PowerTOP` runs on some virtual domains. However, the report for idle state transitions might or might not be accurate as the physical CPU can be shared by different virtual CPUs. Both wakeup count and event report displays information regarding the current virtualized environment.

Options The following options are supported:

`-c` [*processor_id*]
Specifies which CPU the tool should observe.

`-d` [*count*]
Dumps the results of *count* analysis of system activity to the screen.

`-h`
Displays the command's usage.

`-t` [*interval*]
Specifies the interval, in seconds, at which the tool analyzes the system. The possible values are in the range of 1 through 30; the default is 5 seconds.

`-v`
Switches to verbose mode, including noting firings of the kernel cyclic subsystem in the event report.

Examples

EXAMPLE 1 Setting the Interval

The following command sets the interval to two seconds.

```
% powertop -t 2
```

EXAMPLE 2 Analyzing and Dumping System Activity

The following command analyzes and dumps system activity to the standard output four times.

```
% powertop -d 4
```

EXAMPLE 3 Reporting Cyclic Subsystem Activity

The following command reports cyclic subsystem activity.

```
% powertop -v
```

EXAMPLE 4 Analyzing Activity on a Specific Processor

The following command runs PowerTOP and only displays data for CPU 3:

```
% powertop -c 3
```

Exit Status

`0`
Successful operation.

`1`
An error occurred.

`2`
Incorrect usage.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	x86, SPARC
Availability	diagnostic/powertop
Interface Stability	Volatile

See Also

`kstat(1M)`, `psrinfo(1M)`, `uadmin(2)`, `libdevinfo(3LIB)`, `attributes(5)`, `cpr(7)`, `pm(7D)`, `pm-components(9P)`, `removable-media(9P)`

Among non-SunOS man pages, `xscreensaver(1)` and `dtpower(1M)`.

Usage

You must have `DTrace` privileges to run `PowerTOP`.

Name pppd – point to point protocol daemon

Synopsis `pppd [`*tty_name*`] [`*speed*`] [`*options*`]`

Description The point-to-point protocol (PPP) provides a method for transmitting datagrams over serial point-to-point links. PPP is composed of three components: a facility for encapsulating datagrams over serial links, an extensible link control protocol (LCP), and a family of network control protocols (NCP) for establishing and configuring different network-layer protocols.

The encapsulation scheme is provided by driver code in the kernel. `pppd` provides the basic LCP authentication support and several NCPs for establishing and configuring the Internet Protocol (referred to as the IP Control Protocol or "IPCP") and IPv6 (IPV6CP).

Options The following sections discuss the `pppd` options:

Options Files Options are taken from files and the command line. `pppd` reads options from the files `/etc/ppp/options`, `$HOME/.ppprc` and `/etc/ppp/options.`*ttyname* (in that order) before processing the options on the command line. (Command-line options are scanned for the terminal name before the `options.`*ttyname* file is read.) To form the name of the `options.`*ttyname* file, the initial `/dev/` is removed from the terminal name, and any remaining forward slash characters (/) are replaced with dots. For example, with serial device `/dev/cua/a`, option file `/etc/ppp/options.cua.a` is read.

An options file is parsed into a series of words that are delimited by whitespace. Whitespace can be included in a word by enclosing the word in double-quotes ("). A backslash (\) quotes the succeeding character. A hash (#) starts a comment, which continues until the end of the line. There is no restriction on using the `file` or `call` options within an options file.

Frequently Used Options

`<tty_name>` Communicate over the named device. The string `/dev/` is prepended if necessary. If no device name is given, or if the name of the terminal connected to the standard input is given, `pppd` uses that terminal and does not fork to put itself in the background. A value for this option from a privileged source cannot be overridden by a non-privileged user.

`<speed>` Set the baud rate to `<speed>` (a decimal number). The default is to leave the baud rate unchanged. This option is normally needed for dial-out only.

`asyncmap <`*map*`>` Set the `async` character map to <*map*>. The map describes which control characters cannot be successfully received over the serial line. `pppd` asks the peer to send these characters as a 2-byte escape sequence. The argument is a 32 bit hex number, with each bit representing a character to escape. Bit 0 (00000001) represents the character 0x00; bit 31 (80000000) represents the character 0x1f or ^_. If multiple `asyncmap` options are given, the values are `ORed` together. If no `asyncmap` option is given, `pppd` attempts to negotiate a value of 0. If the

peer agrees, this disables escaping of the standard control characters. Use the `default-asyncmap` option to disable negotiation and escape all control characters.

`auth` — Require the peer to authenticate itself before allowing network packets to be sent or received. This option is the default if the system has a default route. If the `auth` or the `noauth` option is not specified, `pppd` allows the peer to use only those IP addresses to which the system does not already have a route.

`call` *name* — Read options from the file `/etc/ppp/peers/`*name*. This file may contain privileged options, including `noauth`, even if `pppd` is not being run by root. The *name* string may not begin with a slash ("/") or include consecutive periods (`".."`) as a pathname component.

`callback` *number* — Request a callback to the given telephone number using Microsoft CBCP.

`connect` *script* — Use the executable or shell command specified by *script* to set up the serial line. This script would typically use the `chat(1M)` program to dial the modem and start the remote `PPP` session. A value for this option originating from a privileged source cannot be overridden by a non-privileged user.

`crtscts` — Use hardware flow control, that is, RTS/CTS, to control the flow of data on the serial port. If the `crtscts`, `nocrtscts`, `cdtrcts` or `nocdtrcts` option is not provided, the hardware flow control setting for the serial port is left unchanged. Some serial ports lack a true RTS output and use this mode to implement unidirectional flow control. The serial port suspends transmission when requested by the modem by means of CTS but cannot request the modem to stop sending to the computer. This mode allows the use of DTR as a modem control line.

`defaultroute` — Add a default route to the system routing tables when IPCP negotiation successfully completes, using the peer as the gateway. This entry is removed when the `PPP` connection is broken. This option is privileged if the `nodefaultroute` option is specified.

`disconnect` *script* — Run the executable or shell command specified by *script* after `pppd` terminates the link. Typically, this script is used to command the modem to hang up if hardware modem control signals are not available. `disconnect` is not run if the modem has already hung up. A value for this option originating from a privileged source cannot be overridden by a non-privileged user.

`escape` *xx,yy,...* — Specifies that certain characters be escaped on transmission regardless of whether the peer requests them to be escaped with its `async` control

character map. The characters to be escaped are specified as a list of hex numbers separated by commas. Note that almost any character can be specified for the `escape` option, unlike the `asyncmap` option which allows only control characters to be specified. Characters that cannot be escaped are those containing hex values 0x20 through 0x3f and 0x5e.

`file` *name* — Read options from file *name*. If this option is used on the command line or in `$HOME/.ppprc`, the file must be readable by the user invoking `pppd`. See "Options Files," above, for a list of files that `pppd` always reads, regardless of the use of this option.

`init` *script* — Run the executable or shell command specified by *script* to initialize the serial line. This script would typically use the `chat(1M)` program to configure the modem to enable auto-answer. A value for this option from a privileged source cannot be overridden by a non-privileged user.

`lock` — Directs `pppd` to create a UUCP-style lock file for the serial device to ensure exclusive access to the device.

`mru` *n* — Set the Maximum Receive Unit (MRU) value to *n*. `pppd` asks the peer to send packets of no more than *n* bytes. Minimum MRU value is 128. Default MRU value is 1500. A value of 296 is recommended for slow links (40 bytes for TCP/IP header + 256 bytes of data). For IPv6, MRU must be at least 1280.

`mtu` *n* — Set the Maximum Transmit Unit (MTU) value to *n*. Unless the peer requests a smaller value via MRU negotiation, `pppd` requests the kernel networking code to send data packets of no more than *n* bytes through the PPP network interface. For IPv6, MTU must be at least 1280.

`passive` — Enables the "passive" option in the LCP. With this option, `pppd` attempts to initiate a connection; if no reply is received from the peer, `pppd` waits passively for a valid LCP packet instead of exiting, as it would without this option.

Options

`<local_IP_address>:<remote_IP_address>`

Set the local and/or remote interface IP addresses. Either one may be omitted, but the colon is required. The IP addresses are specified with a host name or in decimal dot notation, for example: `:10.1.2.3`. The default local address is the first IP address of the system unless the `noipdefault` option is provided. The remote address is obtained from the peer if not specified in any option. Thus, in simple cases, this option is not required. If a local and/or remote IP address is specified with this option, `pppd` will not accept a different value from the peer in the IPCP negotiation unless the `ipcp-accept-local` and/or `ipcp-accept-remote` options are given, respectively.

`allow-fcs` *fcs-type*
Set allowable FCS type(s) for data sent to the peer. The *fcs-type* is a comma-separated list of "crc16", "crc32", "null", or integers. By default, all known types are allowed. If this option is specified and the peer requests a type not listed, a LCP Configure-Nak is sent to request only the listed types.

`allow-ip` *address(es)*
Allow peers to use the given IP address or subnet without authenticating themselves. The parameter is parsed in the same manner as each element of the list of allowed IP addresses is parsed in the secrets files. See the "Authentication" section more more details.

`bsdcomp` *nr,nt*
Request that the peer compress packets that it sends using the BSD-Compress scheme, with a maximum code size of *nr* bits, and agree to compress packets sent to the peer with a maximum code size of *nt* bits. If *nt* is not specified, it defaults to the value given for *nr*. Values in the range 9 to 15 may be used for *nr* and *nt*; larger values provide better compression but consume more kernel memory for compression dictionaries. Alternatively, a value of 0 for *nr* or *nt* disables compression in the corresponding direction. Use `nobsdcomp` or `bsdcomp 0` to disable BSD-Compress compression entirely. If this option is read from a privileged source, a nonprivileged user may not specify a code size larger than the value from the privileged source.

`cdtrcts`
Use a non-standard hardware flow control such as DTR/CTS to control the flow of data on the serial port. If the `crtscts`, `nocrtscts`, `cdtrcts` or `nocdtrcts` option is not specified, the hardware flow control setting for the serial port is left unchanged. Some serial ports lack a true RTS output. Such serial ports use this mode to implement true bi-directional flow control. Note that this flow control mode does not permit using DTR as a modem control line.

`chap-interval` *n*
If this option is given, `pppd` will rechallenge the peer every *n* seconds.

`chap-max-challenge` *n*
Set the maximum number of CHAP challenge transmissions to *n* (default 10).

`chap-restart` *n*
Set the CHAP restart interval (retransmission timeout for challenges) to *n* seconds. The default is 3.

`connect-delay` *n*
Wait for up to *n* milliseconds after the connect script finishes for a valid PPP packet from the peer. When the wait period elapses or when a valid PPP packet is received from the peer, `pppd` begins negotiation by sending its first LCP packet. The default value is 1000 (1 second). A wait period applies only if the `connect` or `pty` option is used.

`datarate` *n*
Set maximum data rate to *n* (in bytes per second) when using the `pty`, `notty`, `record`, or `socket` options.

`debug`
Enables connection debugging facilities. If this option is given, `pppd` logs the contents of all control packets sent or received in a readable form. The packets are logged through syslog with facility `daemon` and level `debug`. This information can be directed to a file by configuring `/etc/syslog.conf` appropriately.

`default-asyncmap`
Disable `asyncmap` negotiation, forcing all control characters to be escaped for both the transmit and the receive direction.

`default-fcs`
Disable FCS Alternatives negotiation entirely. By default, no FCS Alternatives option is sent to the peer, but the option is accepted. If this option is specified by the peer, then LCP Configure-Reject is sent.

`default-mru`
Disable MRU [Maximum Receive Unit] negotiation. With this option, `pppd` uses the default MRU value of 1500 bytes for the transmit and receive directions.

`deflate` *nr,nt,e*
Request that the peer compress packets that it sends, using the `deflate` scheme, with a maximum window size of *2**nr* bytes, and agree to compress packets sent to the peer with a maximum window size of *2**nt* bytes and effort level of *e* (1 to 9). If *nt* is not specified, it defaults to the value given for *nr*. If *e* is not specified, it defaults to 6. Values in the range 9 to 15 may be used for *nr* and *nt*; larger values provide better compression but consume more kernel memory for compression dictionaries. (Value 8 is not permitted due to a zlib bug.) Alternatively, a value of 0 for *nr* or *nt* disables compression in the corresponding direction. Use `nodeflate` or `deflate 0` to disable `deflate` compression entirely. (Note: `pppd` requests deflate compression in preference to BSD-Compress if the peer can do either.) If this option is read from a privileged source, a nonprivileged user may not specify a code size larger than the value from the privileged source.

`demand`
Initiate the link only on demand, that is, when data traffic is present. With this option, the remote IP address must be specified by the user on the command line or in an options file. `pppd` initially configures and enables the interface for IP traffic without connecting to the peer. When traffic is available, `pppd` connects to the peer and performs negotiation, authentication and other actions. When completed, `pppd` passes data packets across the link. The `demand` option implies the `persist` option. If this behavior is not desired, use the `nopersist` option after the `demand` option. The `idle` and `holdoff` options can be used in conjunction with the `demand` option.

`domain` *d*
Append the domain name *d* to the local host name for authentication purposes. For example, if `gethostname()` returns the name `porsche`, but the fully qualified domain name is `porsche.Quotron.COM`, you could specify `domain Quotron.COM`. With this configuration, `pppd` uses the name `porsche.Quotron.COM` for accessing secrets in the secrets file and as the default name when authenticating to the peer. This option is privileged.

`endpoint` *endpoint-value*
Set the endpoint discriminator (normally used for RFC 1990 Multilink PPP operation). The *endpoint-value* consists of a class identifier and a class-dependent value. The class identifier is one of "null," "local," "IP," "MAC," "magic," "phone," or a decimal integer. If present, the class-dependent value is separated from the identifier by a colon (":") or period (".") . This value may be a standard dotted-decimal IP address for class "IP," an optionally colon-or-dot separated hex Ethernet address for class "MAC" (must have 6 numbers), or an arbitrary string of bytes specified in hex with optional colon or dot separators between bytes. Although this option is available, this implementation does not support multilink.

`fcs` *fcs-type*
Set FCS type(s) desired for data sent by the peer. The *fcs-type* is a comma-separated list of `crc16`, `crc32`, `null`, or integers. By default, an FCS Alternatives option is not specified, and the medium-dependent FCS type is used. If this option is specified and the peer sends an LCP Configure-Nak, only the listed types are used. If none are in common, the FCS Alternatives option is omitted from the next LCP Configure-Request to drop back to the default.

`hide-password`
When logging the contents of PAP packets, this option causes `pppd` to exclude the password string from the log. This is the default.

`holdoff` *n*
Specifies how many seconds to wait before re-initiating the link after it terminates. This option is effective only if the `persist` or `demand` option is used. The holdoff period is not applied if the link is terminated because it was idle.

`ident` *string*
Set the LCP Identification string. The default value is a version string similar to that displayed by the `--version` option.

`idle` *n*
Specifies that `pppd` must disconnect if the link is idle for *n* seconds. The link is idle when no data packets (i.e. IP packets) are being sent or received. Do not use this option with the `persist` option but without the `demand` option.

`ipcp-accept-local`
With this option, `pppd` accepts the peer's idea of the local IP address, even if the local IP address is specified in an option.

`ipcp-accept-remote`
With this option, `pppd` accepts the peer's idea of its remote IP address, even if the remote IP address is specified in an option.

`ipcp-max-configure` *n*
Set the maximum number of IPCP Configure-Request transmissions to *n* (default 10).

`ipcp-max-failure` *n*
Set the maximum number of IPCP Configure-NAKs sent before sending Configure-Rejects instead to *n* (default 10).

`ipcp-max-terminate` *n*
Set the maximum number of IPCP terminate-request transmissions to *n* (default 3).

`ipcp-restart` *n*
Set the IPCP restart interval (retransmission timeout) to *n* seconds (default 3).

`ipparam` *string*
Provides an extra parameter to the ip-up and ip-down scripts. When this option is given, the *string* supplied is given as the sixth parameter to those scripts. See the "Scripts" section.

`ipv6` <*local_interface_identifier*>,<*remote_interface_identifier*>
Set the local and/or remote 64-bit interface identifier. Either one may be omitted. The identifier must be specified in standard ASCII notation of IPv6 addresses (for example: `::dead:beef`). If the `ipv6cp-use-ipaddr` option is given, the local and remote identifiers are derived from the respective IPv4 addresses (see above). The `ipv6cp-use-persistent` option can be used instead of the `ipv6 <local>,<remote>` option.

`ipv6cp-accept-local`
Accept peer's interface identifier for the local link identifier.

`ipv6cp-max-configure` *n*
Set the maximum number of IPv6CP Configure-Request transmissions to *n* (default 10).

`ipv6cp-max-failure` *n*
Set the maximum number of IPv6CP Configure-NAKs sent before sending Configure-Rejects instead to *n* (default 10).

`ipv6cp-max-terminate` *n*
Set the maximum number of IPv6CP terminate-request transmissions to *n* (default 3).

`ipv6cp-restart` *n*
Set the IPv6CP restart interval (retransmission timeout) to *n* seconds (default 3).

`ipv6cp-use-ipaddr`
If either the local or remote IPv6 address is unspecified, use the corresponding configured IPv4 address as a default interface identifier. (This option uses the configured addresses, not the negotiated addresses. Do not use it with `ipcp-accept-local` if the local IPv6 identifier is unspecified or with `ipcp-accept-remote` if the remote IPv6 identifier is unspecified.)

`ipv6cp-use-persistent`
Use uniquely-available persistent value for link local address.

`kdebug` *n*
Enable debugging code in the kernel-level PPP driver. Argument *n* is the sum of the following values: `1` to enable general debug messages, `2` to request that contents of received

packets be printed, and 4 to request contents of transmitted packets be printed. Messages printed by the kernel are logged by `syslogd(1M)` to a file directed in the `/etc/syslog.conf` configuration file. Do not use the `kdebug` option to debug failed links. Use the `debug` option instead.

`lcp-echo-failure` *n*
If this option is given, `pppd` presumes the peer to be dead if *n* LCP Echo-Requests are sent without receiving a valid LCP Echo-Reply. If this happens, `pppd` terminates the connection. This option requires a non-zero value for the `lcp-echo-interval` parameter. This option enables `pppd` to terminate after the physical connection is broken (for example, if the modem has hung up) in situations where no hardware modem control lines are available.

`lcp-echo-interval` *n*
If this option is given, `pppd` sends an LCP Echo-Request frame to the peer every *n* seconds. Normally the peer responds to the Echo-Request by sending an Echo-Reply. This option can be used with the `lcp-echo-failure` option to detect that the peer is no longer connected.

`lcp-max-configure` *n*
Set the maximum number of LCP Configure-Request transmissions to *n* (default 10).

`lcp-max-failure` *n*
Set the maximum number of LCP Configure-NAKs sent before starting to send Configure-Rejects instead to *n* (default 10).

`lcp-max-terminate` *n*
Set the maximum number of LCP Terminate-Request transmissions to *n* (default 3).

`lcp-restart` *n*
Set the LCP restart interval (retransmission timeout) to *n* seconds (default 3).

`linkname` *name*
Sets the logical name of the link to *name*. `pppd` creates a file named `ppp-`*name*`.pid` in `/var/run` containing its process ID. This is useful in determining which instance of `pppd` is responsible for the link to a given peer system. This is a privileged option.

`local`
Do not use modem control lines. With this option, `pppd` ignores the state of the CD (Carrier Detect) signal from the modem and does not change the state of the DTR (Data Terminal Ready) signal.

`logfd` *n*
Send log messages to file descriptor *n*. `pppd` sends log messages to (at most) one file or file descriptor (as well as sending the log messages to syslog), so this option and the `logfile` option are mutually exclusive. By default `pppd` sends log messages to `stdout` (file descriptor 1) unless the serial port is open on stdout.

`logfile` *filename*
Append log messages to the file *filename* (and send the log messages to syslog). The file is opened in append mode with the privileges of the user who invoked `pppd`.

`login`
Use the system password database for authenticating the peer using PAP, and record the user in the system `wtmp` file. Note that the peer must have an entry in the `/etc/ppp/pap-secrets` file and the system password database to be allowed access.

`maxconnect` *n*
Terminate the connection after it has been available for network traffic for *n* seconds (that is, *n* seconds after the first network control protocol starts). An LCP Time-Remaining message is sent when the first NCP starts, and again when 5, 2, and 0.5 minutes are remaining.

`maxfail` *n*
Terminate after *n* consecutive failed connection attempts. A value of 0 means no limit. The default value is 10.

`modem`
Use the modem control lines. This option is the default. With this option, `pppd` waits for the CD (Carrier Detect) signal from the modem to be asserted when opening the serial device (unless a connect script is specified), and drops the DTR (Data Terminal Ready) signal briefly when the connection is terminated and before executing the connect script.

`ms-dns` <*addr*>
If `pppd` is acting as a server for Microsoft Windows clients, this option allows `pppd` to supply one or two DNS (Domain Name Server) addresses to the clients. The first instance of this option specifies the primary DNS address; the second instance (if given) specifies the secondary DNS address. If the first instance specifies a name that resolves to multiple IP addresses, then the first two addresses are used. (This option is present in some older versions of `pppd` under the name `dns-addr`.)

`ms-lanman`
If `pppd` connects as a client to a Microsoft server and uses MS-CHAPv1 for authentication, this option selects the LAN Manager password style instead of Microsoft NT.

`ms-wins` <*addr*>
If `pppd` acts as a server for Microsoft Windows or Samba clients, this option allows `pppd` to supply one or two WINS (Windows Internet Name Services) server addresses to the clients. The first instance of this option specifies the primary WINS address; the second instance (if given) specifies the secondary WINS address. As with `ms-dns`, if the name specified resolves to multiple IP addresses, then the first two will be taken as primary and secondary.

`name` *name*
Set the name of the local system for authentication purposes to *name*. This is a privileged option. With this option, `pppd` uses lines in the secrets files that have *name* as the second field to look for a secret to use in authenticating the peer. In addition, unless overridden with the `user` option, *name* is used as the name to send to the peer when authenticating the local system. (Note that `pppd` does not append the domain name to *name*.)

`no-accm-test`
Disable use of `asyncmap` (ACCM) checking using LCP Echo-Request messages. If the `lcp-echo-failure` is used on an asynchronous line, `pppd` includes all control characters in the first *n* LCP Echo-Request messages. If the `asyncmap` is set incorrectly, the link drops rather than continue operation with random failures. This option disables that feature.

`noaccomp`
Disable HDLC Address/Control compression in both directions (send and receive).

`noauth`
Do not require the peer to authenticate itself. This option is privileged.

`nobsdcomp`
Disables BSD-Compress compression; `pppd` will not request or agree to compress packets using the BSD-Compress scheme. This option is not necessary if `noccp` is specified.

`noccp`
Disable CCP (Compression Control Protocol) negotiation. This option should only be required if the peer has bugs or becomes confused by requests from `pppd` for CCP negotiation. If CCP is disabled, then BSD and deflate compression do not need to be separately disabled.

`nocrtscts`
Disable hardware flow control (i.e. RTS/CTS) on the serial port. If the `crtscts`, `nocrtscts`, `cdtrcts` or `nocdtrcts` options are not given, the hardware flow control setting for the serial port is left unchanged.

`nocdtrcts`
This option is a synonym for `nocrtscts`. Either option will disable both forms of hardware flow control.

`nodefaultroute`
Disable the `defaultroute` option. You can prevent non-root users from creating default routes with `pppd` by placing this option in the `/etc/ppp/options` file.

`nodeflate`
Disables deflate compression; `pppd` will not request or agree to compress packets using the deflate scheme. This option is not necessary if `noccp` is specified.

`nodeflatedraft`
Do not use Internet Draft (incorrectly assigned) algorithm number for deflate compression. This option is not necessary if `noccp` is specified.

`nodetach`
Do not detach from the controlling terminal. Without this option, `pppd` forks to become a background process if a serial device other than the terminal on the standard input is specified.

`noendpoint`
Do not send or accept the Multilink Endpoint Discriminator option.

`noident`
Disable use of LCP Identification. LCP Identification messages will not be sent to the peer, but received messages will be logged. (Specify this option twice to completely disable LCP Identification. In this case, `pppd` sends LCP Code-Reject in response to received LCP Identification messages.)

`noip`
Disable IPCP negotiation and IP communication. Use this option only if the peer has bugs or becomes confused by requests from `pppd` for IPCP negotiation.

`noipv6`
Disable IPv6CP negotiation and IPv6 communication. IPv6 is not enabled by default.

`noipdefault`
Disables the default behavior when no local IP address is specified, which is to determine (if possible) the local IP address from the hostname. With this option, the peer must supply the local IP address during IPCP negotiation (unless it specified explicitly on the command line or in an options file).

`nolog`
Do not send log messages to a file or file descriptor. This option cancels the `logfd` and `logfile` options. `nologfd` acts as an alias for this option.

`nomagic`
Disable magic number negotiation. With this option, `pppd` cannot detect a looped-back line. Use this option only if the peer has bugs. Do not use this option to work around the "Serial line is looped back" error message.

`nopam`
This privileged option disables use of pluggable authentication modules. If this option is specified, `pppd` reverts to standard authentication mechanisms. The default is not to use PAM.

`nopcomp`
Disable protocol field compression negotiation in the receive and the transmit direction.

`nopersist`
Exit once a connection has been made and terminated. This is the default unless the `persist` or `demand` option is specified.

`noplink`
Cause `pppd` to use I_LINK instead of I_PLINK. This is the default. When I_LINK is used, the system cleans up terminated interfaces (even when SIGKILL is used) but does not allow `ifconfig(1M)` to unplumb PPP streams or insert or remove modules dynamically. Use the `plink` option if `ifconfig(1M)` modinsert, modremove or unplumb support is needed.

`nopredictor1`
Do not accept or agree to Predictor-1 compression. (This option is accepted for compatibility. The implementation does not support Predictor-1 compression.)

`noproxyarp`
Disable the `proxyarp` option. If you want to prevent users from creating proxy ARP entries with `pppd`, place this option in the `/etc/ppp/options` file.

`notty`
Normally, `pppd` requires a terminal device. With this option, `pppd` allocates itself a pseudo-tty master/slave pair and uses the slave as its terminal device. `pppd` creates a child process to act as a character shunt to transfer characters between the pseudo-tty master and its standard input and output. Thus, `pppd` transmits characters on its standard output and receives characters on its standard input even if they are not terminal devices. This option increases the latency and CPU overhead of transferring data over the ppp interface as all of the characters sent and received must flow through the character shunt process. An explicit device name may not be given if this option is used.

`novj`
Disable Van Jacobson style TCP/IP header compression in both the transmit and the receive direction.

`novjccomp`
Disable the connection-ID compression option in Van Jacobson style TCP/IP header compression. With this option, `pppd` does not omit the connection-ID byte from Van Jacobson compressed TCP/IP headers, nor does it ask the peer to do so. This option is unnecessary if `novj` is specified.

`pam`
This privileged option enables use of PAM. If this is specified, `pppd` uses the `pam(3PAM)` framework for user authentication with a service name of "ppp" if the `login` option and PAP authentication are used. The default is not to use PAM.

`papcrypt`
Indicates that `pppd` should not accept a password which, before encryption, is identical to the secret from the `/etc/ppp/pap-secrets` file. Use this option if the secrets in the `pap-secrets` file are in `crypt(3C)` format.

`pap-max-authreq` *n*
Set the maximum number of PAP authenticate-request transmissions to *n* (default 10).

`pap-restart` *n*
Set the PAP restart interval (retransmission timeout) to *n* seconds (default 3).

`pap-timeout` *n*
Set the maximum time that `pppd` waits for the peer to authenticate itself with PAP to *n* seconds (0= no limit). The default is 30 seconds.

`password` *string*
Password string for authentication to the peer.

`persist`
Do not exit after a connection is terminated; instead try to reopen the connection.

`plink`
Cause `pppd` to use I_PLINK instead of I_LINK. The default is to use I_LINK, which cleans up terminated interface (even if SIGKILL is used), but does not allow `ifconfig(1M)` to unplumb PPP streams or insert or remove modules dynamically. Use this option if `ifconfig(1M)` modinsert/modremove/unplumb support is needed. See also the `plumbed` option.

`plugin` *filename*
Load the shared library object file *filename* as a plugin. This is a privileged option. Unless the filename specifies an explicit path, `/etc/ppp/plugins` and `/usr/lib/inet/ppp` will be searched for the object to load in that order.

`plumbed`
This option indicates that `pppd` should find a plumbed interface and use that for the session. If IPv4 addresses or IPv6 interface IDs or link MTU are otherwise unspecified, they are copied from the interface selected. This mode mimics some of the functionality of the older `aspppd` implementation and may be helpful when `pppd` is used with external applications that use `ifconfig(1M)`.

`pppmux` *timer*
Enable PPP Multiplexing option negotiation and set transmit multiplexing timeout to *timer* microseconds.

`privgroup` *group-name*
Allows members of group *group-name* to use privileged options. This is a privileged option. Because there is no guarantee that members of *group-name* cannot use `pppd` to become root themselves, you should be careful using this option. Consider it equivalent to putting the members of *group-name* in the `root` or `sys` group.

`proxyarp`
Add an entry to the system's Address Resolution Protocol (ARP) table with the IP address of the peer and the Ethernet address of this system. When you use this option, the peer appears to other systems to be on the local Ethernet. The remote address on the PPP link must be in the same subnet as assigned to an Ethernet interface.

`pty` *script*
Specifies that the command *script*, and not a specific terminal device is used for serial communication. `pppd` allocates itself a pseudo-tty master/slave pair and uses the slave as its terminal device. *script* runs in a child process with the pseudo-tty master as its standard input and output. An explicit device name may not be given if this option is used. (Note: if the `record` option is used in conjunction with the `pty` option, the child process will have pipes on its standard input and output.)

`receive-all`
With this option, `pppd` accepts all control characters from the peer, including those marked in the receive `asyncmap`. Without this option, `pppd` discards those characters as specified in *RFC 1662*. This option should be used only if the peer has bugs, as is often found with dial-back implementations.

`record` *filename*
Directs `pppd` to record all characters sent and received to a file named *filename*. *filename* is opened in append mode, using the user's user-ID and permissions. Because this option uses a pseudo-tty and a process to transfer characters between the pseudo-tty and the real serial device, it increases the latency and CPU overhead of transferring data over the PPP interface. Characters are stored in a tagged format with timestamps that can be displayed in readable form using the `pppdump`(1M) program. This option is generally used when debugging the kernel portion of `pppd` (especially CCP compression algorithms) and not for debugging link configuration problems. See the `debug` option.

`remotename` *name*
Set the assumed name of the remote system for authentication purposes to *name*. Microsoft WindowsNT does not provide a system name in its CHAP Challenge messages, and this option is often used to work around this problem.

`refuse-chap`
With this option, `pppd` will not agree to authenticate itself to the peer using standard Challenge Handshake Authentication Protocol (CHAP). (MS-CHAP is not affected.)

`refuse-mschap`
Do not agree to authenticate to peer with MS-CHAPv1. If this option is specified, requests for MS-CHAPv1 authentication from the peer are declined with LCP Configure-Nak. That option does not disable any other form of CHAP.

`refuse-mschapv2`
Do not agree to authenticate to peer with MS-CHAPv2. If specified, this option requests that MS-CHAPv2 authentication from the peer be declined with LCP Configure-Nak. That option does not disable any other form of CHAP.

`refuse-pap`
With this option, `pppd` will not agree to authenticate itself to the peer using Password Authentication Protocol (PAP).

`require-chap`
Require the peer to authenticate itself using standard CHAP authentication. MS-CHAP is not affected.

`require-mschap`
Require the peer to authenticate itself using MS-CHAPv1 authentication.

`require-mschapv2`
Require the peer to authenticate itself using MS-CHAPv2 authentication.

`require-pap`
Require the peer to authenticate itself using PAP authentication.

`show-password`
When logging contents of PAP packets, this option causes `pppd` to show the password string in the log message.

`silent`
With this option, `pppd` will not transmit LCP packets to initiate a connection until a valid LCP packet is received from the peer. This is like the "passive" option with older versions of `pppd` and is retained for compatibility, but the current `passive` option is preferred.

`small-accm-test`
When checking the `asyncmap` (ACCM) setting, `pppd` uses all 256 possible values by default. See `no-accm-test`. This option restricts the test so that only the 32 values affected by standard ACCM negotiation are tested. This option is useful on very slow links.

`socket` *host*:*port*
Connect to given host and port using TCP and run PPP over this connection.

`sync`
Use synchronous HDLC serial encoding instead of asynchronous. The device used by `pppd` with this option must have sync support. Currently supports `zs`, `se`, and `hsi` drivers.

`unit` *n*
Set PPP interface unit number to *n*, if possible.

`updetach`
With this option, `pppd` detaches from its controlling terminal after establishing the PPP connection. When this is specified, messages sent to `stderr` by the connect script, usually `chat(1M)`, and debugging messages from the debug option are directed to `pppd`'s standard output.

`usehostname`
Enforce the use of the hostname with domain name appended, if given, as the name of the local system for authentication purposes. This overrides the `name` option. Because the `name` option is privileged, this option is normally not needed.

`usepeerdns`
Ask the peer for up to two DNS server addresses. Addresses supplied by the peer, if any, are passed to the `/etc/ppp/ip-up` script in the environment variables DNS1 and DNS2. In addition, `pppd` creates an `/etc/ppp/resolv.conf` file containing one or two nameserver lines with the address(es) supplied by the peer.

`user` *name*
Sets the name used for authenticating the local system to the peer to *name*.

`vj-max-slots` *n*
Sets the number of connection slots to be used by the Van Jacobson TCP/IP header compression and decompression code to *n*, which must be between 2 and 16 (inclusive).

`welcome` *script*
Run the executable or shell command specified by *script* before initiating PPP negotiation, after the connect script, if any, has completed. A value for this option from a privileged source cannot be overridden by a non-privileged user.

`xonxoff`
Use software flow control, that is, XON/XOFF, to control the flow of data on the serial port.

Obsolete Options

The following options are obsolete:

`+ua` *name* Read a PAP user name and password from the file *name*. This file must have two lines for name and password. Name and password are sent to the peer when the peer requests PAP authentication.

`+ipv6` Enable IPv6 and IPv6CP without specifying interface identifiers.

`--version` Show version number and exit.

`--help` Show brief help message and exit.

Extended Description

The following sections discuss miscellaneous features of `pppd`:

Security

`pppd` allows system administrators to provide legitimate users with PPP access to a server machine without fear of compromising the security of the server or the network it runs on. Access control is provided by restricting IP addresses the peer may use based on its authenticated identity (if any), and through restrictions on options a non-privileged user may use. Options that permit potentially insecure configurations are privileged. Privileged options are accepted only in files that are under the control of the system administrator or when `pppd` is being run by root.

By default, `pppd` allows an unauthenticated peer to use a given IP address only if the system does not already have a route to that IP address. For example, a system with a permanent connection to the wider Internet will normally have a default route, meaning all peers must authenticate themselves to set up a connection. On such a system, the `auth` option is the default. Conversely, a system with a PPP link that comprises the only connection to the Internet probably does not possess a default route, so the peer can use virtually any IP address without authenticating itself.

Security-sensitive options are privileged and cannot be accessed by a non-privileged user running `pppd`, either on the command line, in the user's `$HOME/.pppprc` file, or in an options file read using the `file` option. Privileged options may be used in `/etc/ppp/options` file or in an options file read using the `call` option. If `pppd` is run by the root user, privileged options can be used without restriction. If the `/etc/ppp/options` file does not exist, then only root may invoke `pppd`. The `/etc/ppp/options` file must be created (but may be empty) to allow ordinary non-root users to access `pppd`.

When opening the device, `pppd` uses the invoking user's user ID or the root UID (that is, 0), depending if the device name was specified by the user or the system administrator. If the device name comes from a privileged source, that is, `/etc/ppp/options` or an options file read using the `call` option, `pppd` uses full root privileges when opening the device. Thus, by creating an appropriate file under `/etc/ppp/peers`, the system administrator can allow users

to establish a PPP connection via a device that they would not normally have access to. Otherwise pppd uses the invoking user's real UID when opening the device.

Authentication

During the authentication process, one peer convinces the other of its identity by sending its name and some secret information to the other. During authentication, the first peer becomes the "client" and the second becomes the "server." Authentication names can (but are not required to) correspond to the peer's Internet hostnames.

pppd supports four authentication protocols: the Password Authentication Protocol (PAP) and three forms of the Challenge Handshake Authentication Protocol (CHAP). With the PAP protocol, the client sends its name and a cleartext password to the server to authenticate itself. With CHAP, the server initiates the authentication exchange by sending a challenge to the client who must respond with its name and a hash value derived from the shared secret and the challenge.

The PPP protocol is symmetrical, meaning that each peer may be required to authenticate itself to the other. Different authentication protocols and names can be used for each exchange.

By default, pppd authenticates if requested and does not require authentication from the peer. However, pppd does not authenticate itself with a specific protocol if it has no secrets that can do so.

pppd stores authentication secrets in the `/etc/ppp/pap-secrets` (for PAP), and `/etc/ppp/chap-secrets` (for CHAP) files. Both files use the same format. pppd uses secrets files to authenticate itself to other systems and to authenticate other systems to itself.

Secrets files contain one secret per line. Secrets are specific to a particular combination of client and server and can only be used by that client to authenticate itself to that server. Each line in a secrets file has a minimum of three fields that contain the client and server names followed by the secret. Often, these three fields are followed by IP addresses that are used by clients to connect to a server.

A secrets file is parsed into words, with client name, server name and secrets fields allocated one word each. Embedded spaces or other special characters within a word must be quoted or escaped. Case is significant in all three fields.

A secret beginning with an at sign (“@”) is followed by the name of a file containing the secret. An asterisk (*) as the client or server name matches any name. When choosing a match, pppd selects the one with the fewest wildcards. Succeeding words on a line are interpreted by pppd as acceptable IP addresses for that client. IP Addresses are disallowed if they appear in lines that contain only three words or lines whose first word begins with a hyphen (“-”). To allow any address, use "*". An address starting with an exclamation point (”!”) indicates that the specified address is not acceptable. An address may be followed by "/" and a number *n* to indicate a whole subnet (all addresses that have the same value in the most significant *n* bits).

In this form, the address may be followed by a plus sign ("+") to indicate that one address from the subnet is authorized, based on the ppp network interface unit number in use. In this case, the host part of the address is set to the unit number, plus one.

When authenticating the peer, `pppd` chooses a secret with the peer's name in the first field of the secrets file and the name of the local system in the second field. The local system name defaults to the hostname, with the domain name appended if the `domain` option is used. The default can be overridden with the `name` option unless the `usehostname` option is used.

When authenticating to the peer, `pppd` first determines the name it will use to identify itself to the peer. This name is specified with the `user` option. If the `user` option is not used, the name defaults to the host name of the local system. `pppd` then selects a secret from the secrets file by searching for an entry with a local name in the first field and the peer's name in the second field. `pppd` will know the name of the peer if standard CHAP authentication is used because the peer will have sent it in the Challenge packet. However, if MS-CHAP or PAP is being used, `pppd` must determine the peer's name from the options specified by the user. The user can specify the peer's name directly with the `remotename` option. Otherwise, if the remote IP address was specified by a name, rather than in numeric form, that name will be used as the peer's name. If that fails, `pppd` uses the null string as the peer's name.

When authenticating the peer with PAP, the supplied password is compared with data in the secrets file. If the password and secret do not match, the password is encrypted using `crypt()` and checked against the secret again. If the `papcrypt` option is given, the first unencrypted comparison is omitted for better security, and entries must thus be in encrypted `crypt(3C)` form.

If the `login` option is specified, the username and password are also checked against the system password database. This allows you to set up the `pap-secrets` file to enable PPP access only to certain users, and to restrict the set of IP addresses available to users. Typically, when using the `login` option, the secret in `/etc/ppp/pap-secrets` would be "", which matches any password supplied by the peer. This makes having the same secret in two places unnecessary. When `login` is used, the `pam` option enables access control through `pam(3PAM)`.

Authentication must be completed before IPCP (or other network protocol) can be started. If the peer is required to authenticate itself and fails, `pppd` closes LCP and terminates the link. If IPCP negotiates an unacceptable IP address for the remote host, IPCP is closed. IP packets are sent or received only when IPCP is open.

To allow hosts that cannot authenticate themselves to connect and use one of a restricted set of IP addresses, add a line to the `pap-secrets` file specifying the empty string for the client name and secret.

Additional `pppd` options for a given peer may be specified by placing them at the end of the secrets entry, separated by two dashes (––). For example

```
peername servername secret ip-address -- novj
```

Routing When IPCP negotiation is complete, pppd informs the kernel of the local and remote IP addresses for the PPP interface and creates a host route to the remote end of the link that enables peers to exchange IP packets. Communication with other machines generally requires further modification to routing tables and/or Address Resolution Protocol (ARP) tables. In most cases the defaultroute and/or proxyarp options are sufficient for this, but further intervention may be necessary. If further intervention is required, use the /etc/ppp/ip-up script or a routing protocol daemon.

To add a default route through the remote host, use the defaultroute option. This option is typically used for "client" systems; that is, end-nodes that use the PPP link for access to the general Internet.

In some cases it is desirable to use proxy ARP, for example on a server machine connected to a LAN, to allow other hosts to communicate with the remote host. proxyarp instructs pppd to look for a network interface on the same subnet as the remote host. That is, an interface supporting broadcast and ARP that is not a point-to-point or loopback interface and that is currently up. If found, pppd creates a permanent, published ARP entry with the IP address of the remote host and the hardware address of the network interface.

When the demand option is used, the interface IP addresses are already set at the time when IPCP comes up. If pppd cannot negotiate the same addresses it used to configure the interface, it changes the interface IP addresses to the negotiated addresses. This may disrupt existing connections. Using demand dialing with peers that perform dynamic IP address assignment is not recommended.

Scripts pppd invokes scripts at various stages during processing that are used to perform site-specific ancillary processing. These scripts may be shell scripts or executable programs. pppd does not wait for the scripts to finish. The scripts are executed as root (with the real and effective user-id set to 0), enabling them to update routing tables, run privileged daemons, or perform other tasks. Be sure that the contents of these scripts do not compromise your system's security. pppd runs the scripts with standard input, output and error redirected to /dev/null, and with an environment that is empty except for some environment variables that give information about the link. The pppd environment variables are:

DEVICE	Name of the serial tty device.
IFNAME	Name of the network interface.
IPLOCAL	IP address for the link's local end. This is set only when IPCP has started.
IPREMOTE	IP address for the link's remote end. This is set only when IPCP has started.
PEERNAME	Authenticated name of the peer. This is set only if the peer authenticates itself.
SPEED	Baud rate of the tty device.
ORIG_UID	Real user-id of user who invoked pppd.

`PPPLOGNAME` Username of the real user-id who invoked `pppd`. This is always set.

`pppd` also sets the following variables for the ip-down and auth-down scripts:

`CONNECT_TIME` Number of seconds between the start of PPP negotiation and connection termination.

`BYTES_SENT` Number of bytes sent at the level of the serial port during the connection.

`BYTES_RCVD` Number of bytes received at the level of the serial port during the connection.

`LINKNAME` Logical name of the link, set with the `linkname` option.

If they exist, `pppd` invokes the following scripts. It is not an error if they do not exist.

`/etc/ppp/auth-up` Program or script executed after the remote system successfully authenticates itself. It is executed with five command-line arguments: `interface-name peer-name user-name tty-device speed`. Note that this script is not executed if the peer does not authenticate itself, for example, when the `noauth` option is used.

`/etc/ppp/auth-down` Program or script executed when the link goes down if `/etc/ppp/auth-up` was previously executed. It is executed in the same manner with the same parameters as `/etc/ppp/auth-up`.

`/etc/ppp/ip-up` A program or script that is executed when the link is available for sending and receiving IP packets (that is, IPCP has come up). It is executed with six command-line arguments: `interface-name tty-device speed local-IP-address remote-IP-address ipparam`.

`/etc/ppp/ip-down` A program or script which is executed when the link is no longer available for sending and receiving IP packets. This script can be used for undoing the effects of the `/etc/ppp/ip-up` script. It is invoked in the same manner and with the same parameters as the `ip-up` script.

`/etc/ppp/ipv6-up` Similar to `/etc/ppp/ip-up`, except that it is executed when the link is available for sending and receiving IPv6 packets. Executed with six command-line arguments: `interface-name tty-device speed local-link-local-address remote-link-local-address ipparam`.

`/etc/ppp/ipv6-down` Similar to `/etc/ppp/ip-down`, but executed when IPv6 packets can no longer be transmitted on the link. Executed with the same parameters as the ipv6-up script.

Examples

EXAMPLE 1 Using the auth Option

The following examples assume that the `/etc/ppp/options` file contains the `auth` option.

EXAMPLE 1 Using the auth Option *(Continued)*

`pppd` is commonly used to dial out to an ISP. You can do this using the "`pppd call isp`" command where the `/etc/ppp/peers/isp` file is set up to contain a line similar to the following:

```
cua/a 19200 crtscts connect '/usr/bin/chat -f /etc/ppp/chat-isp' noauth
```

For this example, `chat(1M)` is used to dial the ISP's modem and process any login sequence required. The `/etc/ppp/chat-isp` file is used by `chat` and could contain the following:

```
ABORT "NO CARRIER"
ABORT "NO DIALTONE"
ABORT "ERROR"
ABORT "NO ANSWER"
ABORT "BUSY"
ABORT "Username/Password Incorrect"
"" "at"
OK "at&f&d2&c1"
OK "atdt2468135"
"name:" "^Umyuserid"
"word:" "\qmypassword"
"ispts" "\q^Uppp"
"~-^Uppp-~"
```

See the `chat(1M)` man page for details of `chat` scripts.

EXAMPLE 2 Using `pppd` with `proxyarp`

`pppd` can also provide a dial-in ppp service for users. If the users already have login accounts, the simplest way to set up the ppp service is to let the users log in to their accounts and run `pppd` as shown in the following example:

```
example% pppd proxyarp
```

EXAMPLE 3 Providing a User with Access to PPP Facilities

To provide a user with access to the PPP facilities, allocate an IP address for the user's machine, create an entry in `/etc/ppp/pap-secrets` or `/etc/ppp/chap-secrets`. This enables the user's machine to authenticate itself. For example, to enable user "Joe" using machine "joespc" to dial in to machine "server" and use the IP address "joespc.my.net," add the following entry to the `/etc/ppp/pap-secrets` or `/etc/ppp/chap-secrets` files:

```
joespc    server    "joe's secret"    joespc.my.net
```

Alternatively, you can create another username, for example "ppp," whose login shell is `/usr/bin/pppd` and whose home directory is `/etc/ppp`. If you run `pppd` this way, add the options to the `/etc/ppp/.pppprc` file.

EXAMPLE 3 Providing a User with Access to PPP Facilities *(Continued)*

If your serial connection is complex, it may be useful to escape such control characters as XON (^Q) and XOFF (^S), using `asyncmap a0000`. If the path includes a telnet, escape ^] (`asyncmap 200a0000`). If the path includes a `rlogin` command, add `escape ff` option to the options, because `rlogin` removes the window-size-change sequence [0xff, 0xff, 0x73, 0x73, followed by any 8 bytes] from the stream.

Exit Status The `pppd` exit status indicates errors or specifies why a link was terminated. Exit status values are:

`0` `pppd` has detached or the connection was successfully established and terminated at the peer's request.

`1` An immediately fatal error occurred. For example, an essential system call failed.

`2` An error was detected in the options given. For example, two mutually exclusive options were used, or `/etc/ppp/options` is missing and the user is not root.

`3` `pppd` is not `setuid-root` and the invoking user is not root.

`4` The kernel does not support PPP. For example, the PPP kernel driver is not included or cannot be loaded.

`5` `pppd` terminated because it was sent a SIGINT, SIGTERM or SIGHUP signal.

`6` The serial port could not be locked.

`7` The serial port could not be opened.

`8` The connect script failed and returned a non-zero exit status.

`9` The command specified as the argument to the `pty` option could not be run.

`10` The PPP negotiation failed because no network protocols were able to run.

`11` The peer system failed or refused to authenticate itself.

`12` The link was established successfully, but terminated because it was idle.

`13` The link was established successfully, but terminated because the connect time limit was reached.

`14` Callback was negotiated and an incoming call should arrive shortly.

`15` The link was terminated because the peer is not responding to echo requests.

`16` The link was terminated by the modem hanging up.

`17` The PPP negotiation failed because serial loopback was detected.

`18` The init script failed because a non-zero exit status was returned.

`19` Authentication to the peer failed.

Files

`/var/run/spppn.pid`	Process-ID for `pppd` process on PPP interface unit *n*.
`/var/run/ppp-name.pid`	Process-ID for `pppd` process for logical link name (see the `linkname` option).
`/etc/ppp/pap-secrets`	Usernames, passwords and IP addresses for PAP authentication. This file should be owned by root and not readable or writable by any other user, otherwise `pppd` will log a warning.
`/etc/ppp/chap-secrets`	Names, secrets and IP addresses for all forms of CHAP authentication. The `/etc/ppp/pap-secrets` file should be owned by `root` should not readable or writable by any other user, otherwise, `pppd` will log a warning.
`/etc/ppp/options`	System default options for `pppd`, read before user default options or command-line options.
`$HOME/.ppprc`	User default options, read before `/etc/ppp/options.`*ttyname*.
`/etc/ppp/options.`*ttyname*	System default options for the serial port in use; read after `$HOME/.ppprc`. The *ttyname* component of this filename is formed when the initial `/dev/` is stripped from the port name (if present), and slashes (if any) are converted to dots.
`/etc/ppp/peers`	Directory with options files that may contain privileged options, even if `pppd` was invoked by a user other than `root`. The system administrator can create options files in this directory to permit non-privileged users to dial out without requiring the peer to authenticate, but only to certain trusted peers.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/network/ppp
Interface Stability	Committed

See Also `chat(1M)`, `ifconfig(1M)`, `crypt(3C)`, `pam(3PAM)`, `attributes(5)`

Haskin, D., Allen, E. *RFC 2472 – IP Version 6 Over PPP*. Network Working Group. December 1998.

Jacobson, V. *RFC 1144, Compressing TCP/IP Headers for Low-Speed Serial Links*. Network Working Group. February, 1990

Lloyd, B., Simpson, W. *RFC 1334, PPP Authentication Protocols*. Network Working Group. October 1992.

McGregor, G. *RFC 1332, The PPP Internet Protocol Control Protocol (IPCP)*. Network Working Group. May 1992.

Rivest, R. *RFC 1321, The MD5 Message-Digest Algorithm*. Network Working Group. April 1992

Simpson, W. *RFC 1661, The Point-to-Point Protocol (PPP)*. Network Working Group. July 1994.

Simpson, W. *RFC 1662, HDLC-like Framing* . Network Working Group. July 1994.

Notes These signals affect `pppd` behavior:

`SIGINT, SIGTERM`	Terminate the link, restore the serial device settings and exit.
`SIGHUP`	Terminate the link, restore the serial device settings and close the serial device. If the `persist` or `demand` option is specified, `pppd` attempts to reopen the serial device and start another connection after the holdoff period. Otherwise `pppd` exits. If received during the holdoff period, `SIGHUP` causes `pppd` to end the holdoff period immediately.
`SIGUSR1`	Toggles the state of the `debug` option and prints link status information to the log.
`SIGUSR2`	Causes `pppd` to renegotiate compression. This is useful to re-enable compression after it has been disabled as a result of a fatal decompression error. (Fatal decompression errors generally indicate a bug in an implementation.)

Diagnostics Messages are sent to the syslog daemon using facility `LOG_DAEMON`. To see error and debug messages, edit the `/etc/syslog.conf` file to direct the messages to the desired output device or file, or use the `updetach` or `logfile` options.

The `debug` option causes the contents of all LCP, PAP, CHAP or IPCP control packets sent or received to be logged. This is useful if PPP negotiation does not succeed or if authentication fails.

Debugging can also be enabled or disabled by sending a `SIGUSR1` signal, which acts as a toggle to the `pppd` process.

Name pppoec – PPPoE chat utility

Synopsis

```
pppoec [-omillisecs] [-smillisecs] [-v] device
     [service [ [except]server... [only]]]

pppoec [-omillisecs] [-v] -i [device]
```

Description The `pppoec` utility implements the client-side negotiation of PPPoE. It is intended to be used with the `pppd(1M)` `connect` option, in the same manner as the `chat(1M)` utility is used for asynchronous dial-up PPP.

When given with the `-i` flag, `pppoec` sends out a broadcast query on the given interface named by the *device* parameter. You can specify no other arguments in this mode. All responding PPPoE servers and the offered services are displayed on standard output.

Otherwise, when given without the `-i` flag, `pppoec` does the full PPPoE client-side negotiation. The *device* parameter is the intended Ethernet interface, and must already be plumbed with `sppptun(1M)`. The optional *service* parameter specifies a particular service desired; other offered services will be ignored. The optional *server* parameter specifies a specific server desired. You can specify *server* as an Ethernet address in the usual x:x:x:x:x:x format (with "*" in any of the six byte positions interpreted to mean "any"), or as a symbolic name resolved through `/etc/ethers` (or NIS), or as a PPPoE access concentrator name. The sense of the match (true or false) can be inverted by specifying the keyword `except` before this string. This parameter can be specified more than once, and the first match is taken.

If you specify the *server* parameter, then the selected servers become "preferred." If no preferred server responds, then the first responding server is used instead. To exclude non-matching servers entirely, append the keyword `only`.

Options The following options are supported:

`-i` Sends out broadcast query over interface specified by *device*.

`-o` Sets the initial wait time in milliseconds for PADO from the server before PADI is retried. The default is 500 milliseconds for normal operation, or 3000 milliseconds (3 seconds) for inquiry (`-i`) mode.

`-s` Sets the initial wait time in milliseconds for PADS from the server before PADR is retried. The default is 2000 milliseconds (2 seconds).

`-v` Displays verbose progress messages, including all PPPoE messages sent, and all state machine transitions.

You normally do not need to adjust the parameters set with `-o` and `-s`. They are provided for coping with unusually slow servers.

Operands The following operands are supported:

device plumbed Ethernet interface

server preferred server or, if you specify `only`, the specified server

service desired service; other available services are ignored

Examples

EXAMPLE 1 Connecting to Any Service on hme0

The following command enables you to connect to any PPPoE service on `hme0`:

```
# /usr/bin/pppd sppptun plugin pppoe.so \
connect "/usr/lib/inet/pppoec hme0" debug
```

Often, a command such as the preceding is specified in an `/etc/ppp/peers` file instead. For example, enter the following in `/etc/ppp/peers/myisp`:

```
sppptun
plugin pppoe.so
connect "/usr/lib/inet/pppoec hme0"
debug
```

To invoke the PPP connection described in the file, enter:

```
% /usr/bin/pppd call myisp
```

Note that, because the `/etc/ppp/peers` files are considered privileged by `pppd`, you need not be root to invoke the preceding command.

EXAMPLE 2 Connecting to a Particular Service

A more complex example: on `hme0`, connect to only the `internet` service offered by PPPoE servers with access concentrator name `isp`, but not to any Ethernet addresses starting with `40:0:1a`.

```
# /usr/lib/inet/pppoec hme0 internet except 40:0:1a:*:*:* isp only
```

Note that the `except 40:0:1a:*:*:*` filter must come before `isp`, because the filters are first-match.

Exit Status

The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

Files

`/usr/lib/inet/pppoec`	executable command
`/dev/sppptun`	Solaris PPP tunneling device driver.
`/etc/ppp/connect-errors`	usual location of error output (see DIAGNOSTICS, below)

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/network/ppp/tunnel

See Also pppd(1M), sppptun(1M), pppoed(1M), sppptun(7M)

RFC 2516, Method for Transmitting PPP Over Ethernet (PPPoE), Mamakos et al, February 1999

Diagnostics Error messages are written to standard error, which is normally redirected by pppd to /etc/ppp/connect-errors. The errors can also be redirected to pppd's standard output by using the updetach option.

If you specify the -v, verbose progress messages are displayed, including all PPPoE messages sent, and all state machine transitions. Specifying the updetach or nodetach pppd option is helpful when using verbose mode.

Name pppoed – PPPoE server daemon

Synopsis `ppoed [`*options*`]`

Description The `pppoed` daemon implements the server-side negotiation of PPPoE. When a client requests service from this daemon, a copy of `pppd(1M)` is invoked to handle the actual PPP communication.

At startup, options are read from the command line and the `/etc/ppp/pppoe` file. After these options have been read, options in the per-device `/etc/ppp/pppoe.`*device* files are read, using the device names specified on the command line or in `/etc/ppp/pppoe`. Device names are not permitted in the per-device files. It is not an error if any of these files are absent; missing files are ignored.

Options are reread in the same order on `SIGHUP`. Except for the possibility of short delays due to the processing time, `SIGHUP` does not interfere with any client operations. Current status, including options read, is dumped to `/tmp/pppoed.`*pid* on `SIGINT`.

The options are used to set up a list of services to be offered to PPPoE clients on the broadcast domains (Ethernet subnets) specified by the named devices. Option parsing is always in one of two modes, either global mode or service mode. The initial mode at the beginning of each file (and the command line) is global mode. Options specified in global mode serve as default values for subsequently defined services. Service mode is entered by the `service` *name* option. In this mode, the named option is defined. Options that appear in this mode override any global mode definitions for the current service.

The option parsing follows standard shell tokenizing rules, using whitespace to delimit tokens, quotes to enclose strings that can contain whitespace, and escape sequences for special characters. Environment variables are substituted using familiar `$VAR` and `${VAR}` syntax and set using `NEWVAR=`*string*. Variables are both usable in subsequent options and provided to the `pppd(1M)` processes spawned for each client, but they are interpreted as they are encountered during option processing. Thus, all set variables are seen by all processes spawned; position in the configuration files has no effect on this.

Options The `pppoed` daemon supports the following options:

`client [except]` *client-list* This option restricts the clients that may receive the service. If the `except` keyword is given, then the clients on the list cannot access the service, but others can. If this keyword is not given, then only the listed clients can access the service.

This option can be specified more than once for a given service. For a given client, first match among all listed options encountered specifies the handling. If it matches an option with `except` specified, then access is denied. Otherwise, it is granted. The `client` list within a service is prepended to any list specified in the global context.

If no `client` options are given or if all options are specified with `except`, then all clients are permitted by default. If any `client` options without `except` are specified, then no clients are permitted by default.

The *client-list* is a comma-separated list of client identifiers. The match is made if any client on the list matches; thus, these are logically "ORed" together. Each client identifier can be either a symbolic name (resolved through `/etc/ethers` or NIS, as defined by `/etc/nsswitch.conf`) or a hexadecimal Ethernet address in the format `x:x:x:x:x:x`. In the latter case, any byte of the address can be "*", which matches any value in that position. For example, `40:0:1a:*:*:*` matches Ethernet adapters from the manufacturer assigned block `40:0:1a`.

`debug`

Increase debug logging detail level by one. The detail levels are 0 (no logging), 1 (errors only; the default), 2 (warnings), 3 (informational messages), and 4 (debug messages). Log messages are written by default to `syslog(3C)` using facility *daemon* (see the `log` option below). When specified on the command line or in the global context of the `/etc/ppp/pppoe` file, this option also sets the daemon's default (non-service-related) detail level.

`device` *device-list*

Specify the devices on which the service is available. The *device-list* is a comma-separated list of logical device names (without the leading `/dev/`), such as `hme0`. This option is ignored if encountered in the per-device `/etc/ppp/pppoe.`*device* files.

`extra` *string*

Specifies extra options to `pppd(1M)`. It defaults to "`plugin pppoe.so directtty`" and usually does not need to be overridden.

`file` *path*

Suspends parsing of the current file, returns to global mode, and reads options from *path*. This file must be present and readable; if it is not, an error is logged. When the end of that file is reached, processing returns to the current file and the mode is reset to global again.

The global mode options specified in files read by this command use the options set in the current file's global mode; this condition extends to any file included by those files. All files read are parsed as though the command line had specified this option, and thus inherit the command line's global modes.

This option can be used to revert to global mode at any point in an option file by specifying `file /dev/null`.

`group` *name*
: Specifies the group ID (symbolic or numeric) under which `pppd` is executed. If `pppoed` is not run as root, this option is ignored.

`log` *path*
: Specifies an alternate debug logging file. Debug messages are sent to this file instead of `syslog`. The special name `syslog` is recognized to switch logging back to `syslog`. When specified on the command line or in the global context of the `/etc/ppp/pppoe` file, this option also sets the daemon's default (non-service-related) log file.

`nodebug`
: Set debug logging detail level to 0 (no logging). When specified on the command line or in the global context of the `/etc/ppp/pppoe` file, this option also sets the daemon's default (non-service-related) detail level.

`nowildcard`
: Specifies that the current service should not be included in response to clients requesting "any" service. The client must ask for this service by name. When specified on the command line or in the global context of the `/etc/ppp/pppoe` file, this option causes `pppoed` to ignore all wildcard service requests.

`path` *path*
: Specifies the path to the `pppd` executable. Defaults to `/usr/bin/pppd`.

`pppd` *string*
: Passes command-line arguments to `pppd`. It can be used to set the IP addresses or configure security for the session. The default value is the empty string.

`server` *string*
: Specifies the PPPoE Access Concentrator name to be sent to the client. It defaults to "Solaris PPPoE".

`service` *name*
: Closes any service being defined and begins definition of a new service. The same service name can be used without conflict on multiple devices. If the same service name is used on a single device, then the last definition encountered during parsing overrides all previous definitions.

`user` *name*
: Specifies the user ID, symbolic or numeric, under which `pppd` is executed. If `pppoed` is not run as root, this option is ignored.

`wildcard`
: Specifies that the service should be included in responses to client queries that request "any" service, which is done by requesting a service name of length zero. When specified on the command line or in the global context of the

/etc/ppp/pppoe file, this option causes pppoed to ignore all wildcard service requests. This is the default.

Examples

EXAMPLE 1 Configuring for Particular Services

In the /etc/ppp/pppoe file:

```
service internet
    device $DEV
    pppd "proxyarp 192.168.1.1:"
service debugging
    device hme0,$DEV
    pppd "debug proxyarp 192.168.1.1:"
```

You then invoke the daemon with:

```
example% /usr/lib/inet/pppoed DEV=eri0
```

The lines in /etc/ppp/pppoe and the preceding command result in offering services "internet" and "debugging" (and responding to wildcard queries) on interface eri0, and offering only service "debugging" on interface hme0.

Signals

The pppoed daemon responds to the following signals:

SIGHUP — Causes pppoed to reparse the original command line and all configuration files, and close and reopen any log files.

SIGINT — Causes a snapshot of the state of the pppoed daemon to be written to /tmp/pppoed.*pid* (where *pid* is the decimal process ID of the daemon).

Files

/usr/lib/inet/pppoed	executable command
/dev/sppptun	Solaris PPP tunneling device driver
/etc/ppp/pppoe	main configuration option file
/etc/ppp/pppoe.*device*	per–device configuration option file
/etc/ppp/pppoe-errors	location of output from pppd's stderr
/etc/ppp/pppoe.if	list of Ethernet interfaces to be plumbed at boot time
/tmp/pppoed.*pid*	ASCII text file containing dumped pppoed state information

Attributes

See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/network/ppp/tunnel

See Also pppd(1M), pppoec(1M), sppptun(1M), sppptun(7M)

Mamakos, L., et al. *RFC 2516, A Method for Transmitting PPP Over Ethernet (PPPoE)*. Network Working Group. February 1999

Notes Because `pppd` is installed setuid root, this daemon need not be run as root. However, if it is not run as root, the `user` and `group` options are ignored.

The Ethernet interfaces to be used must be plumbed for PPPoE using the sppptun(1M) utility before services can be offered.

The daemon operate runs even if there are no services to offer. If you want to modify a configuration, it is not necessary to terminate the daemon. Simply use `pkill -HUP pppoed` after updating the configuration files.

The PPPoE protocol is far from perfect. Because it runs directly over Ethernet, there is no possibility of security and the MTU is limited to 1492 (violating RFC 1661's default value of 1500). It is also not possible to run the client and the server of a given session on a single machine with a single Ethernet interface for testing purposes. The client and server portions of a single session must be run on separate Ethernet interfaces with different MAC addresses.

Name pppstats – print PPP statistics

Synopsis

```
pppstats [-a] [-v] [-r] [-z] [-c <count>] [-w <secs>]
     [interface]
```

Description The `pppstats` utility reports PPP-related statistics at regular intervals for the specified PPP interface. If the interface is unspecified, `pppstats` defaults to `sppp0`. The display is split horizontally into input and output sections containing columns of statistics describing the properties and volume of packets received and transmitted by the interface.

Options The `pppstats` options are:

`-a` Display absolute values rather than deltas. With this option, all reports show statistics for the time elapsed since the link was initiated. Without this option, the second and subsequent reports show statistics for the time since the last report.

`-c` *count* Repeat the display *count* times. If this option is not specified, the default repeat count is 1 if the `-w` option is not specified, otherwise infinity.

`-r` Display additional statistics summarizing the compression ratio achieved by the packet compression algorithm in use.

`-v` Display additional statistics relating to the performance of the Van Jacobson TCP header compression algorithm.

`-w` *wait* Pause *wait* seconds between each display. If this option is not specified, the default interval is five seconds.

`-z` Instead of the standard display, show statistics indicating the performance of the packet compression algorithm in use.

Extended Description The following fields are printed on the input side when the `-z` option is not used:

`IN` Total number of bytes received by this interface.

`PACK` Total number of packets received by this interface.

`VJCOMP` Number of header-compressed TCP packets received by this interface.

`VJUNC` Number of header-uncompressed TCP packets received by this interface. Not reported when the `-r` option is specified.

`VJERR` Number of corrupted or bogus header-compressed TCP packets received by this interface. Not reported when the `-r` option is specified.

`VJTOSS` Number of VJ header-compressed TCP packets dropped on reception by this interface because of preceding errors. Only reported when the `-v` option is specified.

`NON-VJ` Total number of non-TCP packets received by this interface. Only reported when the `-v` option is specified.

RATIO Compression ratio achieved for received packets by the packet compression scheme in use, defined as the uncompressed size divided by the compressed size. Only reported when the -r option is specified.

UBYTE Total number of bytes received, after decompression of compressed packets. Only reported when the -r option is specified.

The following fields are printed on the output side:

OUT Total number of bytes transmitted from this interface.

PACK Total number of packets transmitted from this interface.

VJCOMP Number of TCP packets transmitted from this interface with VJ-compressed TCP headers.

VJUNC Number of TCP packets transmitted from this interface with VJ-uncompressed TCP headers. Not reported when the -r option is specified.

NON-VJ Total number of non-TCP packets transmitted from this interface. Not reported when the -r option is specified.

VJSRCH Number of searches for the cached header entry for a VJ header compressed TCP packet. Only reported when the -v option is specified.

VJMISS Number of failed searches for the cached header entry for a VJ header compressed TCP packet. Only reported when the -v option is specified.

RATIO Compression ratio achieved for transmitted packets by the packet compression scheme in use, defined as the size before compression divided by the compressed size. Only reported when the -r option is specified.

UBYTE Total number of bytes to be transmitted before packet compression is applied. Only reported when the -r option is specified.

When the -z option is specified, pppstats displays the following fields relating to the packet compression algorithm currently in use. If packet compression is not in use, these fields display zeroes. The fields displayed on the input side are:

COMPRESSED BYTE Number of bytes of compressed packets received.

COMPRESSED PACK Number of compressed packets received.

INCOMPRESSIBLE BYTE Number of bytes of incompressible packets (that is, those which were transmitted in uncompressed form) received.

INCOMPRESSIBLE PACK Number of incompressible packets received.

COMP RATIO Recent compression ratio for incoming packets, defined as the uncompressed size divided by the compressed size (including both compressible and incompressible packets).

The fields displayed on the output side are:

`COMPRESSED BYTE`	Number of bytes of compressed packets transmitted.
`COMPRESSED PACK`	Number of compressed packets transmitted.
`INCOMPRESSIBLE BYTE`	Number of bytes of incompressible packets received; that is, those that were transmitted by the peer in uncompressed form.
`INCOMPRESSIBLE PACK`	Number of incompressible packets transmitted.
`COMP RATIO`	Recent compression ratio for outgoing packets.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/network/ppp
Interface Stability	Committed

See Also `pppd(1M)`, `attributes(5)`

Name praudit – print contents of an audit trail file

Synopsis

```
praudit [-lrsx] [-ddel] [filename]...
```

Description `praudit` reads the listed *filename*s (or standard input, if no *filename* is specified) and interprets the data as audit trail records as defined in `audit.log(4)`. By default, times, user and group IDs (UIDs and GIDs, respectively) are converted to their ASCII representation. Record type and event fields are converted to their ASCII representation. A maximum of 100 audit files can be specified on the command line.

Options The following options are supported:

`-d`*del*
: Use *del* as the field delimiter instead of the default delimiter, which is the comma. If *del* has special meaning for the shell, it must be quoted. The maximum size of a delimiter is three characters. The delimiter is not meaningful and is not used when the `-x` option is specified.

`-l`
: Print one line per record.

`-r`
: Print records in their raw form. Times, UIDs, GIDs, record types, and events are displayed as integers. This option is useful when naming services are offline. The `-r` option and the `-s` option are exclusive. If both are used, a format usage error message is output.

`-s`
: Display records in their short form. Numeric fields' ASCII equivalents are looked up by means of the sources specified in the `/etc/nsswitch.conf` file (see `nsswitch.conf(4)`). All numeric fields are converted to ASCII and then displayed. The short ASCII representations for the record type and event fields are used. This option and the `-r` option are exclusive. If both are used, a format usage error message is output.

`-x`
: Print records in XML form. Tags are included in the output to identify tokens and fields within tokens. Output begins with a valid XML prolog, which includes identification of the DTD which can be used to parse the XML.

Files

`/etc/security/audit_event`
: Audit event definition and class mappings.

`/etc/security/audit_class`
: Audit class definitions.

`/usr/share/lib/xml/dtd`
: Directory containing the verisioned DTD file referenced in XML output, for example, `adt_record.dtd.1`.

`/usr/share/lib/xml/style`
: Directory containing the versioned XSL file referenced in XML output, for example, `adt_record.xsl.1`.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os
Interface Stability	See below.

The command stability is evolving. The output format is unstable.

See Also getent(1M), getpwuid(3C), gethostbyaddr(3NSL), ethers(3SOCKET), getipnodebyaddr(3SOCKET), audit.log(4), audit_class(4), audit_event(4), group(4), nsswitch.conf(4), passwd(4), attributes(5)

See the section on Auditing in *Oracle Solaris 11.1 Administration: Security Services*.

Name projadd – administer a new project on the system

Synopsis

```
projadd [-n] [-f filename] [-p projid [-o]] [-c comment]
     [-U user [,user]... ] [-G group [,group]... ]
     [ [-K name [=value [,value]...]...]] project
```

Description `projadd` adds a new project entry to the `/etc/project` file. If the files backend is being used for the project database, the new project is available for use immediately upon the completion of the `projadd` command.

Options The following options are supported:

`-c` *comment*
: Add a project comment. Comments are stored in the project's entry in the `/etc/project` file. Generally, comments contain a short description of the project and are used as the field for the project's full name.

  Specify *comment* as a text string. *comment* cannot contain a colon (`:`) or NEWLINE.

`-f` *filename*
: Specify the project file to modify. If no *filename* is specified, the system project file, `/etc/project`, is modified.

`-G` *group*[,group...]
: Specify a group list for the project.

`-K` *name*[=*value*[,*value*...]
: Specify an attribute list for the project. Multiple `-K` options can be specified to set values on multiple keys, such as:

  `-K` *key1*=*value1* `-K "`*key2*`=(`*value2a*`),(`*value2b*`)"`

  Resource control attributes use parentheses to specify values for a key. Because many user shells interpret parentheses as special characters, it is best to enclose an argument to `-K` that contains parentheses with double quotes, as shown above and in EXAMPLES, below. See `resource_controls(5)` for a description of the resource controls you can specify for a project.

`-n`
: Syntax check. Check the format of the existing system project file and modifications only. The contents of the existing project file, such as user names, group names, and resources that are specified in the project attributes are not checked.

`-o`
: This option allows the project ID specified by the `-p` option to be non-unique within the project file.

`-p` *projid*
: Set the project ID of the new project.

  Specify *projid* as a non-negative decimal integer below `UID_MAX` as defined in `limits.h`. *projid* defaults to the next available

unique number above the highest number currently assigned. For example, if *projid*s `100`, `105`, and `200` are assigned, the next default *projid* is `201`. *projid*s between `0`-`99` are reserved by SunOS.

`-U` *user*[,user...] Specify a user list for the project.

Operands The following operands are supported:

project The name of the project to create. The *project* operand is a string consisting of characters from the set of alphabetic characters, numeric characters, underline (_), and hyphen (-). The period ('.') is reserved for projects with special meaning to the operating system. The first character of the project name must be a letter. An error message is displayed if these restrictions are not met.

Examples **EXAMPLE 1** Adding a Project

The following command creates the project `salesaudit` and sets the resource controls specified as arguments to the `-K` option.

```
projadd -p 111 -G sales,finance -c "Auditing Project" \
   -K "rcap.max-rss=10GB" \
   -K "process.max-file-size=(priv,50MB,deny)" \
   -K "task.max-lwps=(priv,100,deny)" salesaudit
```

This command would produce the following entry in `/etc/project`:

```
salesaudit:111:Auditing Project::sales,finance: \
process.max-file-size=(priv,52428800,deny); \
rcap.max-rss=10737418240;task.max-lwps=(priv,100,deny)
```

Note that the preceding would appear as one line in `/etc/project`.

Comparing the `projadd` command and resulting output in `/etc/project`, note the effect of the scaling factor in the resource cap (`rcap.max-rss=10GB`) and the resource control (`process.max-file-size=(priv,50MB,deny)`). Modifiers, such as B, KB, and MB, and scaling factors are specified in `resource_controls(5)`.

Exit Status The following exit values are returned:

`0` Successful completion.

`2` The command syntax was invalid. A usage message for `projadd` is displayed.

`3` An invalid argument was provided to an option.

`4` The *projid* given with the `-p` option is already in use.

`5` The project files contain an error. See `project(4)`.

`6` The project to be added, group, user, or resource does not exist.

9 The project is already in use.

10 Cannot update the `/etc/project` file.

Files `/etc/project` System project file

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os
Interface Stability	See below.

Invocation is evolving. Human readable output is unstable.

See Also `projects(1)`, `groupadd(1M)`, `groupdel(1M)`, `groupmod(1M)`, `grpck(1M)`, `projdel(1M)`, `projmod(1M)`, `useradd(1M)`, `userdel(1M)`, `usermod(1M)`, `project(4)`, `attributes(5)`, `resource_controls(5)`

Notes In case of an error, `projadd` prints an error message and exits with a non-zero status.

`projadd` adds a project definition only on the local system. If a network name service such as NIS or LDAP is being used to supplement the local `/etc/project` file with additional entries, `projadd` cannot change information supplied by the network name service.

Name projdel – delete a project from the system

Synopsis

```
projdel [-f filename] project
```

Description The `projdel` utility deletes a project from the system and makes the appropriate changes to the system file.

Options The following options are supported:

`-f` *filename* Specify the project file to modify. If no *filename* is specified, the system project file, `/etc/project`, is modified.

Operands The following operands are supported:

project The name of the project to be deleted.

Exit Status The following exit values are returned:

`0` Successful completion.

`2` The command syntax was invalid. A usage message for `projdel` is displayed.

`3` An invalid argument was provided to an option.

`4` The *projid* given with the `-p` option is already in use.

`5` The project files contain an error. See `project(4)`.

`6` The project to be modified, group, user, or resource does not exist.

`9` The project is already in use.

`10` Cannot update the `/etc/project` file.

Files `/etc/project` System project file

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os
Interface stabilty	See below.

Invocation is evolving. Human readable output is unstable.

See Also `projects(1)`, `groupadd(1M)`, `groupdel(1M)`, `groupmod(1M)`, `grpck(1M)`, `logins(1M)`, `projadd(1M)`, `projmod(1M)`, `useradd(1M)`, `userdel(1M)`, `usermod(1M)`, `project(4)`, `attributes(5)`

Diagnostics In case of an error, `projdel` prints an error message and exits with a non-zero status.

Notes `projdel` deletes a project definition only on the local system. If a network name service such as NIS or LDAP is being used to supplement the local `/etc/project` file with additional entries, `projdel` cannot change information supplied by the network name service.

Name projmod – modify a project's information on the system

Synopsis

```
projmod [-n] [-A|-f filename | -]

projmod [-n] [-A|-f filename | -] [-p projid [-o]]
     [-c comment] [-a|-s|-r] [-U user [,user]... ]
     [-G group [,group]... ]
     [ [-K name [=value [,value]...]...]]
     [-l new_projectname] project
```

Description The `projmod` utility modifies a project's definition on the system. `projmod` changes the definition of the specified project and makes the appropriate project-related system file and file system changes.

Options The following options are supported:

`-A` Apply the project's resource controls, as defined in the system's project database, to the project if it is active.

`-a` Specify that the users, groups, attributes, or attribute values specified by the `-U`, `-G` or `-K` options should be added to the project, rather than replacing the existing member or attribute list.

`-c` *comment* Specify *comment* as a text string. Generally, *comment* contains a short description of the project. This information is stored in the project's `/etc/project` entry.

`-f` *filename* | – Specify the project file to modify or validate or specify input from stdin for validation. As noted under OPERANDS, if you do not specify a project in a `projmod` command line, `projmod` validates the argument to `-f`. If you do not use this option, the system project file, `/etc/project`, is modified.

`-G` *group* [,*group*...] Specify a replacement list of member groups of the project. When used in conjunction with the `-a` or `-r` options, this option specifies a list of groups to be added or removed from the project.

`-K` *name*[=*value*[,*value*...] Specify a replacement list of project attributes for the project. When used in conjunction with the `-a`, `-r`, or `-s` options, this option specifies a list of attribute values to be added, removed, or replaced in the project. Attributes must be delimited by semicolons (`;`). Multiple `-K` options can be specified to set, add, remove, or substitute values on multiple keys, such as:

`-K` *key1=value1* `-K "`*key2=(value2a),(value2b)*`"`

Resource control attributes use parentheses to specify values for a key. Because many user shells interpret parentheses as special

characters, it is best to enclose an argument to `-K` that contains parentheses with double quotes, as shown above and in EXAMPLES, below. See `resource_controls(5)` for a description of the resource controls you can specify for a project.

`-l` *new_projectname*
: Specify the new project name for the project. The *new_projectname* argument is a string consisting of characters from the set of alphabetic characters, numeric characters, period (`.`), underline (_), and hyphen (`-`). The first character should be alphabetic. An error message is written if these restrictions are not met. The project name must also be unique within the project file.

`-n`
: Syntax check. Check the format of the existing system project file and modifications only. The contents of the existing project file, such as user names, group names, and resources that are specified in the project attributes are not checked.

`-o`
: This option allows the project ID specified by the `-p` option to be non-unique within the project file.

`-p` *projid*
: Specify a new project ID for the project. It must be a non-negative decimal integer less than MAXUID as defined in `param.h`. This value must be unique within the project file if the `-o` option is not specified.

`-r`
: Specify that the users, groups, attributes, or attribute values specified by the `-U`, `-G` or `-K` options should be removed from the project, rather than replacing the existing member or attribute list.

`-s`
: Specify that the list of attributes specified by the `-K` option should have their values replaced. If the attributes do not exist, they are added as if the a option was used. This option has no effect the `-U` or `-G` options.

`-U` *user* [,*user*...]
: Specify a replacement list of member users of the project. When used in conjunction with the `-a` or `-r` options, this option specifies a list of users to be added or removed from the project.

Operands The following operands are supported:

project
: An existing project name to be modified or displayed.

(none)
: If no operand is given, the project file is validated without modifying any project.

Examples

EXAMPLE 1 Using the -K Option for Addition of an Attribute Value

Consider the following project(4) entry:

```
salesaudit:111:Auditing Project::sales,finance: \
  process.max-file-size=(priv,52428800,deny); \
  task.max-lwps=(priv,100,deny)
```

The preceding would appear as one line in `/etc/project`. For this and the following examples, the focus is on the attributes field in the `project` entry. That is, the last field, the field following the last semicolon.

The attributes field for the project `salesaudit` lists the following resource control:

```
task.max-lwps=(priv,1000,signal=KILL)
```

The following `projmod` command adds an action clause to the preceding entry:

```
# projmod -a -K "task.max-lwps=(priv,100,deny)" salesaudit
```

...with the resulting attributes field in the entry for `salesaudit`:

```
task.max-lwps=(priv,100,deny),(priv,1000,signal=KILL)
```

EXAMPLE 2 Using the -K Option for the Substitution of an Attribute Value

Assume an attributes field in a project(4) entry for the project `salesaudit` that lists the following resource control:

```
task.max-lwps=(priv,100,deny),(priv,1000,signal=KILL)
```

The following `projmod` command substitutes the action clause specified in the command for the action clauses in the preceding entry:

```
# projmod -s -K "task.max-lwps=(priv,500,signal=SIGSTOP)" salesaudit
```

...with the resulting attributes field in the entry for `salesaudit`:

```
task.max-lwps=(priv,500,signal=SIGSTOP)
```

EXAMPLE 3 Using the -K Option for Removal of an Attribute Value

Assume an attributes field in a project(4) entry for a project `salesaudit` that lists the following resource control:

```
task.max-lwps=(priv,100,deny),(priv,1000,signal=KILL)
```

The following `projmod` command removes the first action clause from the preceding entry:

```
# projmod -r -K "task.max-lwps=(priv,100,deny)" salesaudit
```

...with the resulting attributes field in the entry for `salesaudit`:

EXAMPLE 3 Using the -K Option for Removal of an Attribute Value *(Continued)*

```
task.max-lwps=(priv,1000,signal=KILL)
```

EXAMPLE 4 Specifying Multiple Attribute Values

Suppose you want to achieve the following resource controls for the project `salesaudit`:

```
task.max-lwps=(priv,100,deny)
process.max-file-size=(priv,50MB,deny)
```

The following `projmod` command adds these resource controls for `salesaudit`:

```
# projmod -a -K "task.max-lwps=(priv,100,deny)" \
-K "process.max-file-size=(priv,50MB,deny)" salesaudit
```

...with the resulting attributes field in the entry for `salesaudit`:

```
task.max-lwps=(priv,100,deny);process.max-file-size=(priv,52428800,deny)
```

In this example, note the effect of the use of the modifier and scaling factor for the resource control `process.max-file-size`. The specification in `projmod`:

```
"process.max-file-size=(priv,50MB,deny)"
```

...becomes, in `/etc/project`:

```
process.max-file-size=(priv,52428800,deny)
```

That is, `50MB` is expanded to `52428800`. The modifiers, such as MB, and scaling factors you can use for resource controls are specified in `resource_controls(5)`.

EXAMPLE 5 Binding a Pool to a Project

The following command sets the `project.pool` attribute for the project `sales`.

```
# projmod -a -K project.pool=salespool sales
```

EXAMPLE 6 Evaluating Input from stdin

The following command uses the `-f` option without a project name operand to evaluate the contents of an NIS projects map.

```
# ypcat project | projmod -f -
```

Exit Status

In case of an error, `projmod` prints an error message and exits with one of the following values:

The following exit values are returned:

`0` Successful completion.

2 The command syntax was invalid. A usage message for `projmod` is displayed.

3 An invalid argument was provided to an option.

4 The *projid* given with the `-p` option is already in use.

5 The project files contain an error. See `project(4)`.

6 The project to be modified, group, user, or resource does not exist.

9 The project is already in use.

10 Cannot update the `/etc/project` file.

Files

`/etc/group` System file containing group definitions

`/etc/project` System project file

`/etc/passwd` System password file

`/etc/shadow` System file containing users' encrypted passwords and related information

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os
Interface Stability	See below.

Invocation is evolving. Human readable output is unstable.

See Also

`groupadd(1M)`, `groupdel(1M)`, `groupmod(1M)`, `projadd(1M)`, `projdel(1M)`, `useradd(1M)`, `userdel(1M)`, `usermod(1M)`, `passwd(4)`, `project(4)`, `attributes(5)`, `resource_controls(5)`

Notes

The `projmod` utility modifies project definitions only in the local `/etc/project` file. If a network name service such as NIS or LDAP is being used to supplement the local files with additional entries, `projmod` cannot change information supplied by the network name service. However `projmod` verifies the uniqueness of project name and project ID against the external name service.

Name prstat – report active process statistics

Synopsis

```
prstat [-acHJLmRrtTv] [-d u | d] [-C psrsetlist] [-h lgrplist]
     [-j projlist] [-k tasklist] [-n ntop[,nbottom]]
     [-p pidlist] [-P cpulist] [-s key | -S key ]
     [-u euidlist] [-U uidlist] [-z zoneidlist] [-Z]
     [interval [count]]
```

Description The `prstat` utility iteratively examines all active processes on the system and reports statistics based on the selected output mode and sort order. `prstat` provides options to examine only processes matching specified PIDs, UIDs, zone IDs, CPU IDs, and processor set IDs.

The `-j`, `-k`, `-C`, `-p`, `-P`, `-u`, `-U`, and `-z` options accept lists as arguments. Items in a list can be either separated by commas or enclosed in quotes and separated by commas or spaces.

If you do not specify an option, `prstat` examines all processes and reports statistics sorted by CPU usage.

Options The following options are supported:

`-a`
Report information about processes and users. In this mode `prstat` displays separate reports about processes and users at the same time.

`-c`
Print new reports below previous reports instead of overprinting them.

`-C` *psrsetlist*
Report only processes or lwps that are bound to processor sets in the given list. Each processor set is identified by an integer as reported by `psrset(1M)`. The load averages displayed are the sum of the load averages of the specified processor sets (see `pset_getloadavg(3C)`). Processes with one or more LWPs bound to processor sets in the given list are reported even when the `-L` option is not used.

`-d u | d`
Specify `u` for a printed representation of the internal representation of time. See `time(2)`. Specify `d` for standard date format. See `date(1)`.

`-h` *lgrplist*
Report only processes or lwps whose home *lgroup* is in the given list of *lgroups*. No processes or lwps will be listed for invalid *lgroups*.

`-H`
Report information about home *lgroup*. In this mode, `prstat` adds an extra column showing process or lwps home *lgroup* with the header LGRP.

`-j` *projlist*
Report only processes or lwps whose project ID is in the given list. Each project ID can be specified as either a project name or a numerical project ID. See `project(4)`.

`-J`
Report information about processes and projects. In this mode `prstat` displays separate reports about processes and projects at the same time.

`-k` *tasklist*
Report only processes or lwps whose task ID is in *tasklist*.

`-L`
Report statistics for each light-weight process (LWP). By default, `prstat` reports only the number of LWPs for each process.

`-m`
Report microstate process accounting information. In addition to all fields listed in `-v` mode, this mode also includes the percentage of time the process has spent processing system traps, text page faults, data page faults, waiting for user locks and waiting for CPU (latency time).

`-n` *ntop*[,*nbottom*]
Restrict number of output lines. The *ntop* argument determines how many lines of process or `lwp` statistics are reported, and the *nbottom* argument determines how many lines of user, task, or projects statistics are reported if the `-a`, `-t`, `-T`, `-J` or `-Z` options are specified. By default, `prstat` displays as many lines of output that fit in a window or terminal. When you specify the `-c` option or direct the output to a file, the default values for `ntop` and `nbottom` are `15` and `5`.

`-p` *pidlist*
Report only processes whose process ID is in the given list.

`-P` *cpulist*
Report only processes or lwps which have most recently executed on a CPU in the given list. Each CPU is identified by an integer as reported by `psrinfo(1M)`.

`-R`
Put `prstat` in the real time scheduling class. When this option is used, `prstat` is given priority over time-sharing and interactive processes. This option is available only for superuser.

`-r`
Disable lookups for user names and project names. (Note that this does not apply to lookups for the `-j`, `-u`, or `-U` options.)

`-s` *key*
Sort output lines (that is, processes, `lwps`, or users) by *key* in descending order. Only one *key* can be used as an argument.

There are five possible key values:

cpu
Sort by process CPU usage. This is the default.

pri
Sort by process priority.

rss
Sort by resident set size.

size
Sort by size of process image.

time
Sort by process execution time.

`-S` *key*
Sort output lines by *key* in ascending order. Possible *key* values are the same as for the `-s` option. See `-s`.

`-t`
Report total usage summary for each user. The summary includes the total number of processes or LWPs owned by the user, total size of process images, total resident set size, total cpu time, and percentages of recent cpu time and system memory.

`-T`
Report information about processes and tasks. In this mode `prstat` displays separate reports about processes and tasks at the same time.

`-u` *euidlist*
Report only processes whose effective user ID is in the given list. Each user ID may be specified as either a login name or a numerical user ID.

`-U` *uidlis*t
Report only processes whose real user ID is in the given list. Each user ID may be specified as either a login name or a numerical user ID.

`-v`
Report verbose process usage. This output format includes the percentage of time the process has spent in user mode, in system mode, and sleeping. It also includes the number of voluntary and involuntary context switches, system calls and the number of signals received. Statistics that are not reported are marked with the `-` sign.

`-z` *zoneidlist*
Report only processes or LWPs whose zone ID is in the given list. Each zone ID can be specified as either a zone name or a numerical zone ID. See `zones(5)`.

`-Z`
Report information about processes and zones. In this mode, `prstat` displays separate reports about processes and zones at the same time.

Output The following list defines the column headings and the meanings of a `prstat` report:

PID
The process ID of the process.

USERNAME
The real user (login) name or real user ID.

SWAP
The sum of swap reservations of the associated processes for each user, project, task, or zone. This counts shared memory only once for each user, project, task, or zone. Swap is reserved when anonymous memory is allocated or files are mapped private. The value of swap is expressed in kilobytes (`K`), megabytes (`M`), or gigabytes (`G`).

RSS
The resident set size of the process (RSS), in kilobytes (K), megabytes (M), or gigabytes (G). The RSS value is an estimate provided by `proc(4)` that might underestimate the actual resident set size. Users who want to get more accurate usage information for capacity planning should use the `-x` option to `pmap(1)` instead.

STATE
The state of the process:

cpu*N*
Process is running on CPU *N*.

sleep
Sleeping: process is waiting for an event to complete.

wait
Waiting: process is waiting for CPU usage to drop to the CPU-caps enforced limits. See the description of `CPU-caps` in `resource_controls(5)`.

run
Runnable: process in on run queue.

zombie
Zombie state: process terminated and parent not waiting.

stop
Process is stopped.

PRI
The priority of the process. Larger numbers mean higher priority.

NICE
Nice value used in priority computation. Only processes in certain scheduling classes have a nice value.

TIME
The cumulative execution time for the process.

CPU
The percentage of recent CPU time used by the process. If executing in a non-global `zone` and the pools facility is active, the percentage will be that of the processors in the processor set in use by the pool to which the `zone` is bound.

PROCESS
The name of the process (name of executed file).

LWPID
The `lwp` ID of the `lwp` being reported.

NLWP
The number of `lwps` in the process.

With the some options, in addition to a number of the column headings shown above, there are:

NPROC
Number of processes in a specified collection.

MEMORY
Percentage of memory used by a specified collection of processes.

The following columns are displayed when the `-v` or `-m` option is specified

USR
The percentage of time the process has spent in user mode.

SYS
The percentage of time the process has spent in system mode.

TRP
The percentage of time the process has spent in processing system traps.

TFL
The percentage of time the process has spent processing text page faults.

DFL
The percentage of time the process has spent processing data page faults.

LCK
The percentage of time the process has spent waiting for user locks.

SLP
The percentage of time the process has spent sleeping.

LAT
The percentage of time the process has spent waiting for CPU.

VCX
The number of voluntary context switches.

ICX
The number of involuntary context switches.

SCL
The number of system calls.

SIG
The number of signals received.

Under the `-L` option, one line is printed for each `lwp` in the process and some reporting fields show the values for the `lwp`, not the process.

The following column is displayed when the `-H` option is specified:

LGRP
The home *lgroup* of the process or lwp.

Operands The following operands are supported:

count
Specifies the number of times that the statistics are repeated. By default, `prstat` reports statistics until a termination signal is received.

interval
Specifies the sampling interval in seconds; the default interval is `5` seconds.

Examples **EXAMPLE 1** Reporting the Five Most Active Super-User Processes

The following command reports the five most active super-user processes running on `CPU1` and `CPU2`:

```
example% prstat -u root -n 5 -P 1,2 1 1

PID   USERNAME  SWAP   RSS STATE  PRI  NICE      TIME  CPU PROCESS/LWP
 306   root     3024K 1448K sleep   58    0   0:00.00 0.3% sendmail/1
 102   root     1600K  592K sleep   59    0   0:00.00 0.1% in.rdisc/1
 250   root     1000K  552K sleep   58    0   0:00.00 0.0% utmpd/1
 288   root     1720K 1032K sleep   58    0   0:00.00 0.0% sac/1
   1   root      744K  168K sleep   58    0   0:00.00 0.0% init/1
TOTAL:       25, load averages:  0.05, 0.08, 0.12
```

EXAMPLE 2 Displaying Verbose Process Usage Information

The following command displays verbose process usage information about processes with lowest resident set sizes owned by users `root` and `john`.

```
example% prstat -S rss -n 5 -vc -u root,john

 PID USERNAME USR SYS TRP TFL DFL LCK SLP LAT VCX ICX SCL SIG PROCESS/LWP
   1 root     0.0 0.0  -   -   -   -  100  -    0   0   0   0 init/1
 102 root     0.0 0.0  -   -   -   -  100  -    0   0   3   0 in.rdisc/1
 250 root     0.0 0.0  -   -   -   -  100  -    0   0   0   0 utmpd/1
1185 john    0.0 0.0  -   -   -   -  100  -    0   0   0   0 csh/1
 240 root     0.0 0.0  -   -   -   -  100  -    0   0   0   0 switchd/4
 TOTAL:       71, load averages:  0.02, 0.04, 0.08
```

Exit Status The following exit values are returned:

`0`
Successful completion.

`1`
An error occurred.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also `date(1)`, `lgrpinfo(1)`, `plgrp(1)`, `proc(1)`, `ps(1)`, `time(2)`, `psrinfo(1M)`, `psrset(1M)`, `sar(1M)`, `pset_getloadavg(3C)`, `proc(4)`, `project(4)`, `attributes(5)`, `resource_controls(5)`, `zones(5)`

Notes The snapshot of system usage displayed by `prstat` is true only for a split-second, and it may not be accurate by the time it is displayed. When the `-m` option is specified, `prstat` tries to turn on microstate accounting for each process; the original state is restored when `prstat` exits. See `proc(4)` for additional information about the microstate accounting facility.

The total memory size reported in the SWAP and RSS columns for groups of processes can sometimes overestimate the actual amount of memory used by processes with shared memory segments.

Name prtconf – print system configuration

Synopsis

```
/usr/sbin/prtconf [-V] | [-F] | [-x] | [-bpv] | [-acdDlPuv]
     [dev_path]
```

Description The `prtconf` command prints the system configuration information. The output includes the total amount of memory, and the configuration of system peripherals formatted as a device tree.

If a device path is specified on the command line for those command options that can take a device path, `prtconf` will only display information for that device node.

Options The following options are supported:

`-a` Display all the ancestors device nodes, up to the root node of the device tree, for the device specified on the command line.

`-b` Display the firmware device tree root properties for the purpose of platform identification. These properties are “name”, “compatible”, “banner-name” and “model”.

`-c` Display the device subtree rooted at the device node specified on the command line, that is, display all the children of the device node specified on the command line.

`-d` Display vendor ID and device ID for PCI and PCI Express devices, in addition to the nodename.

`-D` For each system peripheral in the device tree, displays the name of the device driver used to manage the peripheral.

`-l` Show the `/dev/chassis` location associated with the device node. If `-v` is used, `-l` is implied.

`-F` Returns the device path name of the console frame buffer, if one exists. If there is no frame buffer, `prtconf` returns a non-zero exit code. This flag must be used by itself. It returns only the name of the console, frame buffer device or a non-zero exit code. For example, if the console frame buffer on a SUNW,Ultra-30 is `ffb`, the command returns: `/SUNW,ffb@1e,0:ffb0`. This option could be used to create a symlink for `/dev/fb` to the actual console device.

`-p` Displays information derived from the device tree provided by the firmware (PROM) on SPARC platforms or the booting system on x86 platforms.The device tree information displayed using this option is a snapshot of the initial configuration and may not accurately reflect reconfiguration events that occur later.

`-P` Includes information about pseudo devices. By default, information regarding pseudo devices is omitted.

`-u` Together with `-v`, displays information for each device listing properties from the vendor and admin lists, if any.

`-v` Specifies verbose mode.

`-V` Displays platform-dependent PROM (on SPARC platforms) or booting system (on x86 platforms) version information. This flag must be used by itself. The output is a string. The format of the string is arbitrary and platform-dependent.

`-x` Reports if the firmware on this system is 64-bit ready. Some existing platforms may need a firmware upgrade in order to run the 64-bit kernel. If the operation is not applicable to this platform or the firmware is already 64-bit ready, it exits silently with a return code of zero. If the operation is applicable to this platform and the firmware is not 64-bit ready, it displays a descriptive message on the standard output and exits with a non-zero return code. The hardware platform documentation contains more information about the platforms that may need a firmware upgrade in order to run the 64-bit kernel.

This flag overrides all other flags and must be used by itself.

Operands The following operands are supported:

dev_path The path to a target device minor node, device nexus node, or device link for which device node configuration information is displayed

Exit Status The following exit values are returned:

`0` No error occurred.

`non-zero` With the `-F` option (SPARC only), a non-zero return value means that the output device is not a frame buffer. With the `-x` option, a non-zero return value means that the firmware is not 64–bit ready. In all other cases, a non-zero return value means that an error occurred.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os
Interface Stability	Uncommitted

See Also `fuser(1M)`, `modinfo(1M)`, `sysdef(1M)`, `driver(4)`, `driver.conf(4)`, `attributes(5)`

Sun Hardware Platform Guide

SPARC Only `openprom(7D)`

Notes The output of the `prtconf` command is highly dependent on the version of the PROM installed in the system. The output will be affected in potentially all circumstances.

The `driver not attached` message means that no driver is currently attached to that instance of the device. In general, drivers are loaded and installed (and attached to hardware instances) on demand, and when needed, and may be uninstalled and unloaded when the device is not in use.

On x86 platforms, the use of `prtconf -vp` provides a subset of information from `prtconf -v`. The value of integer properties from `prtconf -vp` might require byte swapping for correct interpretation.

Name prtdiag – display system diagnostic information

Synopsis

```
/usr/sbin/prtdiag [-v] [-l]
```

Description `prtdiag` displays system configuration and diagnostic information on `sun4u`, `sun4v`, and x86 systems.

The diagnostic information lists any failed field replaceable units (FRUs) in the system.

The interface, output, and location in the directory hierarchy for `prtdiag` are uncommitted and subject to change in future releases.

`prtdiag` does not display diagnostic information and environmental status when executed on the Sun Enterprise 10000 server. See the `/var/opt/SUNWssp/adm/${SUNW_HOSTNAME}/messages` file on the system service processor (SSP) to obtain such information for this server.

Options The following options are supported:

`-l` Log output. If failures or errors exist in the system, output this information to `syslogd(1M)` only.

`-v` Verbose mode. Displays the time of the most recent AC Power failure, and the most recent hardware fatal error information, and (if applicable) environmental status. The hardware fatal error information is useful to repair and manufacturing for detailed diagnostics of FRUs.

Exit Status The following exit values are returned:

`0` No failures or errors are detected in the system.

`1` Failures or errors are detected in the system.

`2` An internal `prtdiag` error occurred, for example, out of memory.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/library/platform
Interface Stability	Uncommitted*

*The output is unstable.

See Also `modinfo(1M)`, `prtconf(1M)`, `psrinfo(1M)`, `sysdef(1M)`, `syslogd(1M)`, `attributes(5)`, `openprom(7D)`

Notes Not all diagnostic and system information is available on every Solaris platform, and therefore cannot be displayed by `prtdiag`. On those platforms, further information can be obtained from the System Controller.

Name prtdscp – display DSCP IP addresses

Synopsis

```
prtdscp [-v ]

prtdscp [-v ] -h

prtdscp [-v ] -d

prtdscp [-v ] -s
```

Description `prtdscp` displays the IP addresses associated with a Domain to Service Processor Communications Protocol (DSCP) link. If no arguments are specified, `prtdscp` displays the IP addresses on both ends of the DSCP link. The IP address of either the Service Processor or domain side can be displayed separately by the use of the `-s` or `-d` options, respectively.

Options The following options are supported:

`-v` Verbose mode. Print additional details about the program's internal progress to `stderr`.

`-h` Help. Print a brief synopsis of the program's usage and exit. All other command line arguments are ignored.

`-d` Display only the local domain's IP address.

`-s` Display only the remote Service Processor's IP address.

Examples **EXAMPLE 1** Displaying both addresses

The following example displays both the local domain's IP address and the remote SP's IP address:

```
# prtdscp
Domain Address: 192.168.103.2
SP Address: 192.168.103.1
```

EXAMPLE 2 Displaying the local IP address

The following example displays the local domain's IP address:

```
# prtdscp -d
192.168.103.2
```

EXAMPLE 3 Displaying the remote IP address

The following example display the remote SP's IP address:

```
# prtdscp -s
192.168.103.1
```

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/domain-service-processor-protocol/sparc-enterprise, SUNWdscpu.u
Interface Stability	Committed

See Also attributes(5)

Name prtfru – print FRUID-specific information about the FRUs on a system or domain

Synopsis

```
/usr/sbin/prtfru [-d] | [-clx] [container]
```

Description The `prtfru` utility is used to obtain FRUID data from the system or domain. Its output is that of a tree structure echoing the path in the FRU (Field-Replaceable Unit) tree to each container. When a container is found, the data from that container is printed in a tree-like structure as well.

`prtfru` without any arguments will print the FRU hierarchy and all of the FRUID container data. `prtfru` prints to stdout which may be redirected to a file.

Options The following options are supported:

`-c` Prints *only* the containers and their data. This option does not print the FRU tree hierarchy.

`-d` Prints a DTD for the current registry to stdout.

`-l` Prints *only* the FRU tree hierarchy. This option does not print the container data.

`-x` Prints in XML format with a system identifier (SYSTEM) of `prtfrureg.dtd`.

Options `-c` and `-l` can be used together to obtain a list of the containers.

Operands The folowing operand is supported:

container The name of a particular container in the FRU hierarchy, that is, either the name or path/name of a container as displayed in the `-l` option.

Exit Status The following exit values are returned:

`0` All information was found and printed successfully.

`>0` An error has occurred.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/fru-id

See Also `fruadm(1M)`, `attributes(5)`

Name prtpicl – print PICL tree

Synopsis

```
/usr/sbin/prtpicl [-c picl_class] [-v]
```

Description The `prtpicl` command prints the PICL tree maintained by the PICL daemon. The output of `prtpicl` includes the name and PICL class of the nodes.

Options The following options are supported:

`-c` *picl_class* Print only the nodes of the named PICL class.

`-v` Print in verbose mode. In verbose mode, `prtpicl` prints a list of properties and values for each node. Verbose mode is disabled by default.

Exit Status The following exit values are returned:

`0` Successful completion.

`non-zero` An error occurred.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/picl

See Also `picld(1M)`, `attributes(5)`

Name prtvtoc – report information about a disk geometry and partitioning

Synopsis `prtvtoc [-fhs] [-t` *vfstab*`] [-m` *mnttab*`]` *device*

Description The `prtvtoc` command allows the contents of the label to be viewed. The command can be used only by the super-user.

The *device* name can be the file name of a raw device in the form of `/dev/rdsk/c?t?d?s2` or can be the file name of a block device in the form of `/dev/dsk/c?t?d?s2`.

Options The following options are supported:

`-f` Report on the disk free space, including the starting block address of the free space, number of blocks, and unused partitions.

`-h` Omit the headers from the normal output.

`-m` *mnttab* Use *mnttab* as the list of mounted filesystems, in place of `/etc/mnttab`.

`-s` Omit all headers but the column header from the normal output.

`-t` *vfstab* Use *vfstab* as the list of filesystem defaults, in place of `/etc/vfstab`.

Examples **EXAMPLE 1** Using the `prtvtoc` Command

The following example uses the `prtvtoc` command on a 424-megabyte hard disk:

```
example# prtvtoc /dev/rdsk/c0t3d0s2
* /dev/rdsk/c0t3d0s2 partition map
*
* Dimension:
*     512 bytes/sector
*      80 sectors/track
*       9 tracks/cylinder
*     720 sectors/cylinder
*    2500 cylinders
*    1151 accessible cylinders
*
* Flags:
*   1: unmountable
*  10: read-only
* *                          First    Sector   Last
* Partition   Tag   Flags   Sector   Count    Sector   Mount Directory
     0         2     00          0    76320    76319   /
     1         3     01      76320   132480   208799
     2         5     00          0   828720   828719
     5         6     00     208800   131760   340559   /opt
     6         4     00     340560   447120   787679   /usr
     7         8     00     787680    41040   828719   /export/home
example#
```

EXAMPLE 1 Using the `prtvtoc` Command *(Continued)*

The data in the `Tag` column above indicates the type of partition, as follows:

Name	*Number*
UNASSIGNED	0x00
BOOT	0x01
ROOT	0x02
SWAP	0x03
USR	0x04
BACKUP	0x05
STAND	0x06
VAR	0x07
HOME	0x08
ALTSCTR	0x09
CACHE	0x0a
RESERVED	0x0b
SYSTEM	0x0c
BOOT	0x18

The data in the `Flags` column above indicates how the partition is to be mounted, as follows:

Name	*Number*
MOUNTABLE, READ AND WRITE	0x00
NOT MOUNTABLE	0x01
MOUNTABLE, READ ONLY	0x10

EXAMPLE 2 Using the `prtvtoc` Command with the `-f` Option

The following example uses the `prtvtoc` command with the `-f` option on a 424-megabyte hard disk:

```
example# prtvtoc -f /dev/rdsk/c0t3d0s2
FREE_START=0 FREE_SIZE=0 FREE_COUNT=0 FREE_PART=34
```

EXAMPLE 3 Using the `prtvtoc` Command on a Disk Over One Terabyte

The following example uses uses the `prtvtoc` command on a disk over one terabyte:.

```
example# prtvtoc /dev/rdsk/c1t1d0s2
* /dev/rdsk/c1t1d0s2 partition map
*
* Dimensions:
*     512 bytes/sector
* 3187630080 sectors
* 3187630013 accessible sectors
*
* Flags:
*   1: unmountable
*  10: read-only
*
*                          First     Sector    Last
* Partition  Tag  Flags    Sector     Count    Sector  Mount Directory
0      2    00         34    262144    262177
1      3    01     262178    262144    524321
6      4    00     524322 3187089340 3187613661
8     11    00  3187613662     16384 318763004
```

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also `devinfo(1M)`, `fmthard(1M)`, `format(1M)`, `mount(1M)`, `attributes(5)`

Warnings The `mount` command does not check the “not mountable” bit.

Name psradm – change processor operational status

Synopsis

```
psradm -f | -i | -n | -s [-v] [-F] processor_id

psradm -a -f | -i | -n | -s [-v] [-F]
```

Description The `psradm` utility changes the operational status of processors. The legal states for the processor are `on-line`, `off-line`, `spare`, `faulted`, and `no-intr`.

An `on-line` processor processes LWPs (lightweight processes) and can be interrupted by I/O devices in the system.

An `off-line` processor does not process any LWPs. Usually, an `off-line` processor is not interruptible by I/O devices in the system. On some processors or under certain conditions, it might not be possible to disable interrupts for an `off-line` processor. Thus, the actual effect of being `off-line` might vary from machine to machine.

A spare processor does not process any LWPs. A spare processor can be brought `on-line`, `off-line` or to `no-intr` by a privileged user of the system or by the kernel in response to changes in the system state.

A faulted processor is identified by the kernel, which monitors the behavior of processors over time. A privileged user can set the state of a faulted processor to be `on-line`, `off-line`, `spare` or `no-intr`, but must use the force option to do so.

A `no-intr` processor processes LWPs but is not interruptible by I/O devices.

A processor can not be taken `off-line` or made `spare` if there are LWPs that are bound to the processor unless the additional `-F` option is used. The `-F` option removes processor bindings of such LWPs before changing the processor's operational status. On some architectures, it might not be possible to take certain processors `off-line` or `spare` if, for example, the system depends on some resource provided by the processor.

At least one processor in the system must be able to process LWPs. At least one processor must also be able to be interrupted. Since an `off-line` or `spare` processor can be interruptible, it is possible to have an operational system with one processor `no-intr` and all other processors `off-line` or `spare` but with one or more accepting interrupts.

If any of the specified processors are powered off, `psradm` might power on one or more processors.

Only users with the `PRIV_SYS_RES_CONFIG` privilege can use the `psradm` utility.

Options The following options are supported:

`-a` Perform the action on all processors, or as many as possible.

`-f` Take the specified processors `off-line`.

`-F` Force the transition to the additional specified state. Required if one or more of the specified processors was in the faulted state. Set the specified processors to faulted, if no other transition option was specified. Forced transitions can only be made to `faulted`, `spare`, or `off-line` states. Administrators are encouraged to use the `-Q` option for `pbind(1M)` to find out which threads will be affected by forced a processor state transition.

`-i` Set the specified processors `no-intr`.

`-n` Bring the specified processors `on-line`.

`-s` Make the specified processors spare.

`-v` Output a message giving the results of each attempted operation.

Operands The following operands are supported:

processor_id The processor ID of the processor to be set `on-line` or `off-line`, `spare`, or `no-intr`.

Specify *processor_id* as an individual processor number (for example, `3`), multiple processor numbers separated by spaces (for example, `1 2 3`), or a range of processor numbers (for example, `1-4`). It is also possible to combine ranges and (individual or multiple) *processor_id*s (for example, `1-3 5 7-8 9`).

Examples **EXAMPLE 1** Setting Processors to off-line

The following example sets processors 2 and 3 `off-line`:

```
% psradm -f 2 3
```

EXAMPLE 2 Setting Processors to no-intr

The following example sets processors 1 and 2 `no-intr`:

```
% psradm -i 1 2
```

EXAMPLE 3 Setting Processors to spare

The following example sets processors 1 and 2 spare, even if either of the processors was in the faulted state:

```
% psradm -F -s 1 2
```

EXAMPLE 4 Setting All Processors on-line

```
% psradm -a -n
```

EXAMPLE 5 Forcing Processors to off-line

The following example sets processors 1 and 2 offline, and revokes the processor bindings from the processes bound to them:

```
% psradm -F -f 1 2
```

Exit Status The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

Files `/etc/wtmpx` Records logging processor status changes

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also `pbind(1M)`, `psrinfo(1M)`, `psrset(1M)`, `p_online(2)`, `processor_bind(2)`, `attributes(5)`

Diagnostics

`psradm: processor 4: Invalid argument`
The specified processor does not exist in the configuration.

`psradm: processor 3: Device busy`
The specified processor could not be taken `off-line` because it either has LWPs bound to it, is the last `on-line` processor in the system, or is needed by the system because it provides some essential service.

`psradm: processor 3: Device busy`
The specified processor could not be set `no-intr` because it is the last interruptible processor in the system, or or it is the only processor in the system that can service interrupts needed by the system.

`psradm: processor 3: Device busy`
The specified processor is powered off, and it cannot be powered on because some platform-specific resource is unavailable.

`psradm: processor 0: Not owner`
The user does not have permission to change processor status.

`psradm: processor 2: Operation not supported`
The specified processor is powered off, and the platform does not support power on of individual processors.

Name psrinfo – displays information about processors

Synopsis

```
psrinfo [-p] [-v] [processor_id]...

psrinfo [-p] -s processor_id
```

Description `psrinfo` displays information about processors. Each physical processor may support multiple virtual processors. Each virtual processor is an entity with its own interrupt ID, capable of executing independent threads.

Without the *processor_id* operand, `psrinfo` displays one line for each configured processor, displaying whether it is on-line, non-interruptible (designated by no-intr), spare, off-line, faulted or powered off, and when that status last changed. Use the processor_id operand to display information about a specific processor. See `OPERANDS`.

Options The following options are supported:

`-s` *processor_id* Silent mode. Displays `1` if the specified processor is fully on-line. Displays `0` if the specified processor is non-interruptible, spare, off-line, faulted or powered off.

Use silent mode when using `psrinfo` in shell scripts.

`-p` Display the number of physical processors in a system.

When combined with the `-v` option, reports additional information about each physical processor.

`-v` Verbose mode. Displays additional information about the specified processors, including: processor type, floating point unit type and clock speed. If any of this information cannot be determined, `psrinfo` displays `unknown`.

When combined with the `-p` option, reports additional information about each physical processor.

Operands The following operands are supported:

processor_id The processor ID of the processor about which information is to be displayed.

Specify *processor_id* as an individual processor number (for example, `3`), multiple processor numbers separated by spaces (for example, `1 2 3`), or a range of processor numbers (for example, `1-4`). It is also possible to combine ranges and (individual or multiple) *processor_id*s (for example, `1-3 5 7-8 9`).

Examples

EXAMPLE 1 Displaying Information About All Configured Processors in Verbose Mode

The following example displays information about all configured processors in verbose mode.

```
psrinfo -v
```

EXAMPLE 2 Determining If a Processor is On-line

The following example uses `psrinfo` in a shell script to determine if a processor is on-line.

```
if [ "‘psrinfo -s 3 2> /dev/null‘" –eq 1 ]
then
    echo "processor 3 is up"
fi
```

EXAMPLE 3 Displaying Information About the Physical Processors in the System

With no additional arguments, the `-p` option displays a single integer: the number of physical processors in the system:

```
> psrinfo -p
               8
```

`psrinfo` also accepts command line arguments (processor IDs):

```
> psrinfo -p 0 512   # IDs 0 and 512 exist on the
1                    # same physical processor

> psrinfo -p 0 1     # IDs 0 and 1 exist on different
2                    # physical processors
```

In this example, virtual processors `0` and `512` exist on the same physical processor. Virtual processors `0` and `1` do not. This is specific to this example and is and not a general rule.

Exit Status

The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also

`psradm(1M)`, `p_online(2)`, `processor_info(2)`, `attributes(5)`

Diagnostics

`psrinfo: processor 9: Invalid argument` The specified processor does not exist.

Name psrset – creation and management of processor sets

Synopsis

```
psrset -a [-F] processor_set_id processor_id...

psrset -b processor_set_id pid [/lwpid]...

psrset -c [-F] [processor_id]...

psrset -d processor_set_id...

psrset -e processor_set_id command [argument(s)]

psrset -f processor_set_id

psrset [-i] [processor_set_id]...

psrset -n processor_set_id

psrset -p [processor_id]...

psrset [-q] [pid [/lwpid]]...

psrset -Q [processor_set_id]...

psrset -r [-F] processor_id...

psrset -u pid [/lwpid]...

psrset -U [processor_set_id]...
```

Description The `psrset` utility controls the management of processor sets. Processor sets allow the binding of processes or LWPs to groups of processors, rather than just a single processor. Processors assigned to processor sets can run only LWPs that have been bound to that processor set.

This command cannot be used to modify processor disposition when pools are enabled. Use `pooladm(1M)` and `poolcfg(1M)` to modify processor set configuration through the resource pools facility.

Options The following options are supported:

`-a` Assign the specified processors to the specified processor set. With the additional `-F` option, all LWPs bound to the specified processors will be unbound prior to changing processor sets.

This option is restricted to users with the `PRIV_SYS_RES_CONFIG` privilege.

`-b` Bind all or a subset of the LWPs of the specified processes to the specified processor set.

LWPs bound to a processor set are restricted to run only on the processors in that set. Processes can only be bound to non-empty processor sets, that is, processor sets that have had processors assigned to them.

Bindings are inherited, so new LWPs and processes created by a bound LWP have the same binding. Binding an interactive shell to a processor, for example, binds all commands executed by the shell.

This option is restricted to users with the `PRIV_SYS_RES_CONFIG` privilege.

`-c` Create a new processor set and displays the new processor set ID. With the additional `-F` option, all LWPs bound to the specified processors will be unbound prior to assigning them to the processor set being created.

If a list of processors is given, it also attempts to assign those processors to the processor set. If this succeeds, the processors are idle until LWPs are bound to the processor set. This option is restricted to users with the `PRIV_SYS_RES_CONFIG` privilege.

Only a limited number of processor sets can be active (created and not destroyed) at a given time. This limit is always be greater than the number of processors in the system. If the `-c` option is used when the maximum number of processor sets is already active, the command fails.

The following format is used for the first line of output of the `-c` option when the `LC_MESSAGES` locale category specifies the "C" locale. In other locales, the strings `created`, `processor`, and `set` can be replaced with more appropriate strings corresponding to the locale.

`"created processor set %d\n"` *processor set ID*

`-d` Remove the specified processor set, releasing all processors and processes associated with it.

This option is restricted to users with the `PRIV_SYS_RES_CONFIG` privilege.

`-e` Execute a command (with optional arguments) in the specified processor set.

The command process and any child processes are executed only by processors in the processor set.

This option is restricted to users with the `PRIV_SYS_RES_CONFIG` privilege.

`-f` Disables interrupts for all processors within the specified processor set. See `psradm(1M)`.

If some processors in the set cannot have their interrupts disabled, the other processors still have their interrupts disabled, and the command reports an error and return non-zero exit status.

This option is restricted to users with the `PRIV_SYS_RES_CONFIG` privilege.

`-F` Forces the specified processor set operation by unbinding all threads bound to the specified processor. Only the `-a` or the `-r` option can be used in combination with this option. Administrators are encouraged to use the `-Q` option for `pbind(1M)` to find out which threads will be affected by such operation.

`-i` Display a list of processors assigned to each named processor set. If no argument is given, a list of all processor sets and the processors assigned to them is displayed. This is also the default operation if the `psrset` command is not given an option.

`-n` Enable interrupts for all processors within the specified processor set. See `psradm(1M)`.

This option is restricted to users with the `PRIV_SYS_RES_CONFIG` privilege.

`-p` Display the processor set assignments for the specified list of processors. If no argument is given, the processor set assignments for all processors in the system is given.

`-q` Display the processor set bindings of the specified processes or of all processes. If a process is composed of multiple LWPs which have different bindings and the LWPs are not explicitly specified, the bindings of only one of the bound LWPs is displayed. The bindings of a subset of LWPs can be displayed by appending "/lwpids" to the process IDs. Multiple LWPs may be selected using "-" and "," delimiters. See EXAMPLES.

`-Q` Display the LWPs bound to the specified list of processor sets, or all LWPs with processor set bindings.

`-r` Remove a list of processors from their current processor sets. Processors that are removed return to the general pool of processors.

Processors with LWPs bound to them using `pbind(1M)` can be assigned to or removed from processor sets using the `-F` option.

This option is restricted to users with the `PRIV_SYS_RES_CONFIG` privilege.

`-u` Remove the processor set bindings of a subset or all the LWPs of the specified processes, allowing them to be executed on any on-line processor if they are not bound to individual processors through `pbind`.

Users with the `PRIV_SYS_RES_CONFIG` privilege can unbind any process or LWP from any active processor set. Other users can unbind processes and LWPs from processor sets that do not have the `PSET_NOESCAPE` attribute set. In addition, the user must have permission to control the affected processes; the real or effective user ID of the user must match the real or saved user ID of the target processes.

`-U` Removes the bindings of all LWPs bound to the specified list of processor sets, or to any processor set if no argument is specified.

Operands The following operands are supported:

pid Specify *pid* as a process ID.

lwpid The set of LWPIDs of the specified process to be controlled or queried. The syntax for selecting LWP IDs is as follows:

```
2,3,4-8       LWP IDs 2, 3, and 4 through 8
-4            LWPs whose IDs are 4 or below
4-            LWPs whose IDs are 4 or above
```

processor_id Specify *processor_id* as an individual processor number (for example, `3`), multiple processor numbers separated by spaces (for example, `1 2 3`), or a range of processor numbers (for example, `1-4`). It is also possible to combine ranges and (individual or multiple) *processor_id*s (for example, `1-3 5 7-8 9`).

processor_set_id Specify *processor_set_id* as a processor set ID.

Exit Status The following exit values are returned:

`0` Successful completion.

non-0 An error occurred.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os
Interface Stability	Committed

See Also `pbind(1M)`, `pooladm(1M)`, `poolcfg(1M)`, `psradm(1M)`, `psrinfo(1M)`, `processor_bind(2)`, `processor_info(2)`, `pset_bind(2)`, `pset_create(2)`, `pset_info(2)`, `sysconf(3C)`, `libpool(3LIB)`, `attributes(5)`, `privileges(5)`

Diagnostics The following output indicates that the specified process did not exist or has exited:

```
psrset: cannot query pid 31: No such process
```

The following output indicates that the user does not have permission to bind the process:

```
psrset: cannot bind pid 31: Not owner
```

The following output indicates that the user does not have permission to assign the processor:

```
psrset: cannot assign processor 4: Not owner
```

The following output indicates that the specified processor is not on-line, or the specified processor does not exist.

```
psrset: cannot assign processor 8: Invalid argument
```

The following output indicates that an LWP in the specified process is bound to a processor and cannot be bound to a processor set that does not include that processor:

```
psrset: cannot bind pid 67: Device busy
```

The following output indicates that the specified processor could not be added to the processor set. This can be due to bound LWPs on that processor, or because that processor cannot be combined in the same processor set with other processors in that set, or because the processor is the last one in its current processor set:

```
psrset: cannot assign processor 7: Device busy
```

The following output indicates that the specified processor set does not exist:

```
psrset: cannot execute in processor set 8: Invalid argument
```

The following output indicates that the maximum number of processor sets allowed in the system is already active:

```
psrset: cannot create processor set: Not enough space
```

The following output indicates that the pools facility is active.

```
psrset: cannot assign processor 7: Operation not supported
psrset: cannot bind pid 31: Operation not supported
psrset: cannot bind pid 31: Operation not supported
psrset: could not create processor set: Operation not supported
psrset: could not remove processor set 1: Operation not supported
psrset: cannot exec in processor set 1: Operation not supported
psrset: cannot remove processor 7: Operation not supported
psrset: cannot unbind pid 31: Operation not supported
```

Name pwck, grpck – password/group file checkers

Synopsis

```
/usr/sbin/pwck [filename]

/usr/sbin/grpck [filename]
```

Description `pwck` scans the password file and notes any inconsistencies. The checks include validation of the number of fields, login name, user ID, group ID, and whether the login directory and the program-to-use-as-shell exist. The default password file is `/etc/passwd`.

`grpck` verifies all entries in the `group` file. This verification includes a check of the number of fields, group name, group ID, whether any login names belong to more than `NGROUPS_MAX` groups, and that all login names appear in the password file. `grpck` also issues a warning if it finds an entry (a single line) in the `group` file longer than 2047 characters. Such an entry causes group maintenance commands, such as `groupdel(1M)` and `groupmod(1M)`, to fail.

The default group file is `/etc/group`.

All messages regarding inconsistent entries are placed on the `stderr` stream.

Files

- `/etc/group`
- `/etc/passwd`

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also `groupdel(1M)`, `groupmod(1M)`, `getpwent(3C)`, `group(4)`, `passwd(4)`, `attributes(5)`

Diagnostics Group entries in `/etc/group` with no login names are flagged.

`Group file '`*filename*`' is empty`
The `/etc/passwd` or `/etc/group` file is an empty file.

`cannot open file` *filename*`: No such file or directory`
The `/etc/passwd` or `/etc/group` file does not exist.

Notes If no filename argument is given, `grpck` checks the local group file, `/etc/group`, and also makes sure that all login names encountered in the checked group file are known to the system `getpwent(3C)` routine. This means that the login names may be supplied by a network name service.

Name pwconv – installs and updates /etc/shadow with information from /etc/passwd

Synopsis

```
pwconv
```

Description The pwconv command creates and updates /etc/shadow with information from /etc/passwd.

pwconv relies on a special value of 'x' in the password field of /etc/passwd. This value of 'x' indicates that the password for the user is already in /etc/shadow and should not be modified.

If the /etc/shadow file does not exist, this command will create /etc/shadow with information from /etc/passwd. The command populates /etc/shadow with the user's login name, password, and password aging information. If password aging information does not exist in /etc/passwd for a given user, none will be added to /etc/shadow. However, the last changed information will always be updated.

If the /etc/shadow file does exist, the following tasks will be performed:

Entries that are in the /etc/passwd file and not in the /etc/shadow file will be added to the /etc/shadow file.

Entries that are in the /etc/shadow file and not in the /etc/passwd file will be removed from /etc/shadow.

Password attributes (for example, password and aging information) that exist in an /etc/passwd entry will be moved to the corresponding entry in /etc/shadow.

The pwconv command can only be used by the super-user.

Files

- /etc/opasswd
- /etc/oshadow
- /etc/passwd
- /etc/shadow

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also passwd(1), usermod(1M), passwd(4), attributes(5)

Diagnostics pwconv exits with one of the following values:

0 SUCCESS.

1 Permission denied.

2 Invalid command syntax.

3 Unexpected failure. Conversion not done.

4 Unexpected failure. Password file(s) missing.

5 Password file(s) busy. Try again later.

6 Bad entry in `/etc/shadow` file.

Name quot – summarize file system ownership

Synopsis

```
quot [-acfhnv] filesystem...

quot -a [-cfhnv]
```

Description `quot` displays the number of blocks (1024 bytes) in the named *filesystem* (one or more) currently owned by each user. There is a limit of 2048 blocks. Files larger than this will be counted as a 2048 block file, but the total block count will be correct.

Options The following options are supported:

`-a` Generate a report for all mounted file systems.

`-c` Display three columns giving a file size in blocks, the number of files of that size, and a cumulative total of blocks containing files of that size or a smaller size.

`-f` Display three columns giving, for each user, the number of blocks owned, the count of number of files, and the user name. This option is incompatible with the `-c` and `-v` options.

`-h` Estimate the number of blocks in the file. This does not account for files with holes in them.

`-n` Attach names to the list of files read from standard input. `quot -n` cannot be used alone, because it expects data from standard input. For example, the pipeline

```
ncheck myfilesystem | sort +0n | quot -n myfilesystem
```

will produce a list of all files and their owners. This option is incompatible with all other options.

`-v` In addition to the default output, display three columns containing the number of blocks not accessed in the last 30, 60, and 90 days.

Operands *filesystem* mount-point of the filesystem(s) being checked

Usage See `largefile(5)` for the description of the behavior of `quot` when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

Exit Status

`0` Successful operation.

`32` Error condition (bad or missing argument, bad path, or other error).

Files

`/etc/mnttab` Lists mounted file systems.

`/etc/passwd` Used to obtain user names

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also du(1), mnttab(4), passwd(4), attributes(5), largefile(5)

Notes This command can only be used by the super-user.

Name quota – display a user's ufs or zfs file system disk quota and usage

Synopsis

```
quota [-v] [username]
```

Description `quota` displays users' UFS or ZFS disk usage and limits. Only the super-user may use the optional *username* argument to view the limits of other users.

`quota` without options only display warnings about mounted file systems where usage is over quota. Remotely mounted file systems which do not have quotas turned on are ignored.

username can be the numeric UID of a user.

Options `-v` Display user's quota on all mounted file systems where quotas exist.

Usage See `largefile(5)` for the description of the behavior of `quota` when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

Files `/etc/mnttab` list of currently mounted filesystems

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also `edquota(1M)`, `quotaon(1M)`, `quotacheck(1M)`, `repquota(1M)`, `rquotad(1M)`, `attributes(5)`, `largefile(5)`, `zones(5)`

Notes `quota` displays quotas for NFS mounted UFS- or ZFS-based file systems if the `rquotad` daemon is running. See `rquotad(1M)`. In a `zones(5)` environment, `quota` displays quotas only for the zone in which it is invoked.

`quota` can display entries for the same file system multiple times for multiple mount points. For example,

```
# quota -v user1
```

might display identical quota information for `user1` at the mount points `/home/user1`, `/home/user2`, and `/home/user`, if all three mount points are mounted from the same file system with quotas turned on.

Name quotacheck – ufs file system quota consistency checker

Synopsis

```
quotacheck [-fp] [-v] filesystem...

quotacheck -a [-fpv]
```

Description `quotacheck` examines each mounted ufs file system, builds a table of current disk usage, and compares this table against the information stored in the file system's disk quota file. If any inconsistencies are detected, both the quota file and the current system copy of the incorrect quotas are updated.

filesystem is either a file system mount point or the block device on which the file system resides.

`quotacheck` expects each file system to be checked to have a quota file named `quotas` in the root directory. If none is present, `quotacheck` will not check the file system.

`quotacheck` accesses the character special device in calculating the actual disk usage for each user. Thus, the file systems that are checked should be quiescent while `quotacheck` is running.

Options The following options are supported:

`-a` Check the file systems which `/etc/mnttab` indicates are ufs file systems. These file systems must be read-write mounted with disk quotas enabled, and have an `rq` entry in the `mntopts` field in `/etc/vfstab`.

`-f` Force check on file systems with logging enabled. Use in combination with the `-p` option.

`-p` Check quotas of file systems in parallel. For file systems with logging enabled, no check is performed unless the `-f` option is also specified.

`-v` Indicate the calculated disk quotas for each user on a particular file system. `quotacheck` normally reports only those quotas modified.

Usage See `largefile(5)` for the description of the behavior of `quotacheck` when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

Files

`/etc/mnttab` Mounted file systems

`/etc/vfstab` List of default parameters for each file system

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also edquota(1M), quota(1M), quotaon(1M), repquota(1M), attributes(5), largefile(5), quotactl(7I), mount_ufs(1M)

Name quotaon, quotaoff – turn ufs file system quotas on and off

Synopsis

```
quotaon [-v] filesystem...

quotaon -a [-v]

quotaoff [-v] filesystem...

quotaoff -a [-v]
```

Description `quotaon` turns on disk quotas for one or more ufs file systems.

Before a file system may have quotas enabled, a file named `quotas`, owned by root, must exist in the root directory of the file system. See `edquota(1M)` for details on how to modify the contents of this file.

`quotaoff` turns off disk quotas for one or more ufs file systems.

The file systems specified must already be mounted.

These commands update the `mntopts` field of the appropriate entries in `/etc/mnttab` to indicate when quotas are on or off for each file system. If quotas are on, the string `quota` will be added to `mntopts`; if quotas are off, the `quota` string is not present.

filesystem must be either the mount point of a file system, or the block device on which the file system resides.

Options

quotaon

`-a` This option is normally used at boot time to enable quotas. It applies only to those file systems in `/etc/vfstab` which have "rq" in the `mntopts` field, are currently mounted "rw", and have a `quotas` file in the root directory.

`-v` Display a message for each file system after quotas are turned on.

quotaoff

`-a` Force all file systems in `/etc/mnttab` to have their quotas disabled.

`-v` Display a message for each file system affected.

Usage See `largefile(5)` for the description of the behavior of `quotaon` and `quotaoff` when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

Files

`/etc/mnttab` mounted file systems

`/etc/vfstab` list of default parameters for each file system

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also edquota(1M), quota(1M), quotacheck(1M), repquota(1M), mnttab(4), vfstab(4), attributes(5), largefile(5), quotactl(7I)

Name rad – remote administration daemon

Synopsis

```
/usr/lib/rad [-d] [-s] [-S fmri]
     [-M module [ -M module ]...]
     [-m moduledir [ -m moduledir ]...]
     [-t transpec [ -t transpec ]...]
     [-e timeout]
```

Description `rad` is a facility that securely exposes programmatic system administrative and monitoring interfaces to consumers in a variety of high-level languages.

`rad` can be used in the following ways:

- As a service:

  When run as a service, `rad` authenticates connections using `getpeerucred(3C)` or `pam(3PAM)`. When used in this way, consumed APIs are run as the authenticated user. This mode of operation is provided with both local consumers looking to isolate execution of their privileged operations and remote consumers in mind.

- As an unprivileged program:

  When run as an unprivileged program, `rad` serves solely as a bridge between its clients and the administrative APIs it publishes. When used in this way, any interfaces consumed will be run with the rights held by the `rad` process.

`rad` is modular. The APIs published by `rad` are delivered as shared objects, as are the protocols it understands and the transports it can communicate over. Multiple instances of `rad` can run simultaneously, each functioning independently of the others, providing different services to different consumers, and listening for different types of connections on different ports or interfaces. `rad` obtains its configuration from its command-line options, from `smf(5)`, or from a combination of the two.

Options The following options are available for use on the command line:

`-d`
Emit verbose debugging output.

`-e` *timeout*
Specify a connection timeout in seconds. The default value is 180 seconds.

`-m` *moduledir*
Add *moduledir* to the list of directories to scan and load modules from. The `-m` option can be used multiple times to add multiple module directories.

`-M` *module*
Add *module* to the list of modules to load. *module* should be an absolute pathname or a pathname relative to the current working directory. Modules loaded with `-M` take precedence over modules found using `-m`. The `-M` option can be used multiple times to add multiple modules.

`-t` *transpec*
Instantiate a transport specified by transport specification *transpec*. A transport specification has the following format:

```
transport[:option[=value][,option2[=value2]]...]
```

`-s`
Behave as an `svc.startd(1M)` start method. This option has the following effects:

- If the `-S` option is not specified, `rad` will read its configuration from the service identified by `scf_myname()` (see `scf_handle_create(3SCF)`).
- `rad` will use `smf_method(5)`-compatible exit statuses.
- `rad` will daemonize, returning success only once it is ready to handle requests.

`-S` *fmri*
Read configuration from the SMF service instance specified by fmri. When the `-s` option is not specified, configured transports are not read from the service to avoid endpoint conflicts with a running service.

Module directories specified on the command line are searched before module directories configured in SMF, permitting command line configuration to override SMF configuration.

Smf Configuration

When `rad` reads its configuration from smf(5), it reads general configuration from a property group called `config` of type `application`, and reads configuration for each of an arbitrary number of transports from a series of properties groups of type `xport_`*XYZ* where *XYZ* is replaced with the name of the transport type. Multiple instances of a particular transport type can be configured by creating multiple property groups of the corresponding type. The names of the property groups used to configure transports are not important.

The `config` property group contains the following properties:

`moduledir`
A list of astrings. The directories to scan and load modules from.

`modules`
A list of astrings. The file names of specific modules to load.

`debug`
A boolean. If true, `rad` will emit verbose debugging output. Defaults to `false`.

`timeout`
An integer. The maximum time in seconds to wait for an individual response from the client while authenticating. Defaults to `180`.

Service Instances

Two instances of the `svc:/system/rad` SMF service are configured to run `/usr/lib/rad/rad`:

`svc:/system/rad:local`
Configures `rad` to use the `unix` transport, with `AF_UNIX` sockets at:

- `/system/volatile/rad/radsocket`, for `getpeerucred(3C)`-authenticated connections, and...
- `/system/volatile/rad/radsocket-unauth`, for `pam(3PAM)`-authenticated connections.

`svc:/system/rad:remote`
Configures `rad` to use the `tls` transport.

Each service is configured with the following directories in its *moduledir* setting:

`/usr/lib/rad/module`
content-specific modules

`/usr/lib/rad/transport`
transport modules

`/usr/lib/rad/protocol`
protocol modules

`/usr/lib/rad/site-modules`
site-specific modules

Transports Support for different transport types is delivered in module form. Modules for the following transports are supplied with the system: pipes (`pipe`), TCP sockets (`tcp`), TLS sockets (`tls`), and Unix-domain sockets (`unix`). Each transport type has a unique set of configuration properties. The options for an instance of a transport type are configured either by defining properties in an SMF property group or by supplying sub-options to a `-t` command-line option.

The `pipe` transport reads from and writes to a specific file descriptor, as is needed when a process wishes to communicate with a child `rad` process using a pipe. The pipe transport has the following options:

`proto`
An astring. The protocol to use with this transport instance. Defaults to `rad`.

`fd`
An integer. The file descriptor to read from/write to.

`exit`
A boolean. If true, `rad` will exit when communication over the pipe ends. Defaults to `false`.

The `tcp` transport listens for clear-text connections on a TCP socket. The `tcp` transport has the following options:

`proto`
An astring. The protocol to use with this transport instance. Defaults to `rad`.

`port`
An integer. The port to listen on for connections.

`localonly`
A boolean. If true, `rad` will only listen for connections from the local machine. Defaults to `true`.

`pam_service`
An astring. The pam(3PAM) service name to use when authenticating. Defaults to `rad-tcp`. See the "Authenticating with PAM" section below.

The `tls` transport listens for TLS connections on a TCP socket. The `tls` transport has the following options:

`proto`
An astring. The protocol to use with this transport instance. Defaults to `rad`.

`port`
An integer. The port to listen on for connections.

`localonly`
A boolean. If true, `rad` will only listen for connections from the local machine. Defaults to `true`.

`certificate`
An astring. The location of the PEM-formatted x509 certificate to use.

`privatekey`
An astring. The location of the PEM-formatted private key to use.

`generate`
A boolean. If true, and if the specified certificate and `privatekey` do not exist, rad will generate a new certificate and private key using `openssl(5)`. Defaults to `false`.

`pam_service`
An astring. The pam(3PAM) service name to use when authenticating. Defaults to `rad-tls`. See the "Authenticating with PAM" section below.

The `unix` transport listens for connections on an `AF_UNIX` socket. The `unix` transport has the following options:

`proto`
An astring. The protocol to use with this transport instance. Defaults to `rad`.

`path`
An astring. The path to listen on.

`peercred`
A boolean. If true, `rad` will attempt to automatically authenticate connections using `getpeerucred(3C)`. Defaults to `true`.

`pam_service`
n astring. The pam(3PAM) service name to use when authenticating. Defaults to `rad-unix`. See the "Authenticating with PAM" section below.

Authenticating with Pam

When `rad` is run as a service, and `getpeerucred(3C)` is not applicable to the transport being used, `pam(3PAM)` is used to authenticate connections. The PAM service name used is dependent on the transport:

`rad-tls`
when connecting by means of the `tls` transport

`rad-tcp`
when connecting by means of the `tcp` transport

`rad-unix`
when connecting by means of the `unix` transport (and `peercred` is `false`)

`rad`
when connecting by means of any other transport

In rare cases, administrators may need to override the PAM service name used on a per-transport basis. For example, two `rad` TLS transports serving a single `rad` instance, with one listening on a local (more trusted) network and the other on a remote (less trusted) network, could require different PAM configurations.

In such cases, administrators can specify the name of the PAM service to use as a transport configuration property (see the "Transports" section above).

As with all PAM services, PAM will for look for entries corresponding to the PAM service for `rad` in `/etc/pam.conf` first and then `/etc/pam.d/`*service*. If no entries are found PAM will look in `/etc/pam.conf` for entries corresponding to the "other" service. If no "other" entries are found PAM will finally look for entries in `/etc/pam.d/other`.

Files

`/etc/rad/cert.pem`
The location where the remote `rad` instance (`svc:/system/rad:remote`) stores its certificate. This file is readable by all users.

`/etc/rad/key.pem`
The location where the remote `rad` instance (`svc:/system/rad:remote`) stores its private key.

`/system/volatile/rad/radsocket`
The `AF_UNIX` socket where the local `rad` instance (`svc:/system/rad:local`) accepts connections that are implicitly authenticated with `getpeerucred(3C)`.

`/system/volatile/rad/radsocket-unauth`
The `AF_UNIX` socket where the local `rad` instance (`svc:/system/rad:local`) accepts connections that must explicitly authenticate using `pam(3PAM)`.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/management/rad
Interface Stability	Private

See Also radadrgen(1), svc.startd(1M), pipe(2), getpeerucred(3C), pam(3PAM), scf_handle_create(3SCF), attributes(5), openssl(5), smf(5), smf_method(5)

Notes Two instances of rad are delivered by the system. svc:/system/rad:local listens to AF_UNIX connections at the paths /system/volatile/rad/radsocket and /system/volatile/rad/radsocket-unauth, and is enabled by default. The former AF_UNIX socket will automatically authenticate the connecting process using getpeerucred(3C), while the latter requires the connecting process to explicitly authenticate. svc:/system/rad:remote listens to TLS connections on the port 12302, requires all clients to explicitly authenticate, and is disabled by default.

Other system components, including some desktop administrative user interfaces, rely on the local instance of rad (svc:/system/rad:local).

Name raidctl – RAID hardware utility

Synopsis

```
raidctl -C "disks" [-r raid_level] [-z capacity] [-s stripe_size] [-f]
     controller

raidctl -d [-f] volume

raidctl -F filename [-f] controller...

raidctl -a {set | unset} -g disk {volume | controller}

raidctl -p "param=value" [-f] volume

raidctl -c [-f] [-r raid_level] disk1 disk2 [disk3...]

raidctl -l -g disk controller

raidctl -l volume

raidctl -l controller...

raidctl [-l]

raidctl -S [volume | controller]

raidctl -S -g disk controller

raidctl -h
```

Description The `raidctl` utility is a hardware RAID configuration tool that supports different RAID controllers by providing a CLI (command-line interface) to end-users to create, delete or display RAID volume(s). The utility can also used to set properties of a volume, assign hot-spare (HSP) disks to volumes or controllers, and to update firmware/fcode/BIOS for RAID controllers.

The `raidctl` utility requires privileges that are controlled by the underlying file system permissions. Only privileged users can manipulate the RAID system configuration. If a non-privileged user attempts to run `raidctl`, the command fails with an exit status of 1.

The `raidctl` utility, as described in this man page, defines a broad set of command line options to provide management for full-featured RAID controllers. However, support for a given option depends on two elements:

- the presence of a software driver
- the firmware level of the RAID device

The dependency on a software driver is due to the design of `raidctl`. The utility is built on a common library that enables the insertion of plug-in modules for different drivers. Currently, the Solaris operating system is shipped with a plug-in for the `mpt` driver. This plug-in does not support all of the `raidctl` options. On a given storage device, options might be further limited by the device's firmware level.

The level of support for the various `raidctl` options cannot be determined by `raidctl`. The user must rely on the documentation for his RAID controller or hardware platform.

Currently, `raidctl` Currently, `raidctl` provides some level of support for the following RAID controllers:

- LSI1020 SCSI HBA
- LSI1030 SCSI HBA
- LSI1064 SAS HBA
- LSI1068 SAS HBA

All of the above HBAs are maintained by the `mpt` driver, on X86-32/64 and SPARC platforms.

Options The following options are supported:

`-C` *"disks"* [`-r` *raid_level*] [`-z` *capacity*] [`-s` *stripe_size*] [`-f`] controller
Create a RAID volume using specified disks.

When creating a RAID volume using this option, the identity of the newly created volume is automatically generated and raidctl reports it to the user.

The argument specified by this option contains the elements used to form the volume that is created. Elements can be either disks or sub-volumes, where disks are separated by space(s) and a sub-volume is a set of disks grouped by parenthesis. All disks should be in `C.ID.L` expression (for example, `0.1.2` represents a physical disk of channel 0, target id 1, and logical unit number 2). The argument must match the RAID level specified by the `-r` option, even if it's omitted. This means the argument can only be:

for RAID 0
At least 2 disks

for RAID 1
Only 2 disks

for RAID 1E
At least 3 disks

for RAID 5
At least 3 disks

for RAID 10
At least 2 sub-volumes, each sub-volume must be formed by 2 disks

for RAID 50
At least 2 sub-volumes, each sub-volume must be formed by at least 3 disks, and the disk amount in each sub-volume should be the same

For example, the expression "0.0.0 0.1.0" means that the 2 specified disks form a RAID volume, which can either be a RAID 0 or a RAID 1 volume. "(0.0.0 0.1.0)(0.2.0 0.3.0)" means that the first 2 disks and the last 2 disks form 2 sub-volumes, and that these 2 sub-volumes form a RAID 10 volume. See the `EXAMPLES` section for more samples.

The `-r` option specifies the RAID level of the volume that is created. Possible levels are 0, 1, 1E, 5, 10, 50. If this option is omitted, `raidctl` creates a RAID 1 volume by default.

The `-z` option specifies the capacity of the volume that is created. The unit can be tera-bytes, giga-bytes, or mega-bytes (for example, 2t, 10g, 20m, and so on). If this option is omitted, `raidctl` calculates the maximum capacity of the volume that can be created by the specified disks and uses this value to create the volume.

The `-s` option specifies the stripe size of the volume that is created. The possible values are 512, 1k, 2k, 4k, 8k, 16k, 32k, 64k, or 128k. If this option is omitted, `raidctl` chooses an appropriate value for the volume (for example, 64k).

In some cases, the creation of a RAID volume can cause data on specified disks to be lost (for instance, on LSI1020, LSI1030, SAS1064, or SAS1068 HBAs), and `raidctl` prompts the user for confirmation about the creation. Use the `-f` option to force the volume creation without prompting the user for confirmation.

The controller argument is used to identify which RAID controller the specified disks belongs. The `-l` option can be used to list the controller's ID number.

`-d` [`-f`] *volume*
Delete the RAID volume specified as volume. The volume is specified in canonical form (for example, `c0t0d0`).

When a volume is deleted, all data is lost. Therefore, unless the `-f` option is specified, `raidctl` prompts the user for confirmation before deleting the volume.

`-F` *filename* [`-f`] *controller…*
Update the firmware running on the specified controller(s). The `raidctl` utility prompts the user for confirmation of this action, unless the `-f` option is provided.

`-a` {`set` | `unset`} `-g` *disk* {`volume` | `controller`}
If the volume is specified, `raidctl` sets or unsets the disk as a local hot-spare disk dedicated to the volume, depending on the value specified by the `-a` option. If the controller is specified, `raidctl` sets or unsets the disk as a global hot-spare disk.

`-p` *"param=value"* [`-f`] *volume*
Change the property value for a given RAID volume. This option can be used to change cache write policy or to activate a volume. When changing the cache write policy, *param* should be the string `wp` (`SET_WR_POLICY`), and *value* can be either `on` or `off`. When used to activate a volume, *param* should be `state` and *value* should be `activate`.

Changing a RAID volume's property can affect the internal behavior of the RAID controller, so `raidctl` prompts the user for a confirmation before applying the change, unless the `-f` option is specified.

`-c` [`-f`] [`-r` *raid_level*] *disk1 disk2* [*disk3*...]
Create a volume using the specified disks. This is an alternative to the `-C` option with similar functionality. This option is preserved for compatibility reasons, but only works with LSI1020, LSI1030, SAS1064, and SAS1068 HBAs to create RAID 0, RAID 1, or RAID 1E volumes. For other HBAs, the user can only use the `-C` option.

The `-r` option can be used to specify the RAID level of the target volume. If the `-r` option is omitted, `raidctl` creates a RAID 1 volume.

Disks must be specified in Solaris canonical format (for example, c0t0d0).

Creating a RAID 1 volume with this option replaces the contents of disk2 with the contents of disk1.

When the user creates a RAID volume with this option, the RAID volume assumes the identity of disk1. Other disks become invisible and the RAID volume appears as one disk.

Creating a volume with this option is by default interactive. The user must answer a prompt affirmatively to create the volume. Use the `-f` option to force the volume creation without prompting the user for confirmation.

`-l` `-g` *disk controller*
Display information about the specified disk of the given controller. The output includes the following information:

Disk
Displays the disk in `C.ID.L` expression disk.

Vendor
Displays the vendor ID string.

Product
Displays the product ID string.

Capacity
Displays the total capacity of the disk.

Status
Displays the current status of disk. The status can be either "`GOOD`" (operating normally), "`FAILED`" (non-functional), or "`MISSING`" (disk not present).

HSP
Indicates if the disk has been set as a global hot-spare disk, local hot-spare disk, or a normal one. If it is a local hot-spare disk, all volumes which this disk is assigned to are displayed.

GUID
GUID string for the specified disk. This is an additional datum and might be unavailable in some cases.

`-l` *volume*
Display information about the specified volume. The output includes the following information:

Volume
Displays volume in canonical format.

Sub
Displays sub-volumes, if the specified volume is of RAID 10 or RAID 50 volume.

Disk
Displays all disks that form the specified volume.

Stripe Size
Displays the stripe size of the volume.

Status
Displays the status of the specified volume, or the sub-volumes or disks that form the specified volume. For an inactive volume, the status should be `INACTIVE`; otherwise it can be `OPTIMAL` (operating optimally), `DEGRADED` (operating with reduced functionality), `FAILED` (non-functional), or `SYNC` (disks are syncing). For a disk, the status can be `GOOD`, `FAILED`, or `MISSING`.

Cache
Indicates whether the cache is applied to `I/O` write activities. The cache can be either "`ON`" or "`OFF`".

RAID level
Displays the RAID level. The RAID level can be either 0, 1, 1E, 5, 10, or 50.

`-l` *controller ...*
Display information about the specified controller(s). The output includes the following information:

Controller
Displays the RAID controller's ID number.

Type
Displays the RAID controller's product type.

fw_version
Displays the controller's firmware version.

`[-l]`
List all RAID related objects that the `raidctl` utility can manipulate, including all available RAID controllers, RAID volumes, and physical disks. The `-l` option can be omitted.

The output includes the following information:

Controller
Displays the RAID controller's ID number.

Volume
Displays the logical RAID volume name.

Disk
Displays the RAID disk in `C.ID.L` expression.

`-S` [`volume` | `controller`]
Takes a snapshot of the RAID configuration information including all available RAID devices, RAID controllers, volumes, and disks.

Each line of the output specifies a RAID device and its related information, separated by space(s). All volumes and disks belong to the last specified controller.

The output lists the following information:

Controller
Displays the controller ID number, and the controller type string in double-quotation marks.

Volume
Displays the RAID volume name, number of component disks, the `C.ID.L` expression of the component disks, the RAID level, and the status. The status can be either `OPTIMAL`, `DEGRADED`, `FAILED`, or `SYNCING`.

Disk
Displays the `C.ID.L` expression of the disk, and the status. The status can be either `GOOD`, `FAILED`, or `HSP` (disk has been set as a stand-by disk).

If a volume or a controller is specified, a snapshot is only taken of the information for the specified volume or controller.

`-S` `-g` *disk controller*
Takes a snapshot of the information for the specified disk.

`-h`
Print out the usage string.

Examples

EXAMPLE 1 Creating the RAID Configuration

The following command creates a RAID 0 volume of 10G on controller 0, and the stripe size is set to 64k:

```
# raidctl -C "0.0.0 0.2.0" -r 0 -z 10g -s 64k 0
```

The following command creates a RAID 1 volume on controller 2:

```
# raidctl -C "0.0.0 1.1.0" -r 1 2
```

The following command creates a RAID 5 volume on controller 2:

```
# raidctl -C "0.0.0 0.1.0 0.2.0" -r 5 2
```

The following command creates a RAID 10 volume on controller 0:

```
# raidctl -C "(0.0.0 0.1.0)(0.2.0 0.3.0)" -r 10 0
```

The following command creates a RAID 50 volume on controller 0:

```
# raidctl -C "(0.0.0 0.1.0 0.2.0)(0.3.0 0.4.0 0.5.0)" -r 50 0
```

EXAMPLE 2 Displaying the RAID Configuration

The following command displays all available controllers, volumes, and disks:

```
# raidctl -l

Controller: 0
Controller: 2
         Volume:c2t0d0
         Disk: 0.0.0
         Disk: 0.1.0
         Disk: 0.2.0
         Disk: 0.3.0(HSP)
```

The following command displays information about controller 2:

```
# raidctl -l 2

Controller      Type            Fw_version
------------------------------------------------------------
c2              LSI 1030        1.03.39.00
```

The following command displays information about the specified volume:

```
# raidctl -l c2t0d0

Volume                  Size    Stripe  Status  Cache   RAID
         Sub                     Size                    Level
                Disk
----------------------------------------------------------------
c2t0d0                 10240M  64K     OPTIMAL ON      RAID5
                0.0.0   5120M           GOOD
                0.1.0   5120M           GOOD
                0.2.0   5120M           GOOD
```

The following command displays information about disk 0.0.0 on controller 0:

```
# raidctl -l -g 0.0.0  0

Disk    Vendor  Product         Firmware Capacity        Status  HSP
------------------------------------------------------------------------
0.0.0   HITACHI H101473SCSUN72G SQ02     68.3G           GOOD    N/A
GUID:2000000cca02536c
```

EXAMPLE 3 Deleting the RAID Configuration

The following command deletes a volume:

```
# raidctl -d c0t0d0
```

EXAMPLE 4 Updating Flash Images on the Controller

The following command updates flash images on the controller 0:

```
# raidctl -F lsi_image.fw 0
```

EXAMPLE 5 Setting or Unsetting a Hot-Spare Disk

The following command sets disk 0.3.0 on controller 2 as a global hot-spare disk:

```
# raidctl -a set -g 0.3.0 2
```

The following command sets disk 0.3.0 on controller 2 as a local hot-spare disk to volume c2t0d0:

```
# raidctl -a set -g 0.3.0 c2t0d0
```

The following command converts disk 0.3.0 on controller 2 from a global hot-spare disk to a normal one:

```
# raidctl -a unset -g 0.3.0 2
```

The following command removes disk 0.3.0 from being a local hot-spare disk from volume c2t0d0:

```
# raidctl -a unset -g 0.3.0 c2t0d0
```

EXAMPLE 6 Setting the Volume's Property

The following command sets the write policy of the volume to "`off`":

```
# raidctl -a set -p "wp=off" c0t0d0
```

EXAMPLE 7 Creating Volumes with the - c Option

The following command creates a RAID 1 volume:

```
# raidctl -c c0t0d0 c0t1d0
```

The following command creates a RAID 0 volume:

```
# raidctl -c -r 0 c0t1d0 c0t2d0 c0t3d0
```

EXAMPLE 8 Taking a Snapshot of the RAID Configuration

The following command takes a snapshot of all RAID devices:

```
# # raidctl -S

1 "LSI 1030"
c1t1d0 2 0.2.0 0.3.0 1 DEGRADED
0.2.0 GOOD
0.3.0 FAILED
```

The following command takes a snapshot about volume c1t0d0:

EXAMPLE 8 Taking a Snapshot of the RAID Configuration *(Continued)*

```
# raidctl -S c1t0d0

c1t0d0 2 0.0.0 0.1.0 1 OPTIMAL
```

The following command takes a snapshot about disk 0.1.0 on controller 1:

```
# raidctl -S -g 0.1.0 1

0.1.0 GOOD
```

Exit Status

The following exit values are returned:

`0`
Successful completion.

`1`
Invalid command line input or permission denied.

`2`
Request operation failed.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os
Interface Stability	Committed

See Also

`attributes(5)`, `mpt(7D)`

Oracle Solaris Administration: Common Tasks

Warnings

Do not create raid volumes on internal SAS disks if you are going to use the Solaris Multipathing I/O feature (also known as MPxIO). Creating a new raid volume under Solaris Multipathing gives your root device a new GUID which does not match the GUID for the existing devices. This causes a boot failure since your root device entry in `/etc/vfstab` does not match.

Notes

The `-z` option is not supported on systems that use the `mpt` driver and LSI RAID controllers.

Name ramdiskadm – administer ramdisk pseudo device

Synopsis

```
/usr/sbin/ramdiskadm -a name size [g | m | k | b]

/usr/sbin/ramdiskadm -d name

/usr/sbin/ramdiskadm
```

Description The `ramdiskadm` command administers `ramdisk(7D)`, the ramdisk driver. Use `ramdiskadm` to create a new named ramdisk device, delete an existing named ramdisk, or list information about existing ramdisks.

Ramdisks created using `ramdiskadm` are not persistent across reboots.

Options The following options are supported:

`-a` *name size* Create a ramdisk named *name* of size *size* and its corresponding block and character device nodes.

name must be composed only of the characters a-z, A-Z, 0-9, _ (underbar), and - (hyphen), but it must not begin with a hyphen. It must be no more than 32 characters long. Ramdisk names must be unique.

The size can be a decimal number, or, when prefixed with `0x`, a hexadecimal number, and can specify the size in bytes (no suffix), 512-byte blocks (suffix `b`), kilobytes (suffix `k`), megabytes (suffix `m`) or gigabytes (suffix `g`). The size of the ramdisk actually created might be larger than that specified, depending on the hardware implementation.

If the named ramdisk is successfully created, its block device path is printed on standard out.

`-d` *name* Delete an existing ramdisk of the name *name*. This command succeeds only when the named ramdisk is not open. The associated memory is freed and the device nodes are removed.

You can delete only ramdisks created using `ramdiskadm`. It is not possible to delete a ramdisk that was created during the boot process.

Without options, `ramdiskadm` lists any existing ramdisks, their sizes (in decimal), and whether they can be removed by `ramdiskadm` (see the description of the `-d` option, above).

Examples **EXAMPLE 1** Creating a 2MB Ramdisk Named mydisk

```
# ramdiskadm -a mydisk 2m
/dev/ramdisk/mydisk
```

EXAMPLE 2 Listing All Ramdisks

```
# ramdiskadm
Block Device                    Size  Removable
/dev/ramdisk/miniroot     134217728    No
/dev/ramdisk/certfs         1048576    No
/dev/ramdisk/mydisk         2097152    Yes
```

Exit Status

`ramdiskadm` returns the following exit values:

`0` Successful completion.

`>0` An error occurred.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os
Interface Stability	Committed

See Also

`attributes(5)`, `ramdisk(7D)`

Notes

The abilities of `ramdiskadm` and the privilege level of the person who uses the utility are controlled by the permissions of `/dev/ramdiskctl`. Read access allows query operations, for example, listing device information. Write access is required to do any state-changing operations, for example, creating or deleting ramdisks.

As shipped, `/dev/ramdiskctl` is owned by `root`, in group `sys`, and mode 0644, so all users can do query operations but only root can perform state-changing operations. An administrator can give write access to non-privileged users, allowing them to add or delete ramdisks. However, granting such ability entails considerable risk; such privileges should be given only to a trusted group.

Name rcapadm – configure resource capping daemon

Synopsis

```
rcapadm
```

```
rcapadm [ [-n] -E | -D]
     [-i interval=value,...,interval=value] [-c percent]
     [-z zonename -m maxvalue]
```

Description The `rcapadm` command allows a user with the privileges described below to configure various attributes of the resource capping daemon. If used without arguments, `rcapadm` displays the current status of the resource capping daemon if it has been configured. See `rcapd(1M)` for more information.

In the current release of the Solaris operating environment, `rcapadm` is available to users with all privileges and to users who have the Process Management profile in their list of profiles. The System Administrator role includes the Process Management profile.

Options

`-c` *percent*
: Set the minimum physical memory utilization for memory cap enforcement. Caps will not be enforced until the physical memory available to processes is low. The *percent* value should be in the range 0 to 100. The minimum (and default) value is `0`, which means that memory caps are always enforced.

`-D`
: Disable the resource capping daemon so that it will not be started when the system is booted. Also stop the resource capping daemon now, if the `-n` option is not specified and it is currently running.

`-E`
: Enable the resource capping daemon so that it will be started each time the system is booted. Also start the resource capping daemon now, if the `-n` option is not specified and it is not currently running.

`-i` *interval=value,...,interval=value*
: Set intervals for various periodic operations performed by `rcapd`. All intervals are specified in seconds. You can set the following intervals:

  `scan` — The interval at which `rcapd` scans for new processes. The default scan interval is every 15 seconds. The minimum value is 1 second.

  `sample` — The interval of process resident set size sampling. The default sample interval is every 5 seconds. The minimum value is 1 second.

`report` The interval at which various paging statistics are updated by `rcapd`, in seconds. These statistics can be viewed by using `rcapstat`(1SRM). The default reporting interval is every 5 seconds. When the interval is set to `0`, statistics will not be updated.

Note – Paging refers to the act of relocating portions of memory, called pages, to or from physical memory. `rcapd` pages out the most infrequently used pages.

`config` The reconfiguration interval, in seconds. At each reconfiguration event, `rcapd` checks its configuration file for updates, and scans the project databases for new project caps. The default reconfiguration interval is every 60 seconds. The minimum interval is `0`. When the interval is set to `0`, no periodic reconfiguration occurs, although the running daemon can still be reconfigured by sending it `SIGHUP`.

`-m` *maxvalue* Used in conjunction with the `-z` option. Specifies a value for `rcap.max-rss`, a dynamically-set cap on the usage of physical memory for the zone specified by `-z`. You can apply a scale (`K`, `M`, `G`, `T`) to the value you specify. `K` means kilobyte; `M`, megabyte; `G`, gigabyte; and `T`, terabyte. For example, `100M` is 100 megabytes.

To remove an existing cap, specify `0M`.

`-n` Do not affect the running state of the resource capping daemon when enabling or disabling it.

`-z` *zonename* Used in conjunction with the `-m` option. Specifies the zone for which you are dynamically specifying a cap on physical memory usage (using `-m`).

Note – To set a persistent cap on memory usage within a zone, use `zonecfg`(1M).

Examples

EXAMPLE 1 Configuring the Resource Capping Daemon with Immediate Enforcement

```
# rcapadm -E -i scan=15,sample=5,report=5,config=60 -c 0
```

EXAMPLE 2 Specifying a Resource Cap for a Zone

The command shown below specifies the maximum amount of memory that can be consumed by a specified zone. Note that this value lasts only until the next reboot. To set a persistent cap, use `zonecfg(1M)`.

```
# rcapadm -z testzone -m 512M
```

Exit Status

The following exit values are returned:

`0` Successful completion. The modifications to the current configuration were valid and made successfully.

`1` An error occurred. A fatal error occurred either in obtaining or modifying the resource capping configuration.

`2` Invalid command-line options were specified.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/resource-mgmt/resource-caps
Interface Stability	Committed

The `-z` and `-m` options are committed interfaces.

See Also

`rcapstat(1)`, `rcapd(1M)`, `zonecfg(1M)`, `project(4)`, `attributes(5)`, `zones(5)`

"Physical Memory Control Using the Resource Capping Daemon" in *Oracle Solaris Administration: Oracle Solaris Zones, Oracle Solaris 10 Zones, and Resource Management*

Name rcapd – resource cap enforcement daemon

Synopsis

```
rcapd [-d]
```

Description The `rcapd` daemon enforces resource caps on collections of processes. Per-project and per-zone physical memory caps are supported. For information about projects, see `project(4)`. For zones information, see `zones(5)`

When the resident set size (RSS) of a collection of processes exceeds its cap, `rcapd` takes action and reduces the RSS of the collection.

The virtual memory system divides physical memory into segments known as pages. To read data from a file into memory, the virtual memory system reads in individual pages. To reduce resource consumption, the daemon can page out, or relocate, infrequently used pages to an area outside of physical memory.

In the `project` file, caps are defined for projects that have positive values for the following project attribute:

`rcap.max-rss` The total amount of physical memory, in bytes, that is available to the project's member processes

See `project`(4) for a description of project attributes.

For a system with one or more zones, you can dynamically set the `rcap.max-rss` value for a zone with `rcapadm(1M)`. To set a persistent cap on memory usage within a zone, you use `zonecfg(1M)`.

You configure `rcapd` through the use of `rcapadm(1M)`. The daemon can be monitored with `rcapstat(1)`. Configuration changes are incorporated into `rcapd` by sending it `SIGHUP` (see `kill(1)`), or according to the configuration interval (see `rcapadm(1M)`).

Options The following option is supported:

`-d` Enable debug mode. Messages are displayed on the invoking user's terminal.

Examples **EXAMPLE 1** Setting Resident Set Size Cap Attribute

The following line in the `/etc/project` database sets an RSS cap of 1073741824 bytes for a project named `foo`.

```
foo:100::foo,root::rcap.max-rss=10737418240
```

Exit Status The following exit values are returned:

0 Successful completion.

1 An error occurred.

2 Invalid command-line options were specified.

Files `/etc/project` Project database.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/resource-mgmt/resource-caps
Interface Stability	Committed

See Also `rcapstat`(1), `svcs(1)`, `rcapadm(1M)`, `zonecfg(1M)`, `svcadm(1M)`, `project(4)`, `attributes(5)`, `smf(5)`, `zones(5)`

"Physical Memory Control Using the Resource Capping Daemon" in *Oracle Solaris Administration: Oracle Solaris Zones, Oracle Solaris 10 Zones, and Resource Management*

Notes If killed with `SIGKILL`, `rcapd` can leave processes in a stopped state. Use `SIGTERM` to cause `rcapd` to terminate properly.

A collection's RSS can exceed its cap for some time before the cap is enforced, even if sufficient pageable memory is available. This period of time can be reduced by shortening the RSS sampling interval with `rcapadm`.

The `rcapd` service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/system/rcap:default
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. The service's status can be queried using the `svcs(1)` command.

Name rctladm – display or modify global state of system resource controls

Synopsis

```
rctladm [-lu] [-e action] [-d action] [name...]
```

Description The `rctladm` command allows the examination and modification of active resource controls on the running system. An instance of a resource control is referred to as an *rctl*. See `setrctl(2)` for a description of an `rctl`; see `resource_controls(5)` for a list of the `rctl`s supported in the current release of the Solaris operating system. Logging of `rctl` violations can be activated or deactivated system-wide and active `rctl`s (and their state) can be listed.

An `rctladm` command without options is the equivalent of an `rctladm` with the `-l` option. See the description of `-l` below.

Options The following options are supported:

`-d` *action*
`-e` *action*
Disable (`-d`) or enable (`-e`) the global action on the specified `rctl`s. If no `rctl` is specified, no action is taken and an error status is returned. You can use the special token `all` with the disable option to deactivate all global actions on a resource control.

You can set the `syslog` action to a specific degree by assigning a severity level. To do this, specify `syslog=`*level*, where *level* is one of the string tokens given as valid severity levels in `syslog(3C)`. You can omit the common `LOG_` prefix on the severity level. Note that not all `rctl`s support the `syslog` action. See `resource_controls(5)`.

If the enabling of the `syslog` action for an rctl results in a continuous stream of logged messages, log output will be restricted to one message every five seconds. In such a circumstance, some messages will be dropped. No corrective action need to be taken.

`-l`
List information about `rctl`s. The name, global event actions and statuses, and global flags are displayed. If one or more name operands are specified, only those `rctl`s matching the names are displayed.

`-u`
Configure resource controls based on the contents of `/etc/rctladm.conf`. Any name operands are ignored.

Operands The following operands are supported:

name
The name of the `rctl` to operate on. Multiple `rctl` names can be specified. If no names are specified, and the list action has been specified, then all `rctl`s are listed. If the enable or disable action is specified, one or more `rctl` names must be specified.

Examples

EXAMPLE 1 Activating System Logging for Specific Violations

The following command activates system logging of all violations of `task.max-lwps`.

```
# rctladm -e syslog task.max-lwps
#
```

EXAMPLE 2 Examining the Current Status of a Specific Resource

The following command examines the current status of the `task.max-lwps` resource.

```
$ rctladm -l task.max-lwps
task.max-lwps               syslog=DEBUG
$
```

Exit Status

The following exit values are returned:

`0`
Successful completion.

`1`
A fatal error occurred. A message is written to standard error to indicate each resource control for which the operation failed. The operation was successful for any other resource controls specified as operands.

`2`
Invalid command line options were specified.

Files

`/etc/rctladm.conf`
Each time `rctladm` is executed, it updates the contents of `rctladm.conf` with the current configuration.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also

`setrctl(2)`, `getrctl(2)`, `prctl(1)`, `rctlblk_get_global_flags(3C)`, `rctlblk_get_global_action(3C)`, `attributes(5)`, `resource_controls(5)`

Notes

By default, there is no global logging of `rctl` violations.

Name rdate – set system date from a remote host

Synopsis `rdate` *hostname*

Description `rdate` sets the local date and time from the *hostname* given as an argument. You must have the authorization `solaris.system.date` on the local system. Typically, `rdate` is used in a startup script.

`rdate` requests are responded to by the "time" service on the specified host. To enable the "time" service, use the following commands:

```
svcadm enable time:stream
svcadm enable time:dgram
```

Usage The `rdate` command is IPv6–enabled. See `ip6(7P)`.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	service/network/network-clients

See Also `inetd(1M)`, `inetd.conf(4)`, `attributes(5)`, `ip6(7P)`

Name reboot – restart the operating system

Synopsis

```
/usr/sbin/reboot [-dlnq] [-f | -p] [boot_arguments]

/usr/sbin/reboot [-f [-e environment] | -p] [-dlnq] [boot_arguments]
```

Description The `reboot` utility restarts the kernel. The kernel is loaded into memory by the PROM monitor, which transfers control to the loaded kernel.

On x86 systems, when the `-f` flag is specified, the running kernel will load the next kernel into memory, then transfer control to the newly loaded kernel. This form of `reboot` is shown in the second synopsis, above.

Although `reboot` can be run by the super-user at any time, `shutdown(1M)` is normally used first to warn all users logged in of the impending loss of service. See `shutdown(1M)` for details.

The `reboot` utility performs a `sync(1M)` operation on the disks, and then a multi-user reboot is initiated. See `init(1M)` for details. On x86 systems, reboot may also update the boot archive as needed to ensure a successful reboot.

The `reboot` utility normally logs the reboot to the system log daemon, `syslogd(1M)`, and places a shutdown record in the login accounting file `/var/adm/wtmpx`. These actions are inhibited if the `-n` or `-q` options are present.

Normally, the system reboots itself at power-up or after crashes.

Options The following options are supported:

`-d`
Force a system crash dump before rebooting. See `dumpadm(1M)` for information on configuring system crash dumps.

`-e`
If `-f` is present, reboot to the specified boot environment.

This option is currently available only on x86 systems.

`-f`
For x86 systems:

Fast reboot, bypassing firmware and boot loader. The new kernel will be loaded into memory by the running kernel, and control will be transferred to the newly loaded kernel. If disk or kernel arguments are specified, they must be specified before other boot arguments.

For SPARC systems:

Speeds up rebooting by skipping some POST tests.

The service `svc:/system/boot-config:default` is enabled by default. It requires `solaris.system.shutdown` as `action_authorization` and `value_authorization`. When the `config/fastreboot_default` property is set to `true`, `reboot` will behave as `reboot -f`. The value of this property can be changed using `svccfg(1M)` and `svcadm(1M)`, to control the default reboot behavior.

See `EXAMPLES` for details.

`-l`
Suppress sending a message to the system log daemon, `syslogd(1M)` about who executed `reboot`.

`-n`
Avoid calling `sync(2)` and do not log the reboot to `syslogd(1M)` or to `/var/adm/wtmpx`. The kernel still attempts to sync filesystems prior to reboot, except if the `-d` option is also present. If `-d` is used with `-n`, the kernel does not attempt to sync file systems.

`-p`
Reboot to prom. This flag can be used to reboot the system through firmware without changing the default reboot behavior as denoted by the `config/fastreboot_default` property setting in `system/boot-config` service.

The `-p` and `-f` options are mutually exclusive.

`-q`
Quick. Reboot quickly and ungracefully, without shutting down running processes first.

Operands

The following operands are supported:

boot_arguments
An optional *boot_arguments* specifies arguments to the `uadmin(2)` function that are passed to the boot program and kernel upon restart. The form and list of arguments is described in the `boot(1M)` and `kernel(1M)` man pages.. If the arguments are specified, whitespace between them is replaced by single spaces unless the whitespace is quoted for the shell. If the *boot_arguments* begin with a hyphen, they must be preceded by the `--` delimiter (two hyphens) to denote the end of the `reboot` argument list.

Examples

EXAMPLE 1 Passing the `-r` and `-v` Arguments to `boot`

In the following example, the delimiter `--` (two hyphens) must be used to separate the options of `reboot` from the arguments of `boot(1M)`.

```
example# reboot -dl -- -rv
```

EXAMPLE 2 Rebooting Using a Specific Disk and Kernel

The following example reboots using a specific disk and kernel.

```
example# reboot disk1 kernel.test/unix
```

EXAMPLE 3 Fast Rebooting

The following examples use the `-f` option to perform fast reboots.

If the service `svc:/system/boot-config:default` is enabled and property `config/fastreboot_default` is set to `true`, the `-f` option can be omitted.

On an x86 system, the following command reboots to the default entry in the GRUB (see `grub(5)`) menu file `menu.lst`.

```
example# reboot -f
```

The following command reboots to another UFS root disk.

```
example# reboot -f -- '/dev/dsk/c1d0s0'
```

The following command reboots to another ZFS root pool.

```
example# reboot -f -- 'rpool/ROOT/root2'
```

The following command reboots to `mykernel` on the same disk with `-k` option.

```
example# reboot -f -- '/platform/i86pc/mykernel/amd64/unix -k'
```

The following command reboots to `mykernel` off another root disk mounted on `/mnt`.

```
example# reboot -f -- '/mnt/platform/i86pc/mykernel/amd64/unix -k'
```

The following command reboots to `/platform/i86pc/kernel/$ISADIR/unix` on another boot environment named `second_root`.

```
example# reboot -f -e second_root
```

The following command reboots to the same kernel with `-kv` options.

```
example# reboot -f -- '-kv'
```

The following commands disable the fast-reboot-by-default behavior.

```
example# svccfg -s "system/boot-config:default" \
setprop config/fastreboot_default=false
example# svcadm refresh svc:/system/boot-config:default
```

The following commands re-enable the fast-reboot-by-default behavior.

```
example# svccfg -s "system/boot-config:default" \
setprop config/fastreboot_default=true
example# svcadm refresh svc:/system/boot-config:default
```

EXAMPLE 4 Rebooting to a Particular GRUB Menu

The following commands will reboot to entry 2 in the GRUB menu.

```
example# bootadm list-menu
  the location for the active GRUB menu is: /rpool/boot/grub/menu.lst
  default 0
```

EXAMPLE 4 Rebooting to a Particular GRUB Menu *(Continued)*

```
  timeout 10
  0 zfsbe1
  1 zfsbe1 failsafe
  2 zfsbe2
  3 zfsbe2 Solaris xVM
  4 zfsbe2 failsafe
example# reboot 2
```

Files

`/var/adm/wtmpx`
login accounting file

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also

`mdb(1)`, `boot(1M)`, `dumpadm(1M)`, `fsck(1M)`, `halt(1M)`, `init(1M)`, `kernel(1M)`, `shutdown(1M)`, `svcadm(1M)`, `svccfg(1M)`, `sync(1M)`, `syslogd(1M)`, `sync(2)`, `uadmin(2)`, `reboot(3C)`, `attributes(5)`, `grub(5)`

Notes

The `reboot` utility does not execute the scripts in `/etc/rc`*num*`.d` or execute shutdown actions in `inittab(4)`. To ensure a complete shutdown of system services, use `shutdown(1M)` or `init(1M)` to reboot a Solaris system.

Name rem_drv – remove a device driver from the system

Synopsis

```
rem_drv [-b basedir] [-C] [-n] device_driver
```

Description The `rem_drv` command informs the system that the device driver *device_driver* is no longer valid. If possible, `rem_drv` unloads *device_driver* from memory. `rem_drv` also updates the system driver configuration files.

If `rem_drv` has been executed, the next time the system is rebooted it automatically performs a reconfiguration boot (see `kernel(1M)`).

Options The following options are supported:

`-b` *basedir*
Sets the path to the root directory of the diskless client. Used on the server to execute `rem_drv` for a client. The client machine must be rebooted to unload the driver.

Note – The root file system of any non-global zones must not be referenced with the `-b` option. Doing so might damage the global zone's file system, might compromise the security of the global zone, and might damage the non-global zone's file system. See `zones(5)`.

`-C`
Remove dangling device attribute nodes bound to the driver being removed. This causes any device ownership or permissions customizations made to any node not to be preserved if the driver is added back. Recommended for use when reprovisioning a machine from one configuration or use to another where past administrative customizations might not be desired.

`-n`
Do not try to detach and unload *device_driver*, just modify the system configuration files for that driver.

Examples

EXAMPLE 1 Removing the sd Driver

The following example removes the `sd` driver from use:

```
example% rem_drv sd
```

EXAMPLE 2 Removing a Diskless Client

The following example removes the driver from the `sun1` diskless client. The driver is not uninstalled or unloaded until the client machine is rebooted.

```
example% rem_drv -b /export/root/sun1 sd
```

Note the caveat on the use of the `-b` option in the description of that option, above.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also add_drv(1M), kernel(1M), update_drv(1M), attributes(5), zones(5), devfs(7FS)

Name remove_allocatable – remove entries from allocation databases

Synopsis

```
/usr/sbin/remove_allocatable  [-f] -n name
/usr/sbin/remove_allocatable  [-f] [-d] -t dev-type
```

Description `remove_allocatable` removes entries of user allocatable devices from the device allocation mechanism. `remove_allocatable` also removes entries of some non-allocatable devices, such as printers, whose label range is managed by the mechanism.

Options The following options are supported:

`-d` Removes system-supplied default attributes of the device type that is specified with `-t`.

`-f` Force the removal of an entry. `remove_allocatable` exits with an error if this option is not specified when an entry with the specified device name no longer exists.

`-n` *name* Removes the entry for the device *name*.

`-t` *dev-type* Removes devices of type *dev-type*.

Exit Status When successful, `remove_allocatable` returns an exit status of `0` (true). `remove_allocatable` returns a nonzero exit status in the event of an error. The exit codes are as follows:

1 Invocation syntax error

2 Unknown system error

3 Device *name* or *dev-type* not found. This error occurs only when the `-f` option is not specified.

4 Permission denied. User does not have DAC or MAC access to database.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/trusted
Interface Stability	See below.

The invocation is Uncommitted. The options are Uncommitted. The output is Not-an-Interface.

See Also `allocate(1)`, `deallocate(1)`, `add_allocatable(1M)`, `attributes(5)`, `device_clean(5)`

Notes The functionality described on this manual page is available only if the system is configured with Trusted Extensions.

Name removef – remove a file from software database

Synopsis

```
removef [ [-M] -R root_path] [-V fs_file] pkginst path...
removef [ [-M] -R root_path] [-V fs_file] -f pkginst
```

Description `removef` informs the system that the user, or software, intends to remove a pathname. Output from `removef` is the list of input pathnames that may be safely removed (no other packages have a dependency on them).

Options The following options are supported:

`-f`
After all files have been processed, `removef` should be invoked with the `-f` option to indicate that the removal phase is complete.

`-M`
Instruct `removef` not to use the $*root_path*`/etc/vfstab` file for determining the client's mount points. This option assumes the mount points are correct on the server and it behaves consistently with Solaris 2.5 and earlier releases.

`-R` *root_path*
Define the full path name of a directory to use as the *root_path*. All files, including package system information files, are relocated to a directory tree starting in the specified *root_path*. The *root_path* may be specified when installing to a client from a server (for example, `/export/root/client1`).

`removef` inherits the value of the `PKG_INSTALL_ROOT` environment variable. (See ENVIRONMENT VARIABLES, below.) If `PKG_INSTALL_ROOT` is set, such as when the `-R` option is used with `pkgadd(1M)` or `pkgrm(1M)`

Note – The root file system of any non-global zones must not be referenced with the `-R` option. Doing so might damage the global zone's file system, might compromise the security of the global zone, and might damage the non-global zone's file system. See `zones(5)`.

`-V` *fs_file*
Specify an alternative *fs_file* to map the client's file systems. For example, used in situations where the $*root_path*`/etc/vfstab` file is non-existent or unreliable.

Operands The following operands are supported:

path
The pathname to be removed.

pkginst
The package instance from which the pathname is being removed.

Examples

EXAMPLE 1 Using `removef`

The following example uses the `removef` command in an optional pre-install script:

```
echo "The following files are no longer part of this package
      and are being removed."
removef $PKGINST /myapp/file1 /myapp/file2 |
while read pathname
do
     echo "$pathname"
     rm -f $pathname
done
removef -f $PKGINST || exit 2
```

Environment Variables

`removef` inherits the value of the following environment variable. This variable is set when `pkgadd(1M)` or `pkgrm(1M)`

`PKG_INSTALL_ROOT`
: If present, defines the full path name of a directory to use as the system's `PKG_INSTALL_ROOT` path. All product and package information files are then looked for in the directory tree, starting with the specified `PKG_INSTALL_ROOT` path. If not present, the default system path of `/` is used.

Exit Status

`0`
: Successful completion.

`>0`
: An error occurred.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also

`pkginfo(1)`, `pkgmk(1)`, `pkgparam(1)`, `pkgproto(1)`, `pkgtrans(1)`, `installf(1M)`, `pkgadd(1M)`, `pkgask(1M)`, `pkgchk(1M)`, `pkgrm(1M)`, `attributes(5)`, `largefile(5)`

Application Packaging Developer's Guide

Notes

Package commands are `largefile(5)`-aware. They handle files larger than 2 GB in the same way they handle smaller files. In their current implementations, `pkgadd(1M)`, `pkgtrans(1)` and other package commands can process a datastream of up to 4 GB.

Name reparsed – reparse point daemon

Synopsis

```
/usr/lib/reparse/reparsed
```

Description The `reparsed` daemon processes kernel upcalls to interpret "reparse points", which are the basis of Microsoft DFS referrals and NFS referrals support on Solaris SMB and NFS file servers. Only the root user or a user with equivalent privileges can run this daemon. The daemon is under SMF control.

Exit Status

`0`
Daemon started successfully.

`>0`
Daemon failed to start.

Error information is reported to `syslog` at level `LOG_ERR`.

Files

`/usr/lib/reparsed/*.so.1`
Per-service plugins for `reparsed`.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os
Interface Stability	Consolidation Private

See Also `nfsref(1M)`, `svcadm(1M)`, `syslogd(1M)`, `libreparse(3LIB)`, `attributes(5)`

Notes Do not manually stop, start or restart the `reparsed` daemon. If you need to change the state of the daemon, use these commands:

```
# svcadm disable svc:/system/filesystem/reparse
# svcadm enable svc:/system/filesystem/reparse
# svcadm restart svc:/system/filesystem/reparse
```

See `svcadm(1M)` for additional information.

Name repquota – summarize quotas for a ufs file system

Synopsis

```
repquota [-v] filesystem...

repquota -a [-v]
```

Description `repquota` prints a summary of the disk usage and quotas for the specified ufs file systems. The current number of files and amount of space (in kilobytes) is printed for each user along with any quotas created with `edquota(1M)`.

The *filesystem* must have the file `quotas` in its root directory.

Only the super-user may view quotas which are not their own.

Options The following options are supported:

`-a` Report on all mounted ufs file systems that have `rq` in the `mntopts` field of the `/etc/vfstab` file.

`-v` Report quotas for all users, even those who do not consume resources.

Usage See `largefile(5)` for the description of the behavior of `repquota` when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also `edquota(1M)`, `quota(1M)`, `quotacheck(1M)`, `quotaon(1M)`, `attributes(5)`, `largefile(5)`, `quotactl(7I)`

Name rmmount – removable media mounter for CD-ROM, Jaz drive, and others

Synopsis

```
/usr/sbin/rmmount [-D]
```

Description The `rmmount` utility is a removable media mounter that is executed by volume management whenever a removable medium, such as a CD-ROM is inserted. Removable media is managed by an application or a volume manager. `rmmount` can also be called by using `volrmmount(1)`.

Upon insertion of a medium and following invocation of the `volcheck(1)` command, `rmmount` determines what type of file system (if any) is on that medium. If a file system is present, `rmmount` mounts the file system in one of the locations listed below.

For a CD-ROM or a DVD-ROM:

`/cdrom/cdrom0`	symbolic link to mounted CD-ROM in local CD-ROM drive
`/cdrom/CD-ROM_name`	mounted named CD-ROM
`/cdrom/CD-ROM_name/partition`	mounted named CD-ROM with partitioned file system
`/cdrom/unnamed_cdrom`	mounted unnamed CD-ROM

For a Zip drive:

`/rmdisk/zip0`	symbolic link to mounted Zip medium in local Zip drive
`/rmdisk/`*Zip_name*	mounted named Zip medium
`/rmdisk/`*Zip_name*`/partition`	mounted named Zip medium with partitioned file system
`/rmdisk/unnamed_zip`	mounted unnamed Zip medium

For a Jaz drive:

`/rmdisk/jaz0`	symbolic link to mounted Jaz medium in local Jaz drive
`/rmdisk/`*Jaz_name*	mounted named Jaz medium
`/rmdisk/`*Jaz_name*`/partition`	mounted named Jaz medium with partitioned file system
`/rmdisk/unnamed_Jaz`	mounted unnamed Jaz medium

For a generic “rmdisk” drive:

`/rmdisk/rmdisk0`	symbolic link to mounted removable medium in local removable medium drive
`/rmdisk/`*rmdisk_name*	mounted named removable medium
`/rmdisk/`*rmdisk_name*`/partition`	mounted named removable medium with partitioned file system

`/rmdisk/unnamed_rmdisk` mounted unnamed removable medium

If the media is read-only (for example, a CD-ROM), the file system is mounted read-only.

If a file system is not identified, `rmmount` does not mount a file system. See the *Oracle Solaris Administration: Common Tasks* for more information on the location of CD-ROM and other media without file systems.

If a file system type has been determined, it is then checked to see that it is "clean." If the file system is "dirty," `fsck -p` (see `fsck(1M)`) is run in an attempt to clean it. If `fsck` fails, the file system is mounted read-only.

After the mount is complete, "actions" associated with the media type are executed. These actions allow for the notification to other programs that new media are available.

Actions are executed in the order in which they appear in the configuration file. The action function can return either `1` or `0`. If it returns `0`, no further actions will be executed. This allows the function to control which applications are executed.

In order to execute an action, `rmmount` performs a `dlopen(3C)` on the shared object and calls the action function defined within it. The definition of the interface to actions can be found in `/usr/include/rmmount.h`.

File systems mounted by `rmmount` are always mounted with the `nosuid` flag set, thereby disabling setuid programs and access to block or character devices in that file system. Upon ejection, `rmmount` unmounts mounted file systems and executes actions associated with the media type. If a file system is "busy" (that is, it contains the current working directory of a live process), the ejection will fail.

Options `-D` Turn on the debugging output from the `rmmount` `dprintf` calls.

Files `/usr/lib/rmmount/*.so.1` shared objects used by `rmmount`.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/storage/media-volume-manager

See Also `volcheck(1)`, `volrmmount(1)`, `fsck(1M)`, `dlopen(3C)`, `attributes(5)`

Oracle Solaris Administration: Common Tasks

Name rmt – remote magtape protocol module

Synopsis `/usr/sbin/rmt`

Description `rmt` is a program used by the remote dump and restore programs in manipulating a magnetic tape drive through an interprocess communication connection. `rmt` is normally started up with an `rexec(3SOCKET)` or `rcmd(3SOCKET)` call.

The `rmt` program accepts requests that are specific to the manipulation of magnetic tapes, performs the commands, then responds with a status indication. All responses are in ASCII and in one of two forms. Successful commands have responses of:

`A`*number*`\n` where *number* is an ASCII representation of a decimal number.

Unsuccessful commands are responded to with:

`E`*error-number*`\n`*error-message*`\n` where *error-number* is one of the possible error numbers described in `Intro(3)`, and *error-message* is the corresponding error string as printed from a call to `perror(3C)`.

The protocol consists of the following commands:

`S\n` Return the status of the open device, as obtained with a `MTIOCGET` `ioctl` call. If the operation was successful, an "ack" is sent with the size of the status buffer, then the status buffer is sent (in binary).

`C`*device*`\n` Close the currently open device. The *device* specified is ignored.

`I`*operation*`\n`*count*`\n` Perform a `MTIOCOP` `ioctl(2)` command using the specified parameters. The parameters are interpreted as the ASCII representations of the decimal values to place in the *mt_op* and *mt_count* fields of the structure used in the `ioctl` call. When the operation is successful the return value is the *count* parameter.

`L`*offset*`\nwhence\n` Perform an `lseek(2)` operation using the specified parameters. The response value is returned from the `lseek` call.

`O`*device*`\n`*mode*`\n` Open the specified *device* using the indicated *mode*. *device* is a full pathname, and *mode* is an ASCII representation of a decimal number suitable for passing to `open(9E)`. If a device is already open, it is closed before a new open is performed.

`R`*count*`\n` Read *count* bytes of data from the open device. `rmt` performs the requested `read(9E)` and responds with `A`*count-read*`\n` if the read was successful; otherwise an error in standard format is returned. If the read was successful, the data read is sent.

`W`*count*`\n` Write data onto the open device. `rmt` reads *count* bytes from the connection, aborting if a premature EOF is encountered. The

response value is returned from the write(9E) call.

Any other command causes rmt to exit.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	service/network/network-clients

See Also ufsdump(1M), ufsrestore(1M), Intro(3), ioctl(2), lseek(2), perror(3C), rcmd(3SOCKET), rexec(3SOCKET), attributes(5), mtio(7I), open(9E), read(9E), write(9E)

Diagnostics All responses are of the form described above.

Bugs Do not use this for a remote file access protocol.

Name rmvolmgr – HAL-aware removable volume manager

Synopsis

```
/usr/lib/rmvolmgr [-chns]
```

Description The `rmvolmgr` command is a volume manager that can automatically mount and unmount removable media and hot-pluggable storage. The default mount point is `/media`.

`rmvolmgr` is one of a number of Hardware Abstraction Layer (HAL)-aware tools that are shipped with the Solaris operating system. See `hald(1M)`.

Multiple instances of `rmvolmgr` can be run simultaneously. A system instance of `rmvolmgr` runs by default as a service management facility (SMF) service (see `smf(5)`). Its fault management resource identifier (FMRI) is:

```
svc:/system/filesystem/rmvolmgr
```

You can run your own instance of `rmvolmgr` by adding it to the `.xinitrc` file or a similar session startup script. In such a case, the *system* `rmvolmgr` instance will not manage volumes that belong to you, the owner of the startup script. For example, a user logged on to the workstation console (`/dev/console`) who invokes his own instance of `rmvolmgr` will own locally connected devices, such as CD-ROM drives and devices connected to the local USB or FireWire ports.

In addition to mounting volumes under `/media`, `rmvolmgr` also creates legacy symbolic links under `/cdrom` and `/rmdisk`.

`rmvolmgr` also provides backwards compatibility with CDE removable media interfaces by maintaining notification files under `/tmp/.removable`. This functionality can be disabled by using the `-c` option.

The `-c` and `-n` options can also be specified as SMF properties. See "Examples."

Options The following options are supported:

`-c` Disable CDE compatibility.

`-h` Display help information and exit.

`-n` Do not create legacy mountpoint symbolic links.

`-s` Invoke in system instance mode.

Examples **EXAMPLE 1** Using SMF Properties to Set Options

The following `svccfg(1M)` command and subcommands use SMF properties to set the `-c` and `-n` options.

```
example# svccfg
svc:> select rmvolmgr
svc:/system/filesystem/rmvolmgr> listprop rmvolmgr/*
rmvolmgr/legacy_mountpoints     boolean  true
```

EXAMPLE 1 Using SMF Properties to Set Options *(Continued)*

```
rmvolmgr/cde_compatible         boolean  true
svc:/system/filesystem/rmvolmgr> setprop \
  rmvolmgr/legacy_mountpoints=false
svc:/system/filesystem/rmvolmgr> setprop rmvolmgr/cde_compatible=false
svc:/system/filesystem/rmvolmgr> exit
example#
```

Attributes

See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/storage/media-volume-manager
Interface Stability	Volatile

See Also

hald(1M), svccfg(1M), attributes(5), hal(5), smf(5)

Name rndc – name server control utility

Synopsis

```
rndc [-V] [-b src-addr] [-c config-file] [-k key-file] [-s server]
     [-p port] [-y key_id] command
```

Description The `rndc` utility controls the operation of a name server. It supersedes the `ndc` utility that was provided in previous BIND releases. If `rndc` is invoked with no command line options or arguments, it prints a short summary of the supported commands and the available options and their arguments.

The `rndc` utility communicates with the name server over a TCP connection, sending commands authenticated with digital signatures. The only supported authentication algorithm in the current versions of `rndc` and `named(1M)` is HMAC-MD5, which uses a shared secret on each end of the connection. This algorithm provides TSIG-style authentication for the command request and the name server's response. All commands sent over the channel must be signed by a *key_id* known to the server.

The `rndc` utility reads a configuration file to determine how to contact the name server and decide what algorithm and key it should use.

Options The following options are supported:

`-b` *source-address*
Use *source-address* as the source address for the connection to the server. Multiple instances are permitted to allow setting of both the IPv4 and IPv6 source addresses.

`-c` *config-file*
Use *config-file* as the configuration file instead of the default `/etc/rndc.conf`.

`-k` *key-file*
Use *key-file* as the key file instead of the default, `/etc/rndc.key`. The key in `/etc/rndc.key` is used to authenticate commands sent to the server if the *config-file* does not exist.

`-s` *server*
The *server* argument is the name or address of the server that matches a server statement in the configuration file for `rndc`. If no server is supplied on the command line, the host named by the default-server clause in the options statement of the `rndc` configuration file is used.

`-p` *port*
Send commands to TCP port *port* instead of BIND 9's default control channel port, 953.

`-V`
Enable verbose logging.

`-y` *key_id*
Use the key *key_id* from the configuration file. The *key_id* argument must be known by `named` with the same algorithm and secret string for control message validation to succeed. If no *key_id* is specified, `rndc` will first look for a key clause in the server statement of the

server being used, or if no server statement is present for that host, then the `default-key` clause of the options statement. The configuration file contains shared secrets that are used to send authenticated control commands to name servers. It should therefore not have general read or write access.

For the complete set of commands supported by `rndc`, see the *BIND 9 Administrator Reference Manual* or run `rndc` without arguments to see its help message.

Limitations The `rndc` utility does not support all the commands of the BIND 8 `ndc` utility.

There is no way to provide the shared secret for a *key_id* without using the configuration file.

Several error messages tend toward the cryptic.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	network/dns/bind
Interface Stability	Volatile

See Also `named(1M)`, `rndc-confgen(1M)`, `named.conf(4)`, `rndc.conf(4)`, `attributes(5)`

See the BIND 9 *Administrator's Reference Manual*. As of the date of publication of this man page, this document is available at `https://www.isc.org/software/bind/documentation`.

Name rndc-confgen – rndc key generation tool

Synopsis

```
rndc-confgen [-ah] [-b keysize] [-c keyfile] [-k keyname]
     [-p port] [-r randomfile] [-s address] [-t chrootdir]
     [-u user]
```

Description The `rndc-confgen` utility generates configuration files for `rndc(1M)`. This utility can be used as a convenient alternative to writing by hand the `rndc.conf(4)` file and the corresponding `controls` and `key` statements in `named.conf`. It can also be run with the `-a` option to set up a `rndc.key` file and avoid altogether the need for a `rndc.conf` file and a `controls` statement.

Options The following options are supported:

`-a`
Perform automatic `rndc` configuration. This option creates a file `rndc.key` in `/etc` (or however *sysconfdir* was specified when BIND was built) that is read by both `rndc` and `named(1M)` on startup. The `rndc.key` file defines a default command channel and authentication key allowing `rndc` to communicate with `named` with no further configuration.

Running `rndc-confgen` with `-a` specified allows BIND 9 and `rndc` to be used as drop-in replacements for BIND 8 and `ndc`, with no changes to the existing BIND 8 `named.conf` file.

If a more elaborate configuration than that generated by `rndc-confgen -a` is required, for example if `rndc` is to be used remotely, you should run `rndc-confgen` without the `-a` option and set up `rndc.conf` and `named.conf` files, as directed.

`-b` *keysize*
Specify the size of the authentication key in bits. The *keysize* argument must be between 1 and 512 bits; the default is 128.

`-c` *keyfile*
Used with the `-a` option to specify an alternate location for `rndc.key`.

`-h`
Print a short summary of the options and arguments to `rndc-confgen`.

`-k` *keyname*
Specify the key name of the `rndc` authentication key. The *keyname* argument must be a valid domain name. The default is `rndc-key`.

`-p` *port*
Specify the command channel port where `named` listens for connections from `rndc`. The default is 953.

`-r` *randomfile*
Specify a source of random data for generating the authorization. By default, `/dev/random` is used. The *randomdev* argument specifies the name of a character device or file containing random data to be used instead of the default. The special value `keyboard` indicates that keyboard input should be used.

`-s` *address*
Specify the IP address where `named` listens for command channel connections from `rndc`. The default is the loopback address 127.0.0.1.

`-t` *chrootdir*
Used with the `-a` option to specify a directory where `named` will run after the root directory is changed with `chroot(2)`. An additional copy of the `rndc.key` will be written relative to this directory so that it will be found by the `named` in the new directory.

`-u` *user*
Used with the `-a` option to set the owner of the `rndc.key` file generated. If `-t` is also specified only the file in the chroot area has its owner changed.

Examples

EXAMPLE 1 Create Automatic rndc Configuration

The following command creates an automatic `rndc` configuration, so that `rndc` can be used immediately.

```
# rndc-confgen -a
```

EXAMPLE 2 Print a Sample `rndc.conf` File

The following command prints a sample `rndc.conf` file with corresponding `controls` and `key` statements. These statements can subsequently be manually inserted in the file `named.conf`.

```
# rndc-confgen
```

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	network/dns/bind
Interface Stability	Volatile

See Also

`chroot(2)`, `named(1M)`, `rndc(1M)`, `rndc.conf(4)`, `attributes(5)`

See the BIND 9 *Administrator's Reference Manual*. As of the date of publication of this man page, this document is available at `https://www.isc.org/software/bind/documentation`.

Name roleadd – administer a new role account on the system

Synopsis

```
roleadd [-c comment] [-d dir] [-e expire] [-f inactive]
     [-g group] [-G group [, group...]] [-m [-k skel_dir]]
     [-u uid [-o]] [-s shell] [-S repository]
     [-A authorization [,authorization...]] [-K key=value] role

roleadd -D [-b base_dir] [-e expire] [-f inactive]
     [-g group] [-A authorization [,authorization...]]
     [-P profile [,profile...] [-K key=value]]
```

Description `roleadd` adds a role entry to the `passwd` and `shadow` and `user_attr` databases specified by the `-S` option. The default repository is `files`. The `-A` and `-P` options respectively assign authorizations and profiles to the role. Roles cannot be assigned to other roles. The `-K` option adds a *key=value* pair to `user_attr` for a role. Multiple *key=value* pairs can be added with multiple `-K` options.

`roleadd` also creates supplementary group memberships for the role (`-G` option) and creates the home directory (`-m` option) for the role if requested. The new role account remains locked until the `passwd(1)` command is executed.

Specifying `roleadd -D` with the `-g`, `-b`, `-f`, `-e`, or `-K` option (or any combination of these option) sets the default values for the respective fields. See the `-D` option. Subsequent `roleadd` commands without the `-D` option use these arguments.

The system file entries created with this command have a limit of 512 characters per line. Specifying long arguments to several options can exceed this limit.

The role (`role`) field accepts a string of no more than eight bytes consisting of characters from the set of alphabetic characters, numeric characters, period (`.`), underscore (`_`), and hyphen (`-`). The first character should be alphabetic and the field should contain at least one lower case alphabetic character. A warning message is written if these restrictions are not met. A future Solaris release might refuse to accept role fields that do not meet these requirements.

The `role` field must contain at least one character and must not contain a colon (`:`) or a newline (`\n`).

An administrator must be granted the User Management Profile to be able to create a new role. The authorizations required to set the various fields in `passwd`, `shadow` and `user_attr` can be found in `passwd(4)`, `shadow(4)`, and `user_attr(4)`. The authorizations required to assign groups can be found in `group(4)`.

Options The following options are supported:

`-A` *authorization*
: One or more comma separated authorizations defined in `auth_attr(4)`. Only a user or role who has grant rights to the authorization can assign it to an account

`-b` *base_dir*
The default base directory for the system if `-d` *dir* is not specified. *base_dir* is concatenated with the account name to define the home directory. If the `-m` option is not used, *base_dir* must exist.

`-c` *comment*
Any text string. It is generally a short description of the role. This information is stored in the role's `passwd` entry.

`-d` *dir* | *server*:*dir*
Specifies the home directory path for the new role. If no server name is specified, the specified directory is maintained in the `passwd(4)` database.

The optional server name specifies the host on which the home directory resides. Entries in this form depend on the automounter, and are maintained in the `auto_home` map. The path `/home/`*username* is maintained in the `passwd(4)` database. When the user subsequently references `/home/`*username*, the automounter will mount the specified directory on `/home/`*username*.

`-D`
Display the default values for *group*, *base_dir*, *skel_dir*, *shell*, *inactive*, *expire* and *key=value* pairs. When used with the `-g`, `-b`, `-f`, or `-K`, options, the `-D` option sets the default values for the specified fields. The default values are:

group
`other` (GID of 1)

base_dir
`/export/home`

skel_dir
`/etc/skel`

shell
`/bin/pfsh`

inactive
`0`

expire
Null

auths
Null

profiles
Null

key=value (pairs defined in `user_attr(4)`
not present

`-e` *expire*
Specify the expiration date for a role. After this date, no user is able to access this role. The expire option argument is a date entered using one of the date formats included in the template file `/etc/datemsk`. See `getdate(3C)`.

If the date format that you choose includes spaces, it must be quoted. For example, you can enter `10/6/90` or `October 6, 1990`. A null value (`""`) defeats the status of the expired date. This option is useful for creating temporary roles.

`-f` *inactive*
The maximum number of days allowed between uses of a role ID before that ID is declared invalid. Normal values are positive integers. A value of `0` defeats the status.

`-g` *group*
An existing group's integer ID or character-string name. Without the `-D` option, it defines the new role's primary group membership and defaults to the default group. You can reset this default value by invoking `roleadd -D -g` *group*.

`-G` *group*
An existing group's integer ID or character-string name. It defines the new role's supplementary group membership. Duplicates between *group* with the `-g` and `-G` options are ignored. No more than `NGROUPS_MAX` groups can be specified.

`-k` *skel_dir*
A directory that contains skeleton information (such as `.profile`) that can be copied into a new role's home directory. This directory must already exist. The system provides the `/etc/skel` directory that can be used for this purpose.

`-K` *key=value*
A *key=value* pair to add to the role's attributes. Multiple `-K` options can be used to add multiple *key=value* pairs. The generic `-K` option with the appropriate key can be used instead of the specific implied key options (`-A` and `-P`). See `user_attr(4)` for a list of valid *key=value* pairs. The "type" key is not a valid key for this option. Keys can not be repeated.

`-m`
Create the new role's home directory if it does not already exist. If the directory already exists, it must have read, write, and execute permissions by *group*, where *group* is the role's primary group. If the server name specified to the `-d` option is a remote host then the system will not attempt to create the home directory.

If the directory does not already exist and the parent directory is the mount point of a ZFS dataset, then a child of that dataset will be created and mounted at the specified location. The role is delegated permissions to create ZFS snapshots and promote them. The newly created dataset will inherit the encryption setting from its parent. If it is encrypted, the role is granted permission to change its wrapping key.

`-o`
This option allows a UID to be duplicated (non-unique).

`-P` *profile*
One or more comma-separated execution profiles defined in `prof_attr(4)`.

`-s` *shell*
Full pathname of the program used as the user's shell on login. It defaults to an empty field causing the system to use `/bin/pfsh` as the default. The value of *shell* must be a valid executable file.

`-S` *repository*
The valid repositories are `files`, `ldap`. The repository specifies which name service will be updated. The default repository is `files`. When the repository is `files`, the authorizations, profiles, and roles can be present in other name service repositories and can be assigned to a user in the `files` repository. When the repository is `ldap`, all the assignable attributes must be present in the `ldap` repository.

`-u` *uid*
The UID of the new role. This UID must be a non-negative decimal integer below `MAXUID` as defined in `<sys/param.h>`. The UID defaults to the next available (unique) number above the highest number currently assigned. For example, if UIDs 100, 105, and 200 are assigned, the next default UID number is 201. (UIDs from `0`-`99` are reserved for possible use in future applications.)

Exit Status In case of an error, `roleadd` prints an error message and exits with one of the following values:

`1` No permission for attempted operation.

`2` The command syntax was invalid. A usage message for the `usermod` command is displayed.

`3` An invalid argument was provided to an option.

`4` The *gid* or *uid* given with the `-u` option is already in use.

`5` The `password` and `shadow` files are not consistent with each other. `pwconv(1M)` might be of use to correct possible errors. See `passwd(4)` and `shadow(4)`.

`6` The login to be modified does not exist, the *gid* or the *uid* does not exist.

`7` The `group`, `passwd`, or `shadow` file is missing.

`9` A group or user name is already in use.

`10` Cannot update the `passwd`, `shadow`, or `user_attr` file.

`11` Insufficient space to move the home directory (`-m` option).

`12` Unable to create, remove, or move the new home directory.

`13` Requested login is already in use.

`14` Unexpected failure.

`16` Unable to update the group database.

17 Unable to update the project database.

18 Insufficient authorization.

19 Does not have role.

20 Does not have profile.

21 Does not have privilege.

22 Does not have label.

23 Does not have group.

24 System not running Trusted Extensions.

25 Does not have project.

26 Unable to update `auto_home`.

Files `/etc/datemsk`

`/etc/passwd`

`/etc/shadow`

`/etc/group`

`/etc/skel`

`/usr/include/limits.h`

`/etc/user_attr`

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os
Interface Stability	Committed

See Also `auths(1)`, `passwd(1)`, `pfexec(1)`, `profiles(1)`, `roles(1)`, `users(1B)`, `groupadd(1M)`, `groupdel(1M)`, `groupmod(1M)`, `grpck(1M)`, `logins(1M)`, `pwck(1M)`, `userdel(1M)`, `usermod(1M)`, `getdate(3C)`, `auth_attr(4)`, `group(4)`, `passwd(4)`, `prof_attr(4)`, `shadow(4)`, `user_attr(4)`, `attributes(5)`

Diagnostics In case of an error, `roleadd` prints an error message and exits with a non-zero status.

The following indicates that `login` specified is already in use:

```
UX: roleadd: ERROR: login is already in use. Choose another.
```

The following indicates that the *uid* specified with the `-u` option is not unique:

```
UX: roleadd: ERROR: uid uid is already in use. Choose another.
```

The following indicates that the *group* specified with the `-g` option is already in use:

```
UX: roleadd: ERROR: group group does not exist. Choose another.
```

The following indicates that the *uid* specified with the `-u` option is in the range of reserved UIDs (from `0`-`99`):

```
UX: roleadd: WARNING: uid uid is reserved.
```

The following indicates that the *uid* specified with the `-u` option exceeds `MAXUID` as defined in `<sys/param.h>`:

```
UX: roleadd: ERROR: uid uid is too big. Choose another.
```

The following indicates that the `/etc/passwd` or `/etc/shadow` files do not exist:

```
UX: roleadd: ERROR: Cannot update system files - login cannot be created.
```

The following indicates that the user executing the command does not have sufficient authorization to perform the operation:

```
UX: roleadd: ERROR: Permission denied.
```

Name roledel – delete a role's login from the system

Synopsis

```
roledel [-r] [-S repository] role
```

Description The `roledel` utility deletes a role account from the system and makes the appropriate account-related changes to the system file and file system. `roledel` also removes the role from each user's list of assumable roles.

An administrator must be granted the User Management Profile to be able to delete an existing role.

Options The following options are supported:

`-r`
Remove the role's home directory from the system. This directory must exist. The files and directories under the home directory will no longer be accessible following successful execution of the command. The ZFS dataset that was created for the role's home directory will be removed.

The `auto_home` entry for the role will be deleted.

`-S` *repository*
The valid repositories are `files`, `ldap`. The repository specifies which name service will be updated. The default repository is `files`.

Operands The following operands are supported:

role An existing role name to be deleted.

Exit Status The following exit values are returned:

`0` Successful completion.

`2` Invalid command syntax. A usage message for the `roledel` command is displayed.

`6` The account to be removed does not exist.

`8` The account to be removed is in use.

`10` Cannot update the `/etc/group` or `/etc/user_attr` file but the login is removed from the `/etc/passwd` file.

`12` Cannot remove or otherwise modify the home directory.

Files

`/etc/passwd`	system password file
`/etc/shadow`	system file containing roles' encrypted passwords and related information
`/etc/group`	system file containing group definitions
`/etc/user_attr`	system file containing additional role attributes

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also auths(1), passwd(1), profiles(1), roles(1), users(1B), groupadd(1M), groupdel(1M), groupmod(1M), logins(1M), roleadd(1M), rolemod(1M), useradd(1M), userdel(1M), usermod(1M), passwd(4), prof_attr(4), user_attr(4), attributes(5)

Notes The roledel utility only deletes an account definition that is in the group, passwd, shadow, and user_attr databases in the repository.

Name rolemod – modify a role's login information on the system

Synopsis

```
rolemod [-u uid [-o]] [-g group] [-G [+|-]group [, group...]]
     [-d dir [-m]] [-s shell] [-c comment] [-l new_name]
     [-f inactive] [-e expire]
     [-A [+|-]authorization [, authorization]]
     [-S repository]
     [-P [+|-]profile [, profile]] [-K key[+|-]=value] role
```

Description The `rolemod` utility modifies a role's login information on the system. It changes the definition of the specified login and makes the appropriate login-related system file and file system changes.

The system file entries created with this command have a limit of 512 characters per line. Specifying long arguments to several options may exceed this limit.

An administrator must be granted the User Security Profile to be able to modify the security attributes for an existing role. To be able to modify non-security attributes of an existing user requires the User Management Profile. The authorizations required to set the various fields in `passwd`, `shadow` and `user_attr` can be found in `passwd(4)`, `shadow(4)`, `user_attr(4)`. The authorizations required to assign groups can be found in `group(4)`.

Options The following options are supported:

`-A` [+|]*authorization*
: One or more comma separated authorizations as deined in `auth_attr(4)`. Only role with `grant` rights to the `authorization` can assign it to an account. This replaces any existing authorization setting. If no authorization list is specified, the existing setting is removed.

  A prefix `+` adds the authorization to the existing authorization; a prefix `-` removes the authorization from the existing authorization. With no prefix, the value for *authorization* replaces the existing authorization.

`-c` *comment*
: Specify a comment string. *comment* can be any text string. It is generally a short description of the login, and is currently used as the field for the user's full name. This information is stored in the user's `/etc/passwd` entry.

`-d` *dir*
: Specify the new home directory of the role. It defaults to *base_dir/login*, where *base_dir* is the base directory for new login home directories, and `login` is the new login. This creates or modifies an `auto_home` entry for the user.

  The argument to the option can be specified as *server*:*dir* where *server* is the hostname of the machine on which the home directory resides and *dir* is the path to the user's home directory. If the server is a remote host then the home directory needs to be created on the remote host for the system to mount it, when the user logs in. If no server name is specified then the home directory will be created on the host where the command is executed, when the `-m` option is used.

`-e` *expire*
Specify the expiration date for a role. After this date, no role will be able to access this login. The expire option argument is a date entered using one of the date formats included in the template file `/etc/datemsk`. See `getdate(3C)`.

For example, you may enter `10/6/90` or `October 6, 1990`. A value of "`''` defeats the status of the expired date.

`-f` *inactive*
Specify the maximum number of days allowed between uses of a login ID before that login ID is declared invalid. Normal values are positive integers. A value of `0` defeats the status.

`-g` *group*
Specify an existing group's integer ID or character-string name. It redefines the role's primary group membership.

`-G` [+|-]*group*
An existing group's integer ID or character-string name. It defines the new user's supplementary group membership. Duplicates between group with the `-g` and `-G` options are ignored. No more than `NGROUPS_MAX` groups can be specified. GIDs 0-99 are reserved for allocation by the Solaris Operating System.

A prefix + adds the group to the existing group; a prefix - removes the group from the existing group. With no prefix, *group* replaces the existing group.

`-K` *key*[+|-]=*value*
Replace existing or add to a role's *key*=*value* pair attributes. Multiple `-K` options can be used to replace or add multiple *key*=*value* pairs. However, keys must not be repeated. The generic `-K` option with the appropriate key may be used instead of the specific implied key options (`-A` and `-P`). See `user_attr(4)` for a list of valid *key*=*value* pairs. If no value is specified, the existing key is removed.

The keyword `type` can be specified with the value `role` or the value `normal`. When using the value `normal`, the account changes from a role user to a normal user; using the value `role` keeps the account a role user.

A prefix + adds the value to the existing value; a prefix - removes the value from the existing value. With no prefix, *value* replaces the existing value.

The prefix +/- operation is applicable only to the following keys: `auths`, `profiles`, `roles`, `project`, `limitpriv`, and `defaultpriv`.

`-l` *new_logname*
Specify the new login name for the role. The *new_logname* argument is a string no more than eight bytes consisting of characters from the set of alphabetic characters, numeric characters, period (`.`), underline (_), and hypen (–). The first character should be alphabetic and the field should contain at least one lower case alphabetic character. A warning message will be written if these restrictions are not met. A future Solaris release

may refuse to accept login fields that do not meet these requirements. The *new_logname* argument must contain at least one character and must not contain a colon (`:`) or NEWLINE (`\n`).

`-m`
Move the role's home directory to the new directory specified with the `-d` option. If the directory already exists, it must have permissions read/write/execute by *group*, where *group* is the role's primary group. If the server name specified to the `-d` option is a remote host then the system will not attempt to create the home directory.

If the directory does not already exist, a new ZFS dataset will be created. In the global zone, the dataset is created as `rpool/export/home/`*rolename*. For non-global zones, the dataset will be created as *ROOT-dataset*`/export/home/`*rolename*. The mountpoint for the ZFS dataset is `/export/home/`*rolename* by default. If `-d` *path* is specified and it is a path on the local machine, the dataset will be mounted at the specified location. The role is delegated permissions to create ZFS snapshots and promote them. The newly created dataset will inherit the encryption setting from its parent. If it is encrypted, the role is granted permission to change its wrapping key.

`-o`
This option allows the specified UID to be duplicated (non-unique).

`-P` [+|-]*profile*
One or more comma-separated execution profiles defined in `auth_attr(4)`. This replaces any existing profile setting. If no profile list is specified, the existing setting is removed.

A prefix `+` adds the profile to the existing profile; a prefix `-` removes the profile from the existing profile. With no prefix, *profile* replaces the existing profile.

`-s` *shell*
Specify the full pathname of the program that is used as the role's shell on login. The value of *shell* must be a valid executable file.

`-S` *repository*
The valid repositories are `files`, `ldap`. The repository specifies which name service will be updated. The default repository is `files`. When the repository is `files`, the authorizations, profiles, and roles can be present in other name service repositories and can be assigned to a user in the `files` repository. When the repository is `ldap`, all the assignable attributes must be present in the `ldap` repository.

`-u` *uid*
Specify a new UID for the role. It must be a non-negative decimal integer less than `MAXUID` as defined in `<param.h>`. The UID associated with the role's home directory is not modified with this option; a role will not have access to their home directory until the UID is manually reassigned using `chown(1)`.

Operands The following operands are supported:

`login`
An existing login name to be modified.

Examples **EXAMPLE 1** Setting Root Back to a Normal Account

The following command sets the root user back to a normal, non-privileged user.

```
# rolemod -K type=normal root
Found user in files repository.
```

Exit Status In case of an error, `rolemod` prints an error message and exits with one of the following values:

`1` No permission for attempted operation.

`2` The command syntax was invalid. A usage message for the `usermod` command is displayed.

`3` An invalid argument was provided to an option.

`4` The *gid* or *uid* given with the `-u` option is already in use.

`5` The `password` and `shadow` files are not consistent with each other. `pwconv(1M)` might be of use to correct possible errors. See `passwd(4)` and `shadow(4)`.

`6` The login to be modified does not exist, the *gid* or the *uid* does not exist.

`7` The `group`, `passwd`, or `shadow` file is missing.

`9` A group or user name is already in use.

`10` Cannot update the `passwd`, `shadow`, or `user_attr` file.

`11` Insufficient space to move the home directory (`-m` option).

`12` Unable to create, remove, or move the new home directory.

`13` Requested login is already in use.

`14` Unexpected failure.

`16` Unable to update the group database.

`17` Unable to update the project database.

`18` Insufficient authorization.

`19` Does not have role.

`20` Does not have profile.

`21` Does not have privilege.

`22` Does not have label.

23 Does not have group.

24 System not running Trusted Extensions.

25 Does not have project.

26 Unable to update `auto_home`.

Files

`/etc/group`
system file containing group definitions

`/etc/datemsk`
system file of date formats

`/etc/passwd`
system password file

`/etc/shadow`
system file containing users' and roles' encrypted passwords and related information

`/etc/user_attr`
system file containing additional user and role attributes

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os
Interface Stability	Committed

See Also `auths(1)`, `chown(1)`, `passwd(1)`, `profiles(1)`, `users(1B)`, `groupadd(1M)`, `groupdel(1M)`, `groupmod(1M)`, `logins(1M)`, `pwconv(1M)`, `roleadd(1M)`, `roledel(1M)`, `useradd(1M)`, `userdel(1M)`, `usermod(1M)`, `getdate(3C)`, `auth_attr(4)`, `group(4)`, `passwd(4)`, `shadow(4)`, `user_attr(4)`, `attributes(5)`

Name root_archive – manage bootable miniroot archives

Synopsis

```
/boot/solaris/bin/root_archive pack archive root
/boot/solaris/bin/root_archive unpack archive root
```

Description The `root_archive` utility is used to manipulate boot archives. The utility can pack and unpack boot archives and image miniroots. Both `ufs` and `hsfs` (`iso9660`) format archives can be unpacked, although only `ufs` format is generated when packing.

For normal, boot-related system administration, `bootadm(1M)` is recommended.

Subcommands The `root_archive` command has the following subcommands:

`pack` *archive root* Pack the contents of the root directory into the boot archive *archive*.

`unpack` *archive root* Unpack the contents of the boot archive named *archive* to the directory named *root*.

Exit Status The following exit values are returned:

`0` The command completed successfully.

`1` The command exited due to an error.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os
Interface Stability	Committed

See Also `cpio(1)`, `bootadm(1M)`, `mount(1M)`, `attributes(5)`, `lofi(7D)`

Name route – manually manipulate the routing tables

Synopsis

```
route [-fnvq] sub-command [ [modifiers] args]

route [-fnvq] [-p [-R root-dir]] add | delete [modifiers] destination gateway
     [args]

route [-fnvq] change | get [modifiers] destination
     [gateway [args]]

route [-fn] monitor [modifiers]

route [-fnvq] flush [modifiers]

route -p [-R root-dir] show
```

Description `route` manually manipulates the network routing tables. These tables are normally maintained by the system routing daemon, such as `in.routed(1M)` and `in.ripngd(1M)`.

`route` supports a limited number of general options, but a rich command language. Users can specify an arbitrary request that can be delivered by means of the programmatic interface discussed in `route(7P)`.

`route` uses a routing socket and the new message types `RTM_ADD`, `RTM_DELETE`, `RTM_GET`, and `RTM_CHANGE`. While only superusers can modify routing tables, the `RTM_GET` operation is allowed for non-privileged users.

Options

`-f` Flush the routing tables of all gateway entries. If you use the `-f` option in conjunction with any of the `route` sub-commands, `route` flushes the gateways before performing the sub-command. Specify the table to flush by placing the `inet` or `inet6` modifier immediately after the `-f` option. If unspecified, flushing IPv4 (`inet`) routes is the default.

`-n` Prevent attempts to print host and network names symbolically when reporting actions. This option is useful when name servers are unavailable.

`-p` Make changes to the network route tables persistent across system restarts. The operation is applied to the network routing tables first and, if successful, is then applied to the list of saved routes associated with the currently active Network Configuration Profile (refer to `netcfg(1M)` for more information about network configuration profiles). In determining whether an operation was successful, a failure to add a route that already exists or to delete a route that is not in the routing table is ignored. Particular care should be taken when using host or network names in persistent routes, as network-based name resolution services are not available at the time routes are added at startup.

`-q` Suppress all output.

`-R` *root-dir* Specify an alternate root directory where `route` applies changes. This option is ignored unless used in conjunction with the `-p` option. When `-R` is specified, route changes are applied only to the list of saved routes to be used at startup,

not to the network routing tables. In addition, certain checks, such as the existence of network interfaces used with `-ifp`, are skipped. This can be useful from within JumpStart scripts, where the root directory of the system being modified is in a location other than `/`.

`-v` Print additional details in verbose mode.

Subcommands

The following subcommands are supported:

`add` Add a route.

`change` Change aspects of a route (such as its gateway).

`delete` Delete a specific route.

`flush` Remove all gateway entries from the routing table.

`get` Look up and display the route for a destination.

`monitor` Continuously report any changes to the routing information base, routing lookup misses, or suspected network partitionings.

`show` Display the list of routes to be applied at system startup. Can be used only in conjunction with the `-p` option.

The `add` and `delete` sub-commands have the following syntax:

```
route [ -fnvq ] cmd destination gateway [metric/netmask]
```

where *cmd* is `add` or `delete`, *destination* is the destination host or network, and *gateway* is the next-hop intermediary through which packets should be routed. Modifiers described in `OPERANDS` can be placed anywhere on the command line.

The `get` and `change` sub-commands have the following syntax:

```
route [ -fnvq ] cmd destination [gateway [metric/netmask]]
```

where *cmd* is `get` or `change`, *destination* is the destination host or network, and *gateway* is the next-hop intermediary through which packets should be routed. Modifiers described in `OPERANDS` can be placed anywhere on the command line.

The `monitor` sub-command has the following syntax:

```
route monitor [ -inet | -inet6 ]
```

Operands

`route` executes its sub-commands on routes to destinations by way of gateways.

Destinations and Gateways

By default, destination and gateway addresses are interpreted as IPv4 addresses. All symbolic names are tried first as a host name, using `getipnodebyname(3SOCKET)`. If this lookup fails in the AF_INET case, `getnetbyname(3SOCKET)` interprets the name as that of a network.

Including an optional modifier on the command line before the address changes how the `route` sub-command interprets it.

The following modifiers are supported:

`-inet` Force the address to be interpreted as an IPv4 address, that is, under the `AF_INET` address family.

`-inet6` Force the address to be interpreted as an IPv6 address, that is, under the `AF_INET6` address family.

For IPv4 addresses, routes to a particular host are by default distinguished from those to a network by interpreting the Internet address specified as the destination. If the destination has a *local address part* (that is, the portion not covered by the netmask) of `0`, or if the destination is resolved as the symbolic name of a network, then the route is assumed to be to a network; otherwise, it is presumed to be a route to a host.

You can force this selection by using one of the following modifiers:

`-host` Force the destination to be interpreted as a host.

`-net` Force the destination to be interpreted as a network.

For example:

Destination	Destination Equivalent
`128.32`	`-host 128.0.0.32`
`128.32.130`	`-host 128.32.0.130`
`-net 128.32`	`128.32.0.0`
`-net 128.32.130`	`128.32.130.0`

Two modifiers avoid confusion between addresses and keywords (for example., `host` used as a symbolic host name). You can distinguish a *destination* by preceding it with the `-dst` modifier. You can distinguish a gateway address by using the `-gateway` modifier. If the destination is directly reachable by way of an interface requiring no intermediary IP router to act as a gateway, this can be indicated by using the `-interface` or `-iface` modifier.

In the following example, the route does not refer to an external gateway (router), but rather to one of the machine's interfaces. Packets with IP destination addresses matching the destination and mask on such a route are sent out on the interface identified by the gateway address. For interfaces using the ARP protocol, this type of route is used to specify that all matching destinations are local to the physical link. That is, a host could be configured to ARP for all addresses, without regard to the configured interface netmask, by adding a default route using this command. For example:

```
example# route add default hostname -interface
```

where gateway address *hostname* is the name or IP address associated with the network interface over which all matching packets should be sent. On a host with a single network interface, *hostname* is usually the same as the *nodename* returned by the `uname -n` command. See `uname(1)`.

For backward compatibility with older systems, directly reachable routes can also be specified by placing a `0` after the gateway address:

```
example# route add default hostname 0
```

This value was once a route metric, but this metric is no longer used. If the value is specified as `0`, then the destination is directly reachable (equivalent to specifying `-interface`). If it is non-zero but cannot be interpreted as a subnet mask, then a gateway is used (default).

With the `AF_INET` address family or an IPv4 address, a separate subnet mask can be specified. This can be specified in one of the following ways:

- IP address following the gateway address . This is typically specified in *decimal dot* notation as for `inet_addr(3SOCKET)` rather than in symbolic form.
- IP address following the `-netmask` qualifier.
- Slash character and a decimal length appended to the destination address.

If a subnet mask is not specified, the mask used is the subnet mask of the output interface selected by the gateway address, if the classful network of the destination is the same as the classful network of the interface. Otherwise, the classful network mask for the destination address is used.

Each of the following examples creates an IPv4 route to the destination `192.0.2.32` subnet with a subnet mask of `255.255.255.224`:

```
example# route add 192.0.2.32/27 somegateway
example# route add 192.0.2.32 -netmask 255.255.255.224 somegateway
example# route add 192.0.2.32 somegateway 255.255.255.224
```

For IPv6, only the slash format is accepted. The following example creates an IPv6 route to the destination `33fe::` with a netmask of `16` one-bits followed by 112 zero-bits.

```
example# route add -inet6 3ffe::/16 somegateway
```

In cases where the gateway does not uniquely identify the output interface (for example, when several interfaces have the same address), you can use the `-ifp` *ifname* modifier to specify the interface by name. For example, `-ifp lo0` associates the route with the `lo0` interface. If the named interface is an underlying interface in an IPMP (IP multipathing) group, then requests to add a route will automatically be translated to the corresponding IPMP IP interface, and requests to delete or change a route on an underlying interface will fail.

When the routing table contains several equal routes, that is, routes for the same destination and mask, then IP attempts to spread the traffic over those routes. The spreading is such that an individual transport connection uses the same route to avoid packet reordering as seen by, for example, TCP. The details of the spreading algorithm is not documented and is likely to evolve over time.

Routing Flags

Routes have associated flags that influence operation of the protocols when sending to destinations matched by the routes. These flags can be set (and in some cases cleared, indicated by ~) by including the following modifiers on the command line:

Modifier	Flag	Description
`-interface`	`~RTF_GATEWAY`	Destination is directly reachable
`-iface`	`~RTF_GATEWAY`	Alias for interface modifier
`-static`	`RTF_STATIC`	Manually added route
`-nostatic`	`~RTF_STATIC`	Pretend route was added by kernel or routing daemon
`-reject`	`RTF_REJECT`	Emit an ICMP unreachable when matched
`-blackhole`	`RTF_BLACKHOLE`	Silently discard packets
`-proto1`	`RTF_PROTO1`	Set protocol specific routing flag #1
`-proto2`	`RTF_PROTO2`	Set protocol specific routing flag #2
`-private`	`RTF_PRIVATE`	Do not advertise this route
`-multirt`	`RTF_MULTIRT`	Creates the specified redundant route
`-setsrc`	`RTF_SETSRC`	Assigns the default source address
`-indirect`	`RTF_INDIRECT`	Allows adding routes where gateway is not on-link

The optional `-indirect` modifier allows adding routes where the gateway is not directly reachable. When an indirect route is the best match for a packet to be sent or forwarded, then IP proceeds to look up that gateway to find a route that is directly reachable. The `-indirect` modifier can be used even if the gateway is directly reachable.

The optional modifiers `-rtt`, `-rttvar`, `-sendpipe`, `-recvpipe`, `-mtu`, `-hopcount`, `-expire`, and `-ssthresh` provide initial values to quantities maintained in the routing entry by transport level protocols, such as TCP. These can be individually locked either by preceding each modifier to be locked by the `-lock` meta-modifier, or by specifying that all ensuing metrics can be locked by the `-lockrest` meta-modifier.

Some transport layer protocols can support only some of these metrics. The following optional modifiers are supported:

`-expire` Lifetime for the entry. This optional modifier is not currently supported.

`-hopcount` Maximum hop count. This optional modifier is not currently supported.

`-mtu` Maximum MTU in bytes.

`-recvpipe` Receive pipe size in bytes.

`-rtt` Round trip time in microseconds.

`-rttvar` Round trip time variance in microseconds.

`-sendpipe` Send pipe size in bytes.

`-ssthresh` Send pipe size threshold in bytes.

`-secattr` Security attributes of the route. This modifier is available only if the system is configured with the Solaris Trusted Extensions feature.

The `-secattr` modifier has the following format:

`min_sl=`*val*`,max_sl=`*val*`,doi=`*val*`,cipso`

or:

`sl=VAL,doi=VAL,cipso`

In the first form, above, the *val* for `min_sl` and `max_sl` is a sensitivity label in either hex or string form. The *val* for `doi` is a non-negative integer. The route will apply only for packets with the same domain of interpretation as defined by the `doi` value and within the accreditation range defined by the `min_sl` and `max_sl` values. The `cipso` keyword is optional and set by default. Valid `min_sl`, `max_sl` and `doi` keyword/value pairs are mandatory. Note that if *val* contains a space, it must be protected by double quotes.

The second form, above, is equivalent to specifying the first form with the same `VAL` for `min_sl` and `max_sl`. The second form should be used for the `get` command, because `get` uses only a single sensitivity label.

Compatibility

The modifiers `host` and `net` are taken to be equivalent to `-host` and `-net`. To specify a symbolic address that matches one of these names, use the `dst` or `gateway` keyword to distinguish it. For example: `-dst host`

The following two flags are also accepted for compatibility with older systems, but have no effect.

Modifier	Flag
`-cloning`	RTF_CLONING
`-xresolve`	RTF_XRESOLVE

The `-ifa` `hostname` modifier is also accepted, but has no effect.

Files

`/etc/defaultrouter`	List of default routers
`/etc/hosts`	List of host names and net addresses
`/etc/networks`	List of network names and addresses

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also `uname(1)`, `in.ripngd(1M)`, `in.routed(1M)`, `netcfg(1M)`, `netstat(1M)`, `routed(1M)`, `ioctl(2)`, `getipnodebyname(3SOCKET)`, `getnetbyname(3SOCKET)`, `inet_addr(3SOCKET)`, `defaultrouter(4)`, `hosts(4)`, `networks(4)`, `attributes(5)`, `arp(7P)`, `ip(7P)`, `route(7P)`, `routing(7P)`

Diagnostics

`add` [`host` | `network`] *destination*:*gateway flags*
: The specified route is being added to the tables. The values printed are from the routing table entry supplied in the `ioctl(2)` call. If the gateway address used was not the primary address of the gateway (the first one returned by `getipnodebyname(3SOCKET)`) the gateway address is printed numerically as well as symbolically.

`delete` [`host` | `network`] *destination*:*gateway flags*
`change` [`host` | `network`] *destination*:*gateway flags*
: As `add`, but when deleting or changing an entry.

destination `done`
: When the `-f` flag is specified, or the `flush` sub-command is used, each routing table entry deleted is indicated with a message of this form.

`Network is unreachable`
: An attempt to add a route failed because the gateway listed was not on a directly-connected network. Give the next-hop gateway instead.

`not in table`
: A `delete` operation was attempted for an entry that is not in the table.

`entry exists`
: An `add` operation was attempted for a route that already exists in the kernel.

`routing table overflow`	An operation was attempted, but the system was unable to allocate memory to create the new entry.
`insufficient privileges`	An attempt to add, delete, change, or flush a route failed because the calling process does not have appropriate privileges.

Notes Specifying that destinations are local (with the `-interface`modifier) assumes that the routers implement `proxy ARP`, meaning that they respond to ARP queries for all reachable destinations. Normally, using either router discovery or RIP is more reliable and scalable than using proxy ARP. See `in.routed(1M)` for information related to RIP.

Combining the all destinations are local route with subnet or network routes can lead to unpredictable results. The search order as it relates to the all destinations are local route are undefined and can vary from release to release.

Name routeadm – IP forwarding and routing configuration

Synopsis

```
routeadm [-p [option]]

routeadm [-R root-dir] [-e option ...] [-d option...]
 [-r option...] [-s var=value]

routeadm [-l fmri]

routeadm [-m fmri key=value [key=value]...]

routeadm [-u]
```

Description The `routeadm` command is used to administer system-wide configuration for IP forwarding and routing. IP forwarding is the passing of IP packets from one network to another; IP routing is the use of a routing protocol to determine routes.

IP routing functions are also represented as services within the service management facility (SMF), and can be administered by means of `svcadm(1M)` also, using the following fault management resource identifiers (FMRIs):

```
svc:/network/routing/route:default
svc:/network/routing/ripng:default
```

See EXAMPLES for relevant examples.

In addition to enabling and disabling routing and forwarding, `routeadm` is used to interact with SMF-based routing daemon services. Routing daemon services are identified by the presence of a `routeadm` application property group, which `routeadm` uses in administering the given service. Routing daemon services can also specify properties relating to their operation in the `routing` application property group; these can be modified by means of `routeadm -m`. If an FMRI for a service without such a property group is specified, an error is issued and the operation is not carried out. If a routing daemon has not been converted to SMF, the `ipv4`*`[or 6]`*`-routing-daemon`, `ipv4`*`[or 6]`*`-routing-daemon-args`, and `ipv4`*`[or 6]`*`-routing-stop-cmd` variables can be used to specify the appropriate daemon for IPv4 or IPv6 routing. `routeadm` will then run that daemon using the `svc:/network/routing/legacy-routing:ipv4`*`[or 6]`* service as appropriate. This conversion process occurs when you issue an enable (`-e`), disable (`-d`) or an update (`-u`) command.

The first usage, in the SYNOPSIS above, reports the current configuration.

Options The following command-line options are supported:

`-p` [*option*]
: Print the configuration in parseable format. If *option* is specified, only the configuration for the specified option or variable is displayed.

`-R` *root-dir*
Specify an alternate root directory where `routeadm` applies changes. This can be useful from within JumpStart scripts, where the root directory of the system being modified is mounted elsewhere.

Note – The root file system of any non-global zones must not be referenced with the `-R` option. Doing so might damage the global zone's file system, might compromise the security of the global zone, and might damage the non-global zone's file system. See `zones(5)`.

`-e` *option...*
Enable the specified option. The effect is to prepare the associated services for enabling. By means of the `routing-svcs` variable, the routing daemons are specified to be enabled on subsequent boot or when `routeadm -u` is run.

`-d` *option...*
Disable the specified option. The effect is to prepare the associated services for disabling. By means of the `routing-svcs` variable, the routing daemons are specified to be disabled on subsequent boot or when `routeadm -u` is run.

`-l` *fmri*
List all properties in the routing application property group for the SMF routing daemon service.

`-m` *fmri key=value*
Change property value of property *key* to *value* in routing application property group for the SMF routing daemon service. For multi-valued properties, the property name can be used multiple times in the modify operation, and each associated value will be added.

`-r` *option...*
Revert the specified option to the system default. The system defaults are specified in the description of each *option*.

`-u`
Apply the currently configured options to the running system. These options might include enabling or disabling IP forwarding and launching or killing routing daemons, if any are specified. It does not alter the state of the system for those settings that have been set to default. This option is meant to be used by administrators who do not want to reboot to apply their changes. In addition, this option upgrades non-SMF configurations from the invocations of daemon stop commands, which might include a set of arguments, to a simple enabling of the appropriate service.

`-s` *key=value*
Specify string values for specific variables in a comma-separated list with no intervening spaces. If invalid options are specified, a warning message is displayed and the program exits. The following variables can be specified:

`routing-svcs=`*fmrilist*
Specifies the routing daemon services to be enabled. Routing daemon services are determined to be IPv4 or IPv6 (and so enabled or disabled when `routeadm -e/-d ipv4`*(6)*`-routing` is run) on the basis of property values in the `routeadm` application property group. Default: `route:default ripng:default`

`ipv4-routing-daemon=`<*full_path_to_routing_daemon*>
Specifies the routing daemon to be started when `ipv4-routing` is enabled. The routing daemon specified must be an executable binary or shell-script. If the specified program maps to an SMF service, the service will be used, and daemon arguments to the program will be transferred to the properties of the service at enable time. Default: ""

`ipv4-routing-daemon-args=`<*args*>
Specifies the startup arguments to be passed to the `ipv4-routing-daemon` when `ipv4-routing` is enabled. Default: no arguments

`ipv4-routing-stop-cmd=`<*command*>
Specifies the command to be executed to stop the routing daemon when `ipv4-routing` is disabled. <*command*> can be an executable binary or shell-script, or a string that can be parsed by `system`(3C). Default: ""

`ipv6-routing-daemon=`<*full_path_to_routing_daemon*>
Specifies the routing daemon to be started when `ipv6-routing` is enabled. The routing daemon specified must be an executable binary or shell-script. If the specified program maps to an SMF service, the service will be used, and daemon arguments to the program will be transferred to the properties of the service at enable time. Default: ""

`ipv6-routing-daemon-args=`<*args*>
Specifies the startup arguments to be passed to the `ipv6-routing-daemon` when `ipv6-routing` is enabled. Default: ""

`ipv6-routing-stop-cmd=`<*command*>
Specifies the command to be executed to stop the routing daemon when `ipv6-routing` is disabled. <*command*> can be an executable binary or shell-script, or a string that can be parsed by `system`(3C). Default: ""

Multiple `-e`, `-d`, and `-r` options can be specified on the command line. Changes made by `-e`, `-d`, and `-r` are persistent, but are not applied to the running system unless `routeadm` is called later with the `-u` option.

Use the following options as arguments to the `-e`, `-d`, and `-r` options (shown above as *option...*).

`ipv4-forwarding`
Controls the global forwarding configuration for all IPv4 interfaces. The system default is `disabled`. If enabled, IP will forward IPv4 packets to and from interfaces when appropriate. If disabled, IP will not forward IPv4 packets to and from interfaces when appropriate.

`ipv4-routing`
Determines whether an IPv4 routing daemon is run. The system default is `enabled` unless the `/etc/defaultrouter` file exists (see `defaultrouter(4)`), in which case the default is `disabled`. The value of this option reflects the state of all IPv4 routing services, such that if any IPv4 routing service is enabled, `ipv4-routing` is enabled. This allows users to interact with routing services using `svcadm(1M)`, as well as through `routeadm`. IPv4 routing services, specified by means of the `routing-svcs` variable, will be prepared for enable on next boot when the user explicitly enables `ipv4-routing`. The SMF routing daemon service for `in.routed` (`svc:/network/routing/route:default`) is specified by default.

`ipv6-forwarding`
Controls the global forwarding configuration for all IPv6 interfaces. The system default is `disabled`. If enabled, IP will forward IPv6 packets to and from interfaces when appropriate. If disabled, IP will not forward IPv6 packets to and from interfaces when appropriate.

`ipv6-routing`
Determines whether an IPv6 routing daemon is run. The system default is `disabled`. The value of this option reflects the state of all IPv6 routing services, such that, if any IPv6 routing service is enabled, `ipv6-routing` is enabled. This allows users to interact with routing services via `svcadm(1M)` as well as through `routeadm`. IPv6 routing services, specified by means of the `routing-svcs` variable, will be prepared for enable on next boot when the user explicitly enables `ipv6-routing`. The SMF routing daemon service for `in.ripngd` (`svc:/network/routing/ripng:default`) is specified by default.

The forwarding and routing settings are related but not mutually dependent. For example, a router typically forwards IP packets and uses a routing protocol, but nothing would prevent an administrator from configuring a router that forwards packets and does not use a routing protocol. In that case, the administrator would enable forwarding, disable routing, and populate the router's routing table with static routes.

The forwarding settings are global settings. Each interface also has an `IFF_ROUTER` forwarding flag that determines whether packets can be forwarded to or from a particular interface. That flag can be independently controlled by means of `ifconfig(1M)`'s router option. When the global forwarding setting is changed (that is, `-u` is issued to change the value from `enabled` to `disabled` or vice-versa), all interface flags in the system are changed simultaneously to reflect the new global policy. Interfaces configured by means of DHCP automatically have their interface-specific `IFF_ROUTER` flag cleared.

When a new interface is plumbed by means of `ifconfig`, the value of the interface-specific forwarding flag is set according to the current global forwarding value. Thus, the forwarding value forms the “default” for all new interfaces.

Examples

EXAMPLE 1 Enabling IPv4 Forwarding

IPv4 forwarding is disabled by default. The following command enables IPv4 forwarding:

```
example# routeadm -e ipv4-forwarding
```

EXAMPLE 2 Apply Configured Settings to the Running System

In the previous example, a system setting was changed, but will not take effect until the next reboot unless a command such as the following is used:

```
example# routeadm -u
```

EXAMPLE 3 Making a Setting Revert to its Default

To make the setting changed in the first example revert to its default, enter the following:

```
example# routeadm -r ipv4-forwarding
example# routeadm -u
```

EXAMPLE 4 Starting `in.routed` with the `-q` Flag

Setting the `-q` flag is represented in the SMF service by setting the `quiet_mode` property to true. The following sequence of commands starts `in.routed` with the `-q` flag:

```
example# routeadm -m route:default quiet_mode=true
example# routeadm -e ipv4-routing -u
```

See `in.routed(1M)` for details of property names and how they relate to daemon behavior.

Exit Status

The following exit values are returned:

`0` Successful completion.

`!=0` An error occurred while obtaining or modifying the system configuration.

Files

`/etc/inet/routing.conf` Parameters for IP forwarding and routing. (Not to be edited.)

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os
Interface Stability	Committed

See Also

`ifconfig(1M)`, `in.routed(1M)`, `ipadm(1M)`, `svcadm(1M)`, `gateways(4)`, `attributes(5)`, `smf(5)`

Name rpcbind – universal addresses to RPC program number mapper

Synopsis

```
rpcbind [-d] [-w]
```

Description `rpcbind` is a server that converts RPC program numbers into universal addresses. It must be running on the host to be able to make RPC calls on a server on that machine.

When an RPC service is started, it tells `rpcbind` the address at which it is listening, and the RPC program numbers it is prepared to serve. When a client wishes to make an RPC call to a given program number, it first contacts `rpcbind` on the server machine to determine the address where RPC requests should be sent.

`rpcbind` should be started before any other RPC service. Normally, standard RPC servers are started by port monitors, so `rpcbind` must be started before port monitors are invoked.

When `rpcbind` is started, it checks that certain name-to-address translation-calls function correctly. If they fail, the network configuration databases can be corrupt. Since RPC services cannot function correctly in this situation, `rpcbind` reports the condition and terminates.

`rpcbind` maintains an open transport end for each transport that it uses for indirect calls. This is the UDP port on most systems.

The `rpcbind` service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/network/rpc/bind
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. `rpcbind` can only be started by the superuser.

The configuration properties of this service can be modified with `svccfg(1M)`.

The following SMF property is used to allow or disallow access to `rpcbind` by remote clients:

```
config/local_only = true
```

The default value, `true`, shown above, disallows remote access; a value of `false` allows remove access. See EXAMPLES.

The FMRI `svc:network/rpc/bind` property group `config` contains the following property settings:

`enable_tcpwrappers` Specifies that the TCP wrappers facility is used to control access to TCP services. The value `true` enables checking. The default value for `enable_tcpwrappers` is `false`. If the `enable_tcpwrappers` parameter is enabled, then all calls to `rpcbind` originating from non-local addresses are automatically wrapped by the TCP wrappers facility. The `syslog` facility code daemon is used to log allowed connections (using the `info` severity level) and denied

traffic (using the `warning` severity level). See `syslog.conf(4)` for a description of `syslog` codes and severity levels. The Interface Stability of the TCP wrappers facility and its configuration files is Volatile. As the TCP wrappers facility is not controlled by Sun, intrarelease incompatibilities are not uncommon. See `attributes(5)`.

`verbose_logging` — Specifies whether the TCP wrappers facility logs all calls orjust the denied calls. The default is `false`. This option has no effect if TCP wrappers are not enabled.

`allow_indirect` — Specifies whether `rpcbind` allows indirect calls at all. By default, `rpcbind` allows most indirect calls, except to a number of standard services(`keyserv`, `automount`, `mount`, `nfs`, `rquota`, and selected NIS and `rpcbind` procedures). Setting `allow_indirect` to `false` causes all indirect calls to be dropped. The default is `true`. NIS broadcast clients rely on this functionality on NIS servers.

Options

The following options are supported:

`-d` — Run in debug mode. In this mode, `rpcbind` does not fork when it starts. It prints additional information during operation, and aborts on certain errors. With this option, the name-to-address translation consistency checks are shown in detail.

`-w` — Do a warm start. If `rpcbind` aborts or terminates on `SIGINT` or `SIGTERM`, it writes the current list of registered services to `/var/run/portmap.file` and `/var/run/rpcbind.file`. Starting `rpcbind` with the `-w` option instructs it to look for these files and start operation with the registrations found in them. This allows `rpcbind` to resume operation without requiring all RPC services to be restarted.

Examples

EXAMPLE 1 Allowing Remote Access

The following sequence of commands allows remote access to `rpcbind`.

```
# svccfg -s svc:/network/rpc/bind setprop config/local_only = false
# svcadm refresh svc:/network/rpc/bind
```

Files

`/var/run/portmap.file` — Stores the information for RPC services registered over IP based transports for warm start purposes.

`/var/run/rpcbind.file` — Stores the information for all registered RPC services for warm start purposes.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	See below.

TCP wrappers is Volatile.

See Also smf(5), rpcinfo(1M), svcadm(1M), svccfg(1M), rpcbind(3NSL), syslog.conf(4), attributes(5), smf(5)

For information on the TCP wrappers facility, see the `hosts_access(4)` man page, delivered as part of the Solaris operating environment in `/usr/sfw/man` and available in the `security/tcp-wrapper` package.

Notes Terminating `rpcbind` with `SIGKILL` prevents the warm-start files from being written.

All RPC servers are restarted if the following occurs: `rpcbind` crashes (or is killed with `SIGKILL`) and is unable to to write the warm-start files; `rpcbind` is started without the `-w` option after a graceful termination. Otherwise, the warm start files are not found by `rpcbind`.

Name rpc.bootparamd, bootparamd – boot parameter server

Synopsis

```
/usr/sbin/rpc.bootparamd [-d]
```

Description `rpc.bootparamd` is a server process that provides information from a `bootparams` database to diskless clients at boot time. See `bootparams(4)`

The source for the `bootparams` database is determined by the `nsswitch.conf(4)` file (on the machine running the `rpc.bootparamd` process).

Options The following options are supported:

`-d` Display debugging information.

Files

`/etc/bootparams`	boot parameter data base
`/etc/nsswitch.conf`	configuration file for the name-service switch

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/boot/network

See Also `svcs(1)`,`svcadm(1M)`, `bootparams(4)`, `nsswitch.conf(4)`, `attributes(5)`, `smf(5)`

Notes A diskless client requires service from at least one `rpc.bootparamd` process running on a server that is on the same IP subnetwork as the diskless client.

Some routines that compare hostnames use case-sensitive string comparisons; some do not. If an incoming request fails, verify that the case of the hostname in the file to be parsed matches the case of the hostname called for, and attempt the request again.

The `rpc.bootparamd` service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/network/rpc/bootparams
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. The service's status can be queried using the `svcs(1)` command.

Name rpcinfo – report RPC information

Synopsis

```
rpcinfo [-m | -s] [host]

rpcinfo -p [host]

rpcinfo -T transport host prognum [versnum]

rpcinfo -l [-T transport] host prognum versnum

rpcinfo [-n portnum] -u host prognum [versnum]

rpcinfo [-n portnum] -t host prognum [versnum]

rpcinfo -a serv_address -T transport prognum [versnum]

rpcinfo -b [-T transport] prognum versnum

rpcinfo -d [-T transport] prognum versnum
```

Description `rpcinfo` makes an RPC call to an RPC server and reports what it finds.

In the first synopsis, `rpcinfo` lists all the registered RPC services with `rpcbind` on *host*. If *host* is not specified, the local host is the default. If `-s` is used, the information is displayed in a concise format.

In the second synopsis, `rpcinfo` lists all the RPC services registered with `rpcbind`, version 2. Note that the format of the information is different in the first and the second synopsis. This is because the second synopsis is an older protocol used to collect the information displayed (version 2 of the `rpcbind` protocol).

The third synopsis makes an RPC call to procedure 0 of *prognum* and *versnum* on the specified *host* and reports whether a response was received. *transport* is the transport which has to be used for contacting the given service. The remote address of the service is obtained by making a call to the remote `rpcbind`.

The *prognum* argument is a number that represents an RPC program number (see `rpc(4)`).

If a *versnum* is specified, `rpcinfo` attempts to call that version of the specified *prognum*. Otherwise, `rpcinfo` attempts to find all the registered version numbers for the specified *prognum* by calling version 0, which is presumed not to exist; if it does exist, `rpcinfo` attempts to obtain this information by calling an extremely high version number instead, and attempts to call each registered version. Note that the version number is required for `-b` and `-d` options.

The `EXAMPLES` section describe other ways of using `rpcinfo`.

Options

`-T` *transport* — Specify the transport on which the service is required. If this option is not specified, `rpcinfo` uses the transport specified in the `NETPATH` environment variable, or if that is unset or `NULL`, the transport in the `netconfig(4)` database is used. This is a generic option, and can be used in conjunction with other options as shown in the `SYNOPSIS`.

`-a` *serv_address* Use *serv_address* as the (universal) address for the service on *transport* to `ping` procedure 0 of the specified *prognum* and report whether a response was received. The `-T` option is required with the `-a` option. If *versnum* is not specified, `rpcinfo` tries to ping all available version numbers for that program number. This option avoids calls to remote `rpcbind` to find the address of the service. The *serv_address* is specified in universal address format of the given transport.

`-b` Make an RPC broadcast to procedure 0 of the specified *prognum* and *versnum* and report all hosts that respond. If *transport* is specified, it broadcasts its request only on the specified transport. If broadcasting is not supported by any transport, an error message is printed. Use of broadcasting should be limited because of the potential for adverse effect on other systems.

`-d` Delete registration for the RPC service of the specified *prognum* and *versnum*. If *transport* is specified, unregister the service on only that transport, otherwise unregister the service on all the transports on which it was registered. Only the owner of a service can delete a registration, except the superuser, who can delete any service.

`-l` Display a list of entries with a given *prognum* and *versnum* on the specified *host*. Entries are returned for all transports in the same protocol family as that used to contact the remote `rpcbind`.

`-m` Display a table of statistics of `rpcbind` operations on the given *host*. The table shows statistics for each version of `rpcbind` (versions 2, 3 and 4), giving the number of times each procedure was requested and successfully serviced, the number and type of remote call requests that were made, and information about RPC address lookups that were handled. This is useful for monitoring RPC activities on *host*.

`-n` *portnum* Use *portnum* as the port number for the `-t` and `-u` options instead of the port number given by `rpcbind`. Use of this option avoids a call to the remote `rpcbind` to find out the address of the service. This option is made obsolete by the `-a` option.

`-p` Probe `rpcbind` on *host* using version 2 of the `rpcbind` protocol, and display a list of all registered RPC programs. If *host* is not specified, it defaults to the local host. This option is not useful for IPv6; use `-s` (see below) instead. Note that version 2 of the `rpcbind` protocol was previously known as the portmapper protocol.

`-s` Display a concise list of all registered RPC programs on *host*. If *host* is not specified, it defaults to the local host.

`-t` Make an RPC call to procedure 0 of *prognum* on the specified *host* using TCP, and report whether a response was received. This option is made obsolete by the `-T` option as shown in the third synopsis.

`-u` Make an RPC call to procedure 0 of *prognum* on the specified *host* using UDP, and report whether a response was received. This option is made obsolete by the `-T` option as shown in the third synopsis.

Examples

EXAMPLE 1 RPC services.

To show all of the RPC services registered on the local machine use:

```
example% rpcinfo
```

To show all of the RPC services registered with `rpcbind` on the machine named `klaxon` use:

```
example% rpcinfo klaxon
```

The information displayed by the above commands can be quite lengthy. Use the `-s` option to display a more concise list:

```
example% rpcinfo -s klaxon
```

program	vrsn	netid(s)	service	owner
100000	2,3,4	tcp,udp,ticlts,ticots,ticotsord	rpcbind	superuser
100008	1	ticotsord,ticots,ticlts,udp,tcp	walld	superuser
100002	2,1	ticotsord,ticots,ticlts,udp,tcp	rusersd	superuser
100001	2,3,4	ticotsord,ticots,tcp,ticlts,udp	rstatd	superuser
100012	1	ticotsord,ticots,ticlts,udp,tcp	sprayd	superuser
100007	3	ticotsord,ticots,ticlts,udp,tcp	ypbind	superuser
100029	1	ticotsord,ticots,ticlts	keyserv	superuser
100078	4	ticotsord,ticots,ticlts	-	superuser
100024	1	ticotsord,ticots,ticlts,udp,tcp	status	superuser
100021	2,1	ticotsord,ticots,ticlts,udp,tcp	nlockmgr	superuser
100020	1	ticotsord,ticots,ticlts,udp,tcp	llockmgr	superuser

To show whether the RPC service with program number *prognum* and version *versnum* is registered on the machine named `klaxon` for the transport TCP use:

```
example% rpcinfo -T tcp klaxon prognum versnum
```

To show all RPC services registered with version 2 of the `rpcbind` protocol on the local machine use:

```
example% rpcinfo -p
```

To delete the registration for version 1 of the `walld` (program number `100008`) service for all transports use:

```
example# rpcinfo -d 100008 1
```

or

```
example# rpcinfo -d walld 1
```

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also `rpcbind(1M)`, `rpc(3NSL)`, `netconfig(4)`, `rpc(4)`, `attributes(5)`

Name rpc.mdcommd – multi-node disk set services

Synopsis

```
/usr/sbin/rpc.mdcommd
```

Description `rpc.mdcommd` is an `rpc(4)` daemon that functions as a server process. `rpc.mdcommd` manages communication among hosts participating in a multi-node disk set configuration.

`rpc.mdcommd` is invoked by `inetd(1M)`.

Exit Status The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	storage/svm
Interface Stability	Committed

See Also `svcs(1)`, `inetadm(1M)`, `inetd(1M)`, `metaset(1M)`, `svcadm(1M)`, `rpc(3NSL)`, `rpc(4)`, `services(4)`, `attributes(5)`, `smf(5)`

Solaris Volume Manager Administration Guide

Notes The `rpc.mdcommd` service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/network/rpc/mdcomm
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. Responsibility for initiating and restarting this service is delegated to `inetd(1M)`. Use `inetadm(1M)` to make configuration changes and to view configuration information for this service. The service's status can be queried using the `svcs(1)` command.

Name rpc.metad – remote metaset services

Synopsis

```
/usr/sbin/rpc.metad
```

Description `rpc.metad` is an `rpc(4)` daemon (functioning as a server process) that is used to manage local copies of metadevice diskset information. The `rpc.metad` daemon is controlled by `inetadm(1M)`.

Exit Status The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	storage/svm

See Also `svcs(1)`, `inetadm(1M)`, `inetd(1M)`, `metaset(1M)`, `rpc.metamhd(1M)`, `svcadm(1M)`, `rpc(3NSL)`, `services(4)`, `attributes(5)`, `smf(5)`

Solaris Volume Manager Administration Guide

Notes The `rpc.metad` service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/network/rpc/meta:default
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. Responsibility for initiating and restarting this service is delegated to `inetd(1M)`. Use `inetadm(1M)` to make configuration changes and to view configuration information for this service. The service's status can be queried using the `svcs(1)` command.

Name rpc.metamedd – remote mediator services

Synopsis

```
/usr/sbin/rpc.metamedd
```

Description `rpc.metamedd` is an `rpc(4)` server which is used to manage mediator information for use in 2–string HA configurations. The `rpc.metamedd` daemon is controlled by `inetadm(1M)`.

Exit Status The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	storage/svm
Interface Stability	Committed

See Also `svcs(1)`, `inetadm(1M)`, `inetd(1M)`, `svcadm(1M)`, `rpc(4)`, `services(4)`, `attributes(5)`, `smf(5)`

Sun Cluster documentation, *Solaris Volume Manager Administration Guide*

Notes The `rpc.metamedd` service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/network/rpc/metamed:default
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. Responsibility for initiating and restarting this service is delegated to `inetd(1M)`. Use `inetadm(1M)` to make configuration changes and to view configuration information for this service. The service's status can be queried using the `svcs(1)` command.

Name rpc.metamhd – remote multihost disk services

Synopsis

```
/usr/sbin/rpc.metamhd
```

Description `rpc.metamhd` is an `rpc(4)` daemon (functioning as a server process) that is used to manage multi-hosted disks. The `rpc.metamhd` daemon is controlled by `inetadm(1M)`.

Exit Status The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	storage/svm

See Also `svcs(1)`, `inetadm(1M)`, `inetd(1M)`, `metaset(1M)`, `rpc.metad(1M)`, `svcadm(1M)`, `rpc(3NSL)`, `services(4)`, `attributes(5)`, `smf(5)`

Solaris Volume Manager Administration Guide

Notes The `rpc.metamhd` service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/network/rpc/metamh:default
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. Responsibility for initiating and restarting this service is delegated to `inetd(1M)`. Use `inetadm(1M)` to make configuration changes and to view configuration information for this service. The service's status can be queried using the `svcs(1)` command.

Name rpc.rexd, rexd – RPC-based remote execution server

Synopsis

```
/usr/sbin/rpc.rexd [-s]
```

Description `rpc.rexd` is the Oracle Sun RPC server for remote program execution. This daemon is started by `inetd(1M)` whenever a remote execution request is made.

For non-interactive programs, the standard file descriptors are connected directly to TCP connections. Interactive programs involve pseudo-terminals, in a fashion that is similar to the login sessions provided by `rlogin(1)`. This daemon may use NFS to mount file systems specified in the remote execution request.

There is a `10240` byte limit for arguments to be encoded and passed from the sending to the receiving system.

Options The following option is supported:

`-s` Secure. When specified, requests must have valid DES credentials. If the request does not have a DES credential it is rejected. The default publickey credential is rejected. Only newer `on(1)` commands send DES credentials.

If access is denied with an authentication error, you may have to set your publickey with the `chkey(1)` command.

Specifying the `-s` option without presenting secure credentials will result in an error message: `Unix too weak auth (DesONly)!`

Security `rpc.rexd` uses `pam(3PAM)` for account and session management. The PAM configuration policy, configured in `/etc/pam.conf` or per-service files in `/etc/pam.d/`, specifies the modules to be used for `rpc.rexd`. Here is a partial `pam.conf` file with `rpc.rexd` entries for account and session management using the UNIX module:

```
rpc.rexd   account requisite       pam_roles.so.1
rpc.rexd   account required        pam_projects.so.1
rpc.rexd   account required        pam_unix_account.so.1

rpc.rexd      session required      pam_unix_session.so.1
```

The equivalent PAM configuration in `/etc/pam.d/` would be the following entries in `/etc/pam.d/rpc.rexd`:

```
account requisite         pam_roles.so.1
account required                  pam_projects.so.1
account required                  pam_unix_account.so.1

session required                  pam_unix_session.so.1
```

If there are no entries for the `rpc.rexd` service in `/etc/pam.conf` and no `/etc/pam.d/rpc.rexd` file exists, then the entries for the "other" service in `/etc/pam.conf`

will be used. If there are not any entries in `/etc/pam.conf` for the "other" service, then the entries in `/etc/pam.d/other` will be used. `rpc.rexd` uses the `getpwuid()` call to determine whether the given user is a legal user.

Files

`/dev/pts`*n*	Pseudo-terminals used for interactive mode
`/etc/passwd`	Authorized users
`/tmp_rex/rexd??????`	Temporary mount points for remote file systems

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/network/nis

See Also `chkey(1)`, `on(1)`, `rlogin(1)`, `svcs(1)`, `inetadm(1M)`, `inetd(1M)`, `svcadm(1M)`, `pam(3PAM)`, `pam.conf(4)`, `publickey(4)`, `attributes(5)`, `pam_authtok_check(5)`, `pam_authtok_get(5)`, `pam_authtok_store(5)`, `pam_dhkeys(5)`, `pam_passwd_auth(5)`, `pam_unix_account(5)`, `pam_unix_auth(5)`, `pam_unix_session(5)`, `smf(5)`

Diagnostics Diagnostic messages are normally printed on the console, and returned to the requestor.

Notes Root cannot execute commands using `rexd` client programs such as `on(1)`.

The `rpc.rexd` service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/network/rpc/rex:default
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. Responsibility for initiating and restarting this service is delegated to `inetd(1M)`. Use `inetadm(1M)` to make configuration changes and to view configuration information for this service. The service's status can be queried using the `svcs(1)` command.

Name rpc.rstatd, rstatd – kernel statistics server

Synopsis

```
/usr/lib/netsvc/rstat/rpc.rstatd
```

Description `rpc.rstatd` is a server which returns performance statistics obtained from the kernel. `rup(1)` uses `rpc.rstatd` to collect the `uptime` information that it displays.

`rpc.rstatd` is an RPC service.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	service/network/network-servers

See Also `rup(1)`, `svcs(1)`, `inetadm(1M)`, `inetd(1M)`, `svcadm(1M)`, `services(4)`, `attributes(5)`, `smf(5)`

Notes The `rpc.rstatd` service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/network/rpc/rstat:default
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. Responsibility for initiating and restarting this service is delegated to `inetd(1M)`. Use `inetadm(1M)` to make configuration changes and to view configuration information for this service. The service's status can be queried using the `svcs(1)` command.

Name rpc.rusersd, rusersd – network username server

Synopsis

```
/usr/lib/netsvc/rusers/rpc.rusersd
```

Description `rpc.rusersd` is a server that returns a list of users on the host. The `rpc.rusersd` daemon may be started by `inetd(1M)`.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	service/network/network-servers

See Also `svcs(1)`, `inetadm(1M)`, `inetd(1M)`, `svcadm(1M)`, `attributes(5)`, `smf(5)`

Notes The `rpc.rusersd` service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/network/rpc/rusers:default
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. Responsibility for initiating and restarting this service is delegated to `inetd(1M)`. Use `inetadm(1M)` to make configuration changes and to view configuration information for this service. The service's status can be queried using the `svcs(1)` command.

Name rpc.rwalld, rwalld – network rwall server

Synopsis

```
/usr/lib/netsvc/rwall/rpc.rwalld
```

Description `rpc.rwalld` is a server that handles `rwall(1M)` requests. It is implemented by calling `wall(1M)` on all the appropriate network machines. The `rpc.rwalld` daemon may be started by `inetd(1M)`.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	service/network/network-servers

See Also `svcs(1)`, `inetadm(1M)`, `inetd(1M)`, `rwall(1M)`, `svcadm(1M)`, `wall(1M)`, `attributes(5)`, `smf(5)`

Notes The `rpc.rwalld` service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/network/rpc/wall:default
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. Responsibility for initiating and restarting this service is delegated to `inetd(1M)`. Use `inetadm(1M)` to make configuration changes and to view configuration information for this service. The service's status can be queried using the `svcs(1)` command.

Name rpc.smserverd – removable media device server

Synopsis

```
/usr/lib/smedia/rpc.smserverd
```

Description `rpc.smserverd` is a server that handles requests from client applications, such as volume management software, for access to removable media devices. In addition to volume management software, `rmformat(1)` and the CDE Filemanager (when performing removable media operations) are `rpc.smserverd` clients. The `rpc.smserverd` daemon is started by `inetd(1M)` when a client makes a call to a Solaris-internal library to access a SCSI, IDE, or USB device. The daemon is not started if a client attempts to access a PCMCIA device. Once started, the daemon remains active until such time as it is idle for three minutes or more.

The `rpc.smserverd` daemon is provided for the exclusive use of the client applications mentioned above. It has no external, customer-accessible interfaces, including no configuration file.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/storage/removable-media

See Also `svcs(1)`, `inetadm(1M)`, `inetd(1M)`, `svcadm(1M)`, `attributes(5)`, `smf(5)`

Notes The `rpc.smserverd` service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/network/rpc/smserver
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. Responsibility for initiating and restarting this service is delegated to `inetd(1M)`. Use `inetadm(1M)` to make configuration changes and to view configuration information for this service. The service's status can be queried using the `svcs(1)` command.

Name rpc.sprayd, sprayd – spray server

Synopsis

```
/usr/lib/netsvc/spray/rpc.sprayd
```

Description `rpc.sprayd` is a server that records the packets sent by `spray(1M)`. The `rpc.sprayd` daemon may be started by `inetd(1M)`.

The service provided by `rpc.sprayd` is not useful as a networking benchmark as it uses unreliable connectionless transports, (udp for example). It can report a large number of packets dropped when the drops were caused by the program sending packets faster than they can be buffered locally (before the packets get to the network medium).

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	service/network/network-servers

See Also `svcs(1)`, `inetadm(1M)`, `inetd(1M)`, `spray(1M)`, `svcadm(1M)`, `attributes(5)`, `smf(5)`

Notes The `rpc.sprayd` service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/network/rpc/spray:default
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. Responsibility for initiating and restarting this service is delegated to `inetd(1M)`. Use `inetadm(1M)` to make configuration changes and to view configuration information for this service. The service's status can be queried using the `svcs(1)` command.

Name rpc.yppasswdd, yppasswdd – server for modifying NIS password file

Synopsis

```
/usr/lib/netsvc/yp/rpc.yppasswdd [-D directory]
     [-nogecos] [-noshell] [-nopw]
     [-m argument1 argument2...]

/usr/lib/netsvc/yp/rpc.yppasswdd
     [passwordfile [adjunctfile]]
     [-nogecos] [-noshell] [-nopw]
     [-m argument1 argument2...]
```

Description `rpc.yppasswdd` is a server that handles password change requests from `yppasswd(1)`. It changes a password entry in the `passwd`, `shadow`, and `security/passwd.adjunct` files. The `passwd` and `shadow` files provide the basis for the `passwd.byname` and `passwd.byuid` maps. The `passwd.adjunct` file provides the basis for the `passwd.adjunct.byname` and `passwd.adjunct.byuid` maps. Entries in the `passwd`, `shadow` or `passwd.adjunct` files are changed only if the password presented by `yppasswd(1)` matches the encrypted password of the entry. All password files are located in the `PWDIR` directory.

If the `-D` option is given, the `passwd`, `shadow,` or `passwd.adjunct` files are placed under the directory path that is the argument to `-D`.

If the `-noshell`, `-nogecos` or `-nopw` options are given, these fields cannot be changed remotely using `chfn`, `chsh`, or `passwd(1)`.

If the `-m` option is given, a `make(1S)` is performed in `/var/yp` after any of the `passwd`, `shadow`, or `passwd.adjunct` files are modified. All arguments following the flag are passed to `make`.

The second of the listed syntaxes is provided only for backward compatibility. If the second syntax is used, the `passwordfile` is the full pathname of the password file and `adjunctfile` is the full pathname of the optional `passwd.adjunct` file. If a shadow file is found in the same directory as `passwordfile`, the `shadowfile` is used as described above. Use of this syntax and the discovery of a `shadowfile` file generates diagnostic output. The daemon, however, starts normally.

The first and second syntaxes are mutually exclusive. You cannot specify the full pathname of the `passwd`, `passwd.adjunct` files and use the `-D` option at the same time.

The daemon is started automatically on the master server of the passwd map by `ypstart(1M)`, which is invoked at boot time by the `svcs:/network/nis/server:default` service.

The server does not insist on the presence of a `shadow` file unless there is no `-D` option present or the directory named with the `-D` option is `/etc`. In addition, a `passwd.adjunct` file is not necessary. If the `-D` option is given, the server attempts to find a `passwd.adjunct` file in the `security` subdirectory of the named directory. For example, in the presence of `-D /var/yp` the server checks for a `/var/yp/security/passwd.adjunct` file.

If only a `passwd` file exists, then the encrypted password is expected in the second field. If both a `passwd` and a `passwd.adjunct` file exist, the encrypted password is expected in the second field of the adjunct file with *##username* in the second field of the `passwd` file. If all three files are in use, the encrypted password is expected in the `shadow` file. Any deviation causes a password update to fail.

If you remove or add a `shadow` or `passwd.adjunct` file after `rpc.yppasswdd` has started, you must stop and restart the daemon to enable it to recognize the change. See `ypstart(1M)` for information on restarting the daemon.

The `rpc.yppasswdd` daemon considers a shell that has a name that begins with 'r' to be a restricted shell. By default, the daemon does not check whether a shell begins with an 'r'. However, you can tell it to do so by uncommenting the `check_restricted_shell_name=1` line in `/etc/default/yppasswdd`. The result will be to restrict a user's ability to change from his default shell. See `yppasswdd(4)`.

On start up, `yppasswdd` checks for the existence of a NIS to LDAP (N2L) configuration file, `/var/yp/NISLDAPmapping`. If the configuration file is present, the daemon runs in N2L mode. If the file is not present, `yppasswdd` runs in traditional, non-N2L mode.

In N2L mode, changes are written directly to the Directory Information Tree (DIT). If the changes are written successfully, the NIS map is updated. The NIS source files, `passwd`, `shadow`, and `passwd.adjunct`, for example, are not updated. Thus, in N2L mode, the `-D` option is meaningless. In N2L mode, `yppasswdd` propagates changes by calling `yppush(1M)` instead of `ypmake(1M)`. The `-m` option is thus unused.

During an NIS-to-LDAP transition, the `yppasswdd` daemon uses the N2L-specific map, `ageing.byname`, to read and write password aging information to the DIT. If you are not using password aging, then the `ageing.byname` mapping is ignored.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	service/network/nis

See Also

`svcs(1)`, `make(1S)`, `passwd(1)`, `yppasswd(1)`, `inetd(1M)`, `svcadm(1M)`, `ypmake(1M)`, `yppush(1M)`, `ypstart(1M)`, `NISLDAPmapping(4)`, `passwd(4)`, `shadow(4)`, `ypfiles(4)`, `yppasswdd(4)`, `ypserv(4)`, `attributes(5)`, `smf(5)`

Notes

If `make` has not been installed and the `-m` option is given, the daemon outputs a warning and proceeds, effectively ignoring the `-m` flag.

When using the `-D` option, you should make sure that the `PWDIR` of the `/var/yp/Makefile` is set accordingly.

The second listed syntax is supplied only for backward compatibility and might be removed in a future release of this daemon.

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed. The name Yellow Pages is a registered trademark in the United Kingdom of British Telecommunications PLC, and cannot be used without permission.

The NIS server service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svcs:/network/nis/server:default
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. The service's status can be queried using the `svcs(1)` command.

Name rpc.ypupdated, ypupdated – server for changing NIS information

Synopsis

```
/usr/lib/netsvc/yp/rpc.ypupdated [-is]
```

Description `ypupdated` is a daemon that updates information in the Network Information Service (NIS). `ypupdated` consults the `updaters(4)` file in the `/var/yp` directory to determine which NIS maps should be updated and how to change them.

By default, the daemon requires the most secure method of authentication available to it, either DES (secure) or UNIX (insecure).

On start up, `ypupdated` checks for the existence of a NIS to LDAP (N2L) configuration file, `/var/yp/NISLDAPmapping`. If the file is present, `ypupdated` generates an informational message and exits. `ypupdated` is not supported in N2L mode.

Options

`-i` Accept RPC calls with the insecure `AUTH_UNIX` credentials. This allows programmatic updating of the NIS maps in all networks.

`-s` Accept only calls authenticated using the secure RPC mechanism (`AUTH_DES` authentication). This disables programmatic updating of the NIS maps unless the network supports these calls.

Files

`/var/yp/updaters` Configuration file for `rpc.updated` command.

`/var/yp/NISLDAPmapping` Configuration file for N2L

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	service/network/nis
Interface Stability	Committed

See Also `keyserv(1M)`, `updaters(4)`, `NISLDAPmapping(4)`, `attributes(5)`

Oracle Solaris Administration: Naming and Directory Services

Notes The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two services remains the same. Only the name has changed. The name Yellow Pages is a registered trademark in the United Kingdom of British Telecommunications PLC, and it must not be used without permission.

Name rquotad – remote quota server

Synopsis

```
/usr/lib/nfs/rquotad
```

Description `rquotad` is an `rpc(4)` server which returns quotas for a user of a local file system which is mounted by a remote machine over the NFS. The results are used by `quota(1M)` to display user quotas for remote file systems. The `rquotad` daemon is normally invoked by `inetd(1M)`.

Usage See `largefile(5)` for the description of the behavior of `rquotad` when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

Files

`quotas`	quota file at a UFS file system root

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	service/file-system/nfs

See Also `svcs(1)`, `automountd(1M)`, `inetadm(1M)`, `inetd(1M)`, `mount_nfs(1M)`, `quota(1M)`, `share_nfs(1M)`, `svcadm(1M)`, `rpc(4)`, `services(4)`, `attributes(5)`, `largefile(5)`, `smf(5)`

Installing Oracle Solaris 11.1 Systems

Notes The `rquotad` service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/network/nfs/rquota
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. Responsibility for initiating and restarting this service is delegated to `inetd(1M)`. Use `inetadm(1M)` to make configuration changes and to view configuration information for this service. The service's status can be queried using the `svcs(1)` command.

If it is disabled, it is enabled by `mount_nfs(1M)`, `share_nfs(1M)`, and `automountd(1M)` unless its `application/auto_enable` property is set to `false`.

Name rsh, restricted_shell – restricted shell command interpreter

Synopsis

```
/usr/lib/rsh [-acefhiknprstuvx] [argument]...
```

Description `rsh` is a limiting version of the standard command interpreter `sh`, used to restrict logins to execution environments whose capabilities are more controlled than those of `sh` (see `sh(1)` for complete description and usage).

When the shell is invoked, it scans the environment for the value of the environmental variable, `SHELL`. If it is found and `rsh` is the file name part of its value, the shell becomes a restricted shell.

The actions of `rsh` are identical to those of `sh`, except that the following are disallowed:

- changing directory (see `cd(1)`),
- setting the value of `$PATH`,
- pecifying path or command names containing `/`,
- redirecting output (> and >>).

The restrictions above are enforced after *.profile* is interpreted.

A restricted shell can be invoked in one of the following ways:

1. `rsh` is the file name part of the last entry in the `/etc/passwd` file (see `passwd(4)`);
2. the environment variable `SHELL` exists and `rsh` is the file name part of its value; the environment variable `SHELL` needs to be set in the `.login` file;
3. the shell is invoked and `rsh` is the file name part of argument 0;
4. the shell is invoke with the `-r` option.

When a command to be executed is found to be a shell procedure, `rsh` invokes `sh` to execute it. Thus, it is possible to provide to the end-user shell procedures that have access to the full power of the standard shell, while imposing a limited menu of commands; this scheme assumes that the end-user does not have write and execute permissions in the same directory.

The net effect of these rules is that the writer of the *.profile* (see `profile(4)`) has complete control over user actions by performing guaranteed setup actions and leaving the user in an appropriate directory (probably *not* the login directory).

The system administrator often sets up a directory of commands (that is, `/usr/rbin`) that can be safely invoked by a restricted shell. Some systems also provide a restricted editor, `red`.

Exit Status Errors detected by the shell, such as syntax errors, cause the shell to return a non-zero exit status. If the shell is being used non-interactively execution of the shell file is abandoned. Otherwise, the shell returns the exit status of the last command executed.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also Intro(1), cd(1), login(1), rsh(1), sh(1), exec(2), passwd(4), profile(4), attributes(5)

Notes The restricted shell, /usr/lib/rsh, should not be confused with the remote shell, /usr/bin/rsh, which is documented in rsh(1).

Name rtc – provide all real-time clock and UTC-lag management

Synopsis

```
/usr/sbin/rtc [-c] [-z zone-name]
```

Description On x86 systems, the `rtc` command reconciles the difference in the way that time is established between UNIX and MS-DOS systems. UNIX systems utilize Universal Coordinated Time (UTC), while MS-DOS systems utilize local time.

Without arguments, `rtc` displays the currently configured time zone string. The currently configured time zone string is based on what was last recorded by `rtc -z` *zone-name*.

The `rtc` command is not normally run from a shell prompt; it is generally invoked by the system. Commands such as `date(1)` and `rdate(1M)`, which are used to set the time on a system, invoke `/usr/sbin/rtc -c` to ensure that daylight savings time (DST) is corrected for properly.

Options

`-c` This option checks for DST and makes corrections if necessary. It is normally run once a day by a `cron` job.

If there is no RTC time zone or `/etc/rtc_config` file, this option will do nothing.

`-z` *zone-name* This option, which is normally run by the system at software installation time, is used to specify the time zone in which the RTC is to be maintained. It updates the configuration file `/etc/rtc_config` with the name of the specified zone and the current UTC lag for that zone. If there is an existing `rtc_config` file, this command will update it. If not, this command will create it.

Files

`/etc/rtc_config` The data file used to record the time zone and UTC lag. This file is completely managed by `/usr/sbin/rtc`, and it is read by the kernel.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	x86
Availability	system/core-os

See Also `date(1)`, `rdate(1M)`, `attributes(5)`

Name rtquery – query routing daemons for their routing tables

Synopsis

```
rtquery [-np1] [-w timeout] [-r addr] [-a secret] host...

rtquery [-t operation] host...
```

Description The `rtquery` command is used to query a RIP network routing daemon, `in.routed(1M)` or `GateD`, for its routing table by sending a request or poll command. The routing information in any routing response packets returned is displayed numerically and symbolically.

By default, `rtquery` uses the request command. When the -p option is specified, `rtquery` uses the poll command, an undocumented extension to the RIP protocol supported by `GateD`. When querying `GateD`, the poll command is preferred over the request command because the response is not subject to Split Horizon and/or Poisoned Reverse, and because some versions of `GateD` do not answer the request command. `in.routed` does not answer the poll command, but recognizes requests coming from `rtquery` and so answers completely.

The `rtquery` command is also used to turn tracing on or off in `in.routed`.

Options The following options are supported:

`-a passwd=`*XXX*

`-a md5_passwd=`*XXX*|*KeyID*
: Causes the query to be sent with the indicated cleartext or MD5 password.

`-n`
: Displays only the numeric network and host addresses instead of both numeric and symbolic names.

`-p`
: Uses the poll command to request full routing information from `GateD`. This is an undocumented extension RIP protocol supported only by `GateD`.

`-r` *addr*
: Asks about the route to destination *addr*.

`-t` *operation*
: Changes tracing, where *operation* is one of the actions listed below. Requests from processes not running with UID 0 or on distant networks are generally ignored by the daemon except for a message in the system log. `GateD` is likely to ignore these debugging requests.

  on=*tracefile*
  : Turns tracing on, directing tracing into the specified file. That file must have been specified when the daemon was started or have the name, `/var/log/in.routed.trace`.

  more
  : Increases the debugging level.

  off
  : Turns off tracing.

dump — Dumps the daemon's routing table to the current trace file.

`-w` *timeout* — Changes the delay for an answer from each host. By default, each host is given 15 seconds to respond.

`-1` — Queries using RIP version 1 instead of RIP version 2.

Exit Status The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/network/routing

See Also `in.routed(1M)`, `route(1M)`, `gateways(4)`, `attributes(5)`, `icmp(7P)`, `inet(7P)`, `udp(7P)`

Routing Information Protocol, RIPv1, RFC 1058

Routing Information Protocol, RIPv2, RFC 2453, STD 0056

Name runacct – run daily accounting

Synopsis `/usr/lib/acct/runacct [`*mmdd* `[`*state*`]]`

Description `runacct` is the main daily accounting shell procedure. It is normally initiated using `cron`. `runacct` processes connect, fee, disk, and process accounting files. It also prepares summary files for `prdaily` or billing purposes. `runacct` is distributed only to source code licensees.

`runacct` takes care not to damage active accounting files or summary files in the event of errors. It records its progress by writing descriptive diagnostic messages into `active`. When an error is detected, a message is written to `/dev/console`, mail (see `mail(1)`) is sent to `root` and `adm`, and `runacct` terminates. `runacct` uses a series of lock files to protect against re-invocation. The files `lock` and `lock1` are used to prevent simultaneous invocation, and `lastdate` is used to prevent more than one invocation per day.

`runacct` breaks its processing into separate, restartable *states* using `statefile` to remember the last *state* completed. It accomplishes this by writing the *state* name into `statefile`. `runacct` then looks in `statefile` to see what it has done and to determine what to process next. *states* are executed in the following order:

`SETUP`	Move active accounting files into working files.
`WTMPFIX`	Verify integrity of `wtmpx` file, correcting date changes if necessary.
`CONNECT`	Produce connect session records in `tacct.h` format.
`PROCESS`	Convert process accounting records into `tacct.h` format.
`MERGE`	Merge the connect and process accounting records.
`FEES`	Convert output of `chargefee` into `tacct.h` format, merge with connect, and process accounting records.
`DISK`	Merge disk accounting records with connect, process, and fee accounting records.
`MERGETACCT`	Merge the daily total accounting records in `daytacct` with the summary total accounting records in `/var/adm/acct/sum/tacct`.
`CMS`	Produce command summaries.
`USEREXIT`	Any installation dependent accounting programs can be included here.
`CLEANUP`	Clean up temporary files and exit. To restart `runacct` after a failure, first check the `active` file for diagnostics, then fix any corrupted data files, such as `pacct` or `wtmpx`. The `lock`, `lock1`, and `lastdate` files must be removed before `runacct` can be restarted. The argument *mmdd* is necessary if `runacct` is being restarted. *mmdd* specifies the month and day for which `runacct` will rerun the accounting. The entry point for processing is based on the contents of `statefile`; to override this, include the desired *state* on the command line to designate where processing should begin.

Examples

EXAMPLE 1 Starting `runacct`

The following example starts `runacct`:

```
example% nohup runacct 2> /var/adm/acct/nite/fd2log &
```

EXAMPLE 2 Restarting `runacct`

The following example restarts `runacct`:

```
example% nohup runacct 0601 2>> /var/adm/acct/nite/fd2log &
```

EXAMPLE 3 Restarting `runacct` at a Specific State

The following example restarts `runacct` at a specific state:

```
example% nohup runacct 0601 MERGE 2>> /var/adm/acct/nite/fd2log &
```

Files

`/var/adm/wtmpx`	History of user access and administration information
`/var/adm/pacct`*incr*	
`/var/adm/acct/nite/active`	
`/var/adm/acct/nite/daytacct`	
`/var/adm/acct/nite/lock`	
`/var/adm/acct/nite/lock1`	
`/var/adm/acct/nite/lastdate`	
`/var/adm/acct/nite/statefile`	

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/accounting/legacy-accounting

See Also

`acctcom(1)`, `mail(1)`, `acct(1M)`, `acctcms(1M)`, `acctcon(1M)`, `acctmerg(1M)`, `acctprc(1M)`, `acctsh(1M)`, `cron(1M)`, `fwtmp(1M)`, `acct(2)`, `acct.h(3HEAD)`, `utmpx(4)`, `attributes(5)`

Notes

It is not recommended to restart `runacct` in the SETUP *state*. Run SETUP manually and restart using:

`runacct` *mmdd* WTMPFIX

If `runacct` failed in the PROCESS *state,* remove the last `ptacct` file because it will not be complete.

The `runacct` command can process a maximum of

- 6000 distinct sessions

- 1000 distinct terminal lines
- 2000 distinct login names

during a single invocation of the command. If at some point the actual number of any one of these items exceeds the maximum, the command will not succeed.

Do not invoke `runacct` at the same time as `ckpacct`, as there may be a conflict if both scripts attempt to execute `turnacct switch` simultaneously.

Name rwall – write to all users over a network

Synopsis

```
/usr/sbin/rwall hostname...
/usr/sbin/rwall -n netgroup...
/usr/sbin/rwall -h hostname -n netgroup
```

Description `rwall` reads a message from standard input until EOF. It then sends this message, preceded by the line:

```
Broadcast Message . . .
```

to all users logged in on the specified host machines. With the `-n` option, it sends to the specified network groups.

Options

`-n` *netgroup* Send the broadcast message to the specified network groups.

`-h` *hostname* Specify the `hostname`, the name of the host machine.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	service/network/network-clients

See Also `inetd(1M)`, `wall(1M)`, `attributes(5)`

Notes The timeout is fairly short to allow transmission to a large group of machines (some of which may be down) in a reasonable amount of time. Thus the message may not get through to a heavily loaded machine.

Name sar, sa1, sa2, sadc – system activity report package

Synopsis

```
/usr/lib/sa/sadc [t n] [ofile]

/usr/lib/sa/sa1 [t n]

/usr/lib/sa/sa2 [-aAbcdgkmpqruvwy] [-e time] [-f filename]
     [-i sec] [-s time]
```

Description System activity data can be accessed at the special request of a user (see `sar(1)`) and automatically, on a routine basis, as described here. The operating system contains several counters that are incremented as various system actions occur. These include counters for CPU utilization, buffer usage, disk and tape I/O activity, TTY device activity, switching and system-call activity, file-access, queue activity, inter-process communications, and paging. For more general system statistics, use `iostat(1M)`, `sar(1)`, or `vmstat(1M)`.

`sadc` and two shell procedures, `sa1` and `sa2`, are used to sample, save, and process this data.

`sadc`, the data collector, samples system data *n* times, with an interval of *t* seconds between samples, and writes in binary format to *ofile* or to standard output. The sampling interval *t* should be greater than 5 seconds; otherwise, the activity of `sadc` itself may affect the sample. If *t* and *n* are omitted, a special record is written. This facility can be used at system boot time, when booting to a multi-user state, to mark the time at which the counters restart from zero. For example, when accounting is enabled, the `svc:/system/sar:default` service writes the restart mark to the daily data file using the command entry:

```
su sys -c "/usr/lib/sa/sadc /var/adm/sa/sa`date +%d`"
```

The shell script `sa1`, a variant of `sadc`, is used to collect and store data in the binary file `/var/adm/sa/sa`*dd,* where `dd` is the current day. The arguments *t* and *n* cause records to be written *n* times at an interval of *t* seconds, or once if omitted. The following entries in `/var/spool/cron/crontabs/sys` will produce records every 20 minutes during working hours and hourly otherwise:

```
0 * * * 0-6 /usr/lib/sa/sa1
20,40 8–17 * * 1–5 /usr/lib/sa/sa1
```

See `crontab(1)` for details.

The shell script `sa2`, a variant of `sar`, writes a daily report in the file `/var/adm/sa/sar`*dd.* See the `OPTIONS` section in `sar(1)` for an explanation of the various options. The following entry in `/var/spool/cron/crontabs/sys` will report important activities hourly during the working day:

```
5 18 * * 1–5 /usr/lib/sa/sa2 -s 8:00 -e 18:01 -i 1200 -A
```

Files

`/tmp/sa.adrfl`	address file
`/var/adm/sa/sadd`	daily data file
`/var/adm/sa/sardd`	daily report file

`/var/spool/cron/crontabs/sys` used for performance collection

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/accounting/legacy-accounting

See Also `crontab(1)`, `sar(1)`, `svcs(1)`, `timex(1)`, `iostat(1M)`, `svcadm(1M)`, `vmstat(1M)`, `attributes(5)`, `smf(5)`

Oracle Solaris Administration: Common Tasks

Notes The `sar` service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/system/sar
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. The service's status can be queried using the `svcs(1)` command.

Name sasinfo – Serial Attached SCSI HBA port command line interface

Synopsis

```
sasinfo hba [-v] [HBA_Name]...

sasinfo hba-port [-lvy] [-a HBA_Name] [HBA_port_nam]...

sasinfo expander [-vt] [-p HBA_port_Name] [Expander_SAS_Addr]...

sasinfo target-port [-s | -v] [Target_port_SAS_Addr]...

sasinfo logical-unit | lu [-v] [device_path]...

sasinfo [-V]

sasinfo [-?]
```

Description The `sasinfo` utility is a command line interface that collects administrative information on Serial Attached SCSI–2 (SAS–2) host bus adapter (HBA) that supports the Storage Management HBA API (SM-HBA). The utility reports attributes of HBA ports and of expander devices and SCSI target devices that might be connected to those HBA ports.

`sasinfo` is implemented as a set of subcommands, described below.

Sub-commands The following subcommands are supported by `sasinfo`.

`hba`
: Lists information for the HBA referenced by the specified *HBA_name*. If *HBA_name* is not specified, all HBAs on the host will be listed.

`hba-port`
: Lists information for the HBA port referenced by the specified *HBA_port_Name*. If *HBA_port_Name* is not specified, all HBA ports on the host will be listed. Note that HBA ports can be dynamically configured/unconfigured so that the number of HBA ports on an HBA can change as connections to target devices are established. With no connections to a storage device, no HBA port is configured.

`expander`
: Lists hierarchical view of the expander referenced by the specified *Expander_SAS_Addr*. If no argument is specified, all expanders that are visible across all HBA ports on the host will be listed.

`target-port`
: Lists information for those target ports referenced by *Target_port_SAS_Addr* address. If no argument is specified, all target ports that are visible on the host will be listed.

`logical-unit` | `lu`
: Lists the logical unit referenced by the specified *device_path*. If *device_path* is not specified, all SAS logical units will be listed, including the SAS Management Protocol (SMP) target port.

Options The following options are supported.

`-?`, `--help`
Displays usage information

`-a`, `--hba` *HBA_Name*
Retrieve HBA port information from the *HBA_Name* of an HBA on the host. The `-a` option can only be used with the `hba-port` subcommand.

`-l`, `--phy-linkstat`
Lists the link error statistics information for the `phys` on the HBA port referenced by the specified *HBA_port_Name* or all HBA ports if no HBA port is specified. This option is used only with the `hba-port` subcommand.

`-p` *HBA_port_SAS_Addr*, `--port` *HBA_port_SAS_Addr*
Retrieve remote port information from the *HBA_port_SAS_Addr* of the local HBA port on the host. The `-p` option can be used only with the `expander` subcommand.

`-s`, `--scsi`
Lists SCSI attributes for target ports that are requested for display. This option is only used for the `target-port` subcommand.

`-v`, `--verbose`
Display details on hardware information, such as SAS address, topology device, and so forth, based on which subcommand is invoked.

`-V`, `--version`
Displays the version information.

`-y`, `--phy`
Lists the `phy` information on the HBA port specified by *HBA_port_Name* or all HBA ports if no HBA port is specified. This option is used only with the `hba-port` subcommand.

Examples **EXAMPLE 1** Listing All HBAs

The following command lists all HBAs on the host.

```
# sasinfo hba
HBA Name: SUNW-pmcs-0
HBA Name: SUNW-pmcs-1
```

EXAMPLE 2 Listing All HBAs with Details

The following command lists all HBAs on the host, along with related details. Note that each HBA has two HBA ports configured, with each HBA port connected to a storage device.

```
# sasinfo hba -v
HBA Name: SUNW-pmcs-0
    Manufacturer: sun
    Model: SAS Gen-2
    Firmware Version: 1.1
```

EXAMPLE 2 Listing All HBAs with Details *(Continued)*

```
    FCode/BIOS Version: 1.1
    Serial Number: 111-11111
    Driver Name: smvsl
    Driver Version: 1.1
    Number of HBA Ports: 2
HBA Name: SUNW-pmcs-1
    Manufacturer: sun
    Model: SAS Gen-2
    Firmware Version: 1.1
    FCode/BIOS Version: 1.1
    Serial Number: 111-11111
    Driver Name: smvsl
    Driver Version: 1.1
    Number of HBA Ports: 2
```

EXAMPLE 3 Listing All HBA Ports

The following command lists all HBA ports on the host.

```
# sasinfo hba-port
HBA Name: SUNW-pmcs-0
    HBA Port Name: /dev/cfg/c1
    HBA Port Name: /dev/cfg/c2
HBA Name: SUNW-pmcs-1
    HBA Port Name: /dev/cfg/c3
    HBA Port Name: /dev/cfg/c4
```

EXAMPLE 4 Listing all HBA Ports with Details

The following command lists all HBA ports, with accompanying details.

```
# sasinfo hba-port -v
HBA Name: SUNW-pmcs-0
    HBA Port Name: /dev/cfg/c1
        Type: sas-device
        State: online
        Local SAS Address: 5000c50000d756aa
        Attached Port SAS Address: 50800201a5a502bf
        Number of Phys: 4
    HBA Port Name: /dev/cfg/c25000c50000d756cc
        Type: sas-device
        State: online
        Local SAS Address: 5000c50000d756aa
        Attached Port SAS Address: 50800201a5a503bf
        Number of Phys: 4
HBA Name: SUNW-pmcs-1
    HBA Port Name: /dev/cfg/c3
        Type: sas-device
```

EXAMPLE 4 Listing all HBA Ports with Details *(Continued)*

```
        State: online
        Local SAS Address: 5000c50000d756cc
        Attached Port SAS Address: 50800201a5a504bf
        Number of Phys: 4
    HBA Port Name: /dev/cfg/c4
        Type: sas-device
        State: online
        Local SAS Address: 5000c50000d756cc
        Attached Port SAS Address: 50800201a5a505bf
        Number of Phys: 4
```

EXAMPLE 5 Listing phy Information for All HBA Ports

The following command lists phy information for all HBA ports.

```
# sasinfo hba-port -y
HBA Name: SUNW-pmcs-0
    HBA Port Name: /dev/cfg/c1
        Phy Information:
         Identifier: 0
           State: enabled
           HardwareLinkRate(Min/Max): 1.5Gbit/3Gbit
           ProgrammedLinkRate(Min/Max): 1.5Gbit/3Gbit
           NegotiatedLinkRate: 3Gbit
         Identifier: 1
           State: enabled
           HardwareLinkRate(Min/Max): 1.5Gbit/3Gbit
           ProgrammedLinkRate(Min/Max): 1.5Gbit/3Gbit
           NegotiatedLinkRate: 3Gbit
         Identifier: 2
           State: enabled
           HardwareLinkRate(Min/Max): 1.5Gbit/3Gbit
           ProgrammedLinkRate(Min/Max): 1.5Gbit/3Gbit
           NegotiatedLinkRate: 3Gbit
         Identifier: 3
           State: enabled
           HardwareLinkRate(Min/Max): 1.5Gbit/3Gbit
           ProgrammedLinkRate(Min/Max): 1.5Gbit/3Gbit
           NegotiatedLinkRate: 3Gbit
    HBA Port Name: /dev/cfg/c2
        Phy Information:
         Identifier: 4
           State: enabled
           HardwareLinkRate(Min/Max): 1.5Gbit/3Gbit
           ProgrammedLinkRate(Min/Max): 1.5Gbit/3Gbit
           NegotiatedLinkRate: 3Gbit
         Identifier: 5
```

EXAMPLE 5 Listing phy Information for All HBA Ports *(Continued)*

```
            State: enabled
            HardwareLinkRate(Min/Max): 1.5Gbit/3Gbit
            ProgrammedLinkRate(Min/Max): 1.5Gbit/3Gbit
            NegotiatedLinkRate: 3Gbit
          Identifier: 6
            State: enabled
            HardwareLinkRate(Min/Max): 1.5Gbit/3Gbit
            ProgrammedLinkRate(Min/Max): 1.5Gbit/3Gbit
            NegotiatedLinkRate: 3Gbit
          Identifier: 7
            State: enabled
            HardwareLinkRate(Min/Max): 1.5Gbit/3Gbit
            ProgrammedLinkRate(Min/Max): 1.5Gbit/3Gbit
            NegotiatedLinkRate: 3Gbit
HBA Name: SUNW-pmcs-1
    HBA Port Name: /dev/cfg/c3
        Phy Information:
         Identifier: 0
           State: enabled
           HardwareLinkRate(Min/Max): 1.5Gbit/3Gbit
           ProgrammedLinkRate(Min/Max): 1.5Gbit/3Gbit
           NegotiatedLinkRate: 3Gbit
         Identifier: 1
           State: enabled
           HardwareLinkRate(Min/Max): 1.5Gbit/3Gbit
           ProgrammedLinkRate(Min/Max): 1.5Gbit/3Gbit
           NegotiatedLinkRate: 3Gbit
         Identifier: 2
           State: enabled
           HardwareLinkRate(Min/Max): 1.5Gbit/3Gbit
           ProgrammedLinkRate(Min/Max): 1.5Gbit/3Gbit
           NegotiatedLinkRate: 3Gbit
         Identifier: 3
           State: enabled
           HardwareLinkRate(Min/Max): 1.5Gbit/3Gbit
           ProgrammedLinkRate(Min/Max): 1.5Gbit/3Gbit
           NegotiatedLinkRate: 3Gbit
    HBA Port Name: /dev/cfg/c4
        Phy Information:
         Identifier: 4
           State: enabled
           HardwareLinkRate(Min/Max): 1.5Gbit/3Gbit
           ProgrammedLinkRate(Min/Max): 1.5Gbit/3Gbit
           NegotiatedLinkRate: 3Gbit
         Identifier: 5
           State: enabled
```

EXAMPLE 5 Listing phy Information for All HBA Ports *(Continued)*

```
          HardwareLinkRate(Min/Max): 1.5Gbit/3Gbit
          ProgrammedLinkRate(Min/Max): 1.5Gbit/3Gbit
          NegotiatedLinkRate: 3Gbit
        Identifier: 6
          State: enabled
          HardwareLinkRate(Min/Max): 1.5Gbit/3Gbit
          ProgrammedLinkRate(Min/Max): 1.5Gbit/3Gbit
          NegotiatedLinkRate: 3Gbit
        Identifier: 7
          State: enabled
          HardwareLinkRate(Min/Max): 1.5Gbit/3Gbit
          ProgrammedLinkRate(Min/Max): 1.5Gbit/3Gbit
          NegotiatedLinkRate: 3Gbit
```

EXAMPLE 6 Listing phy Link Error Statistics for a Specific HBA Port

The following command lists phy link error statistics for a particular port.

```
# sasinfo hba-port -ly /dev/cfg/c1
HBA Name: SUNW-pmcs-0
  HBA Port Name: /dev/cfg/c1
    Phy Information:
      Identifier: 0
       State: enabled
         HardwareLinkRate(Min/Max): 1.5Gbit/3Gbit
         ProgrammedLinkRate(Min/Max): 1.5Gbit/3Gbit
         NegotiatedLinkRate: 3Gbit
         Link Error Statistics:
           Invalid Dword: 0
           Running Disparity Error: 0
           Loss of Dword Sync: 0
           Reset Problem: 0
      Identifier: 1
         State: enabled
         HardwareLinkRate(Min/Max): 1.5Gbit/3Gbit
         ProgrammedLinkRate(Min/Max): 1.5Gbit/3Gbit
         NegotiatedLinkRate: 3Gbit
         Link Error Statistics:
           Invalid Dword: 0
           Running Disparity Error: 0
           Loss of Dword Sync: 0
           Reset Problem: 0
       Identifier: 2
          State: enabled
          HardwareLinkRate(Min/Max): 1.5Gbit/3Gbit
          ProgrammedLinkRate(Min/Max): 1.5Gbit/3Gbit
          NegotiatedLinkRate: 3Gbit
```

EXAMPLE 6 Listing phy Link Error Statistics for a Specific HBA Port *(Continued)*

```
          Link Error Statistics:
            Invalid Dword: 0
            Running Disparity Error: 0
            Loss of Dword Sync: 0
            Reset Problem: 0
       Identifier: 3
          State: enabled
          HardwareLinkRate(Min/Max): 1.5Gbit/3Gbit
          ProgrammedLinkRate(Min/Max): 1.5Gbit/3Gbit
          NegotiatedLinkRate: 3Gbit
          Link Error Statistics:
            Invalid Dword: 0
            Running Disparity Error: 0
            Loss of Dword Sync: 0
            Reset Problem: 0
```

EXAMPLE 7 Listing Expanders Connected to an HBA Port

The following command lists all expanders that are visible through the specified HBA port.

```
# sasinfo expander -p /dev/cfg/c1
HBA Name: SUNW-pmcs-0
  HBA Port Name: /dev/cfg/c1
    Expander SAS Address(Tier 1): 50800201a5a502bf
       Expander SAS Address(Tier 2): 50800201a5a50233
       Expander SAS Address(Tier 2); 5000c5000d2da812
    Expander SAS Address(Tier 1): 50800201a5a503bf
       Expander SAS Address(Tier 2): 50800201a5a502d2
       Expander SAS Address(Tier 2); 5000c5000d2da823
```

EXAMPLE 8 Listing Detailed Information on Expanders

The following command lists detailed information on all expanders that are visible through the specified HBA port.

```
# sasinfo expander -v -p /dev/cfg/c1
HBA Name: SUNW-pmcs-0
  HBA Port Name: /dev/cfg/c1
    Expander SAS Address(Tier 1): 50800201a5a502bf
      OS Device Name: /dev/smp/expd0
      State: online
        Expander SAS Address(Tier 2): 50800201a5a50233
          OS Device Name: /dev/smp/expd2
          State: online
        Expander SAS Address(Tier 2); 5000c5000d2da812
          OS Device Name: /dev/smp/expd3
          State: online
    Expander SAS Address(Tier 1): 50800201a5a503bf
```

EXAMPLE 8 Listing Detailed Information on Expanders *(Continued)*

```
      OS Device Name: /dev/smp/expd1
      State: online
        Expander SAS Address(Tier 2): 50800201a5a502d2
          OS Device Name: /dev/smp/expd3
          State: online
        Expander SAS Address(Tier 2); 5000c5000d2da823
          OS Device Name: /dev/smp/expd4
          State: online
```

EXAMPLE 9 Listing Target Ports Attached to Expanders

The following command lists all target ports that are attached to expanders connected to a specified HBA port.

```
# sasinfo expander -t -p /dev/cfg/c1
HBA Name: SUNW-pmcs-0
  HBA Port Name: /dev/cfg/c1
     Expander SAS Address(Tier 1): 50800201a5a502bf
       Target Port SAS Address: 50800201a5a504f1
          Expander SAS Address(Tier 2): 50800201a5a50233
            Target Port SAS Address: 50800201a5a502be
            Target Port SAS Address: 5000c5000d2da8b2
          Expander SAS Address(Tier 2); 5000c5000d2da812
            Target Port SAS Address: 50800201a5a502be
            Target Port SAS Address: 50800201a5a508b2
     Expander SAS Address(Tier 1): 50800201a5a503bf
       Target Port SAS Address: 50800201a5a50421
          Expander SAS Address(Tier 2): 50800201a5a502d2
            Target Port SAS Address: 50800201a5a503be
            Target Port SAS Address: 5000c5000d2da7be
          Expander SAS Address(Tier 2); 5000c5000d2da823
            Target Port SAS Address: 50800201a5a503be
            Target Port SAS Address: 5000c5000d2da7be
```

EXAMPLE 10 Listing Target Port Information

The following command lists all target ports discovered on the host.

```
# sasinfo target-port
Target Port SAS Address: 5000c5000bae4009
Target Port SAS Address: 5000c5000baef4b1
Target Port SAS Address: 5000c5000bae3fe1
Target Port SAS Address: 5000c5000bae49d9
Target Port SAS Address: 5000c5000bae36c5
```

EXAMPLE 11 Listing Target Port Information with Topology Details

The following command lists all target ports with the HBA port and expander that they are connected to.

```
# sasinfo target-port -v
Target Port SAS Address: 5000c5000bae4009
  Type: SAS Device
  HBA Port Name: /dev/cfg/c7
    Expander Device SAS Address: 500e004aaaaaaa3f
  HBA Port Name: /dev/cfg/c5
    Expander Device SAS Address: 500e004aaaaaaa3f
Target Port SAS Address: 5000c5000baef4b1
  Type: SAS Device
  HBA Port Name: /dev/cfg/c7
    Expander Device SAS Address: 500e004aaaaaaa3f
  HBA Port Name: /dev/cfg/c5
    Expander Device SAS Address: 500e004aaaaaaa3f
Target Port SAS Address: 5000c5000bae3fe1
  Type: SAS Device
  HBA Port Name: /dev/cfg/c7
    Expander Device SAS Address: 500e004aaaaaaa3f
  HBA Port Name: /dev/cfg/c5
    Expander Device SAS Address: 500e004aaaaaaa3f
Target Port SAS Address: 5000c5000bae49d9
  Type: SAS Device
  HBA Port Name: /dev/cfg/c7
    Expander Device SAS Address: 500e004aaaaaaa3f
  HBA Port Name: /dev/cfg/c5
    Expander Device SAS Address: 500e004aaaaaaa3f
Target Port SAS Address: 5000c5000bae36c5
  Type: SAS Device
  HBA Port Name: /dev/cfg/c7
    Expander Device SAS Address: 500e004aaaaaaa3f
  HBA Port Name: /dev/cfg/c5
    Expander Device SAS Address: 500e004aaaaaaa3f
```

EXAMPLE 12 Listing Target Ports with SCSI Information

The following command lists all target port details, including SCSI information for each target port.

```
# sasinfo target-port -s 5000c5000bae4009
Target Port SAS Address: 5000c5000bae4009
   Type: SAS Device
        HBA Port Name: /dev/cfg/c7
            Expander Device SAS Address: 500e004aaaaaaa3f
                LUN : 0
                  OS Device Name : /dev/rdsk/c6t5000C5000BAE400Bd0s2
                  Vendor:   Sun
```

EXAMPLE 12 Listing Target Ports with SCSI Information *(Continued)*

```
                Product:  J4400
                Device Type: Disk
        HBA Port Name: /dev/cfg/c5
            Expander Device SAS Address: 500e004aaaaaaa3f
                LUN : 0
                  OS Device Name : /dev/rdsk/c6t5000C5000BAE400Bd0s2
                  Vendor:   Sun
                  Product:  J4400
                  Device Type: Disk
```

EXAMPLE 13 Listing the Logical Units

The following command lists the logical units on a host.

```
# sasinfo logical-unit
OS Device Name: /dev/rdsk/c4t50020F2300B4904Ed0s2
OS Device Name: /dev/rdsk/c4t50020F230000B4AFd0s2
```

EXAMPLE 14 Listing Additional Information on Logical Units

The following command displays additional logical unit information using the `-v` option for device `/dev/rmt/0n`.

```
# sasinfo lu -v
OS Device Name: /dev/rdsk/c4t50020F2300B4904Ed0s2
   HBA Port Name: /dev/cfg/c1
     Target Port SAS Address: 50020f2300b4904e
        LUN: 0
   Vendor: Sun
   Product: J4400
   Device Type: Disk
OS Device Name: /dev/rdsk/c4t50020F230000B4AFd0s2
   HBA Port Name: /dev/cfg/c1
     Target Port SAS Address: 50020f230063100b
        LUN: 0
   Vendor: Sun
   Product: J4400
   Device Type: Disk
```

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/storage/sas-utilities
Interface Stability	Committed

See Also attributes(5)

Name savecore – save a crash dump of the operating system

Synopsis `/usr/bin/savecore [-Lvd] [-f` *dumpfile*`] [`*directory*`]`

Description The `savecore` utility saves a crash dump of the kernel (assuming that one was made) and writes a reboot message in the shutdown log. It is invoked by the `dumpadm` service each time the system boots.

`savecore` can be configured by `dumpadm(1M)` to save crash dump data in either a compressed or uncompressed format. For the compressed format, `savecore` saves the crash dump data in the file *directory*`/vmdump.`*N*, where *N* in the pathname is replaced by a number which increments by one each time `savecore` is run in *directory*. The compressed file can be uncompressed in a separate step using the `-f` *dumpfile* option. For the uncompressed format, `savecore` saves the crash dump data in the file *directory*`/vmcore.`*N* and the kernel's namelist in *directory*`/unix.`*N*.

Before writing out a crash dump, `savecore` reads a number from the file *directory*`/minfree`. This is the minimum number of kilobytes that must remain free on the file system containing *directory*. If after saving the crash dump the file system containing *directory* would have less free space the number of kilobytes specified in `minfree`, the crash dump is not saved. if the `minfree` file does not exist, `savecore` assumes a `minfree` value of 1 megabyte.

The `savecore` utility also logs a reboot message using facility `LOG_AUTH` (see `syslog(3C)`). If the system crashed as a result of a panic, `savecore` logs the panic string too.

Options The following options are supported:

`-d`
Disregard dump header valid flag. Force `savecore` to attempt to save a crash dump even if the header information stored on the dump device indicates the dump has already been saved.

`-f` *dumpfile*
Save a crash dump from the specified file instead of from the system's current dump device. When given *directory*`/vmdump.`*N*, uncompress the file to `vmcore.`*N* and `unix.`*N*, where *N* is the same number as in the compressed name.

This option may also be useful if the information stored on the dump device has been copied to an on-disk file by means of the `dd(1M)` command.

`-L`
Save a crash dump of the live running Solaris system, without actually rebooting or altering the system in any way. This option forces `savecore` to save a live snapshot of the system to the dump device, and then immediately to retrieve the data and to write it out to a new set of crash dump files in the specified directory. Live system crash dumps can only be performed if you have configured your system to have a dedicated dump device using `dumpadm(1M)`.

`savecore -L` does not suspend the system, so the contents of memory continue to change while the dump is saved. This means that live crash dumps are not fully self-consistent.

`-v`
Verbose. Enables verbose error messages from `savecore`.

Operands The following operands are supported:

directory
Save the crash dump files to the specified directory. If *directory* is not specified, `savecore` saves the crash dump files to the default `savecore` *directory*, configured by `dumpadm(1M)`.

Files
- *directory*`/vmcore.`*N*
- *directory*`/unix.`*N*
- *directory*`/bounds`
- *directory*`/minfree`
- `/var/crash/`‘`uname -n`‘ (default crash dump directory)

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also `adb(1)`, `mdb(1)`, `svcs(1)`, `dd(1M)`, `dumpadm(1M)`, `svcadm(1M)`, `syslog(3C)`, `attributes(5)`, `smf(5)`

Notes The system crash dump service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/system/dumpadm:default
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. The service's status can be queried using the `svcs(1)` command.

If the dump device is also being used as a swap device, you must run `savecore` very soon after booting, before the swap space containing the crash dump is overwritten by programs currently running.

When `savecore` creates a file it appends the suffix `.partial`. After the file is completed, it is renamed without the suffix. If files are found in the dump directory with this suffix, it means that either `savecore` is still busy, or that it was interrupted before completely writing the file. In the former case, use `ps(1)` to find the PID of the running `savecore` process and wait for it to complete. In the latter case, remove the partial file and recreate it by running `savecore -d`.

Name sbdadm – SCSI Block Disk command line interface

Synopsis

```
sbdadm create-lu [-s, --size size] filename

sbdadm delete-lu filename

sbdadm import-lu filename

sbdadm list-lu

sbdadm modify-lu [-s, --size size] lu_name | filename
```

Description The `sbdadm` command creates and manages SCSI-block-device-based logical units that are registered with the SCSI Target Mode Framework (STMF).

Sub-commands The `sbdadm` command supports the subcommands listed below. Note that if you enter a question mark as an argument to the command (`sbdadm ?`), `sbdadm` responds with a help display.

`create-lu` [`-s`, `--size` *size*] *filename*
Create a logical unit that can be registered with the STMF. For the `-s` option, *size* is an integer followed by one of the following letters, to indicate a unit of size:

`k`	kilobyte
`m`	megabyte
`g`	gigabyte
`t`	terabyte
`p`	petabyte
`e`	exabyte

If you do not specify *size*, the size defaults to the size of *filename*.

The size specified can exceed the size of the file or device.

`delete-lu` *filename*
Deletes an existing logical unit that was created using `sbdadm create-lu`. This effectively unloads the logical unit from the STMF framework. Any existing data on the logical unit remains intact.

`import-lu` *filename*
Imports and loads a logical unit into the STMF that was previously created using `sbdadm create-lu` and was since deleted from the STMF using `sbdadm delete-lu`. On success, the logical unit is again made available to the STMF. *filename* is the filename used in the `sbdadm create-lu` command for this logical unit.

`list-lu`
List all logical units that were created using the `sbdadm create-lu` command.

`modify-lu` [`-s`, `--size` *size*] *lu_name* | *filename*
Modifies attributes of an logical unit created using the `sbdadm create-lu` command. For the `-s` option, *size* is an integer value followed by a unit specifier. The unit specifiers are as described above under `create-lu`. When this option is specified, the existing size of the logical unit is changed to the new size.

The size specified can exceed the size of the file or device represented by *lu_name*.

Operands

`sbdadm` use the following operands:

filename
Name of an existing file or a fully qualified path to a raw block device.

lu_name
The 32-byte hexadecimal representation of the logical unit.

Examples

EXAMPLE 1 Creating a Logical Unit

The following series of commands creates a 10–gigabyte logical unit.

```
# touch /export/lun/0
# sbdadm create-lu -s 10g /export/lun/0
# sbdadm create-lu /dev/rdsk/c1t1d0s0
```

EXAMPLE 2 Listing Logical Units

The following command lists all logical units.

```
# sbdadm list-lu
Found 2 LU(s)

                GUID                  DATA SIZE    SOURCE
--------------------------------  -----------  ------
6000ae4000144f21d92c47b0dd650002  10737352704  /export/lun/0
6000ae4000144f21d92c47b0de300032  134283264    /dev/rdsk/c1t1d0s0
```

Exit Status

`0`
Successful completion.

`non-zero`
An error occurred.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/storage/scsi-target-mode-framework
Interface Stability	Obsolete Committed

See Also stmfadm(1M), attributes(5)

Name sckmd – Sun cryptographic key management daemon

Synopsis

```
/usr/platform/sun4u/lib/sckmd
```

Description `sckmd` is a server process that resides on a high-end system domain to maintain the Internet Protocol Security (IPsec) Security Associations (SAs) needed to secure communications between a Service Processor or System Controller (SC) and platform management software running within a domain. The `dcs`(1M) daemon uses these Security Associations. See `ipsec`(7P) for a description of Security Associations.

The `sckmd` daemon receives SAs from the Service Processor or SC and installs these SAs in a domain's Security Association Database (SADB) using `pf_key`(7P).

`sckmd` starts up at system boot time as an SMF service. The FMRI for the `sckmd` service is:

```
svc:/platform/sun4u/sckmd:default
```

A domain supports only one running `sckmd` process at a time.

Attributes See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsckmx.u, service/key-management/sun-fire-15000
Interface Stability	Committed

See Also `dcs`(1M), `ipsecconf`(1M), `ipsecalgs`(1M), `attributes`(5), `ipsec`(7P), `ipsecah`(7P), `ipsecesp`(7P), `pf_key`(7P)

Notes The `sckmd` service is used only on Sun Fire high-end systems and the SPARC Enterprise Server family. It provides a mechanism for exchanging IPsec keys between a domain and its System Controller (SC) or Service Processor. These platforms use IPsec to secure the communications between the SC or Service Processor and certain platform-specific daemons in the domain. `dcs`(1M) is such a daemon.

The documentation for each platform that supports `sckmd` describes how to configure its use of `IPsec` for such communications. Also, the documentation for each specific application describes how to configure its security policies and `IPsec` options in a manner appropriate for the target platform. Refer to the platform- and application-specific documentation for detailed information.

Name scmadm – storage cache manager administration utility

Synopsis

```
scmadm

scmadm -h

scmadm -e

scmadm -d

scmadm -v

scmadm -C [parameter [= [value]]...]

scmadm -o {system | cd | device} [option]

scmadm -m {cd |  diskname |  all}
```

Description The `scmadm` command provides various options for controlling and gathering information about a storage device cache.

Options If no options are specified, `scmadm` displays a list of configured cache descriptors with disknames, options, and global options. The `scmadm` command supports the following options:

`-h`
Displays usage information for the `scmadm` command.

`-e`
Reads the configuration and enables the storage device cache with those parameters. See `dscfg(1M)`.

`-d`
Shuts down the storage device cache.

`-v`
Displays the cache version number.

`-C` [*parameter*[=[*value*]] ...]
Sets or displays the configuration parameters. If the `-C` option is specified with no arguments the current cache configuration parameters are displayed. If *parameter* is specified, the current value of *parameter* is displayed. If *parameter=value* is specified, the current value of *parameter* is displayed and the parameter is changed to *value*. If *value* is omitted, or if *value* is specified as the null string, " ", or as "-", the parameter is deleted from the configuration and the system uses the default value. Multiple parameters can be specified in a single invocation of the `scmadm` command. A change in a configuration parameter takes effect only when the cache is next restarted.

`-o` { `system` | `cd` | *diskname* } [*option*]
Sets or displays the options for the system or for the cache device specified by `cd` or *diskname*. If the *option* `rdcache` or `nordcache` is specified, the system or specified cache device is set to that option. The option is saved as part of the configuration so that the option persists. See `dscfg(1M)`. To notify the system to "forget" about a saved option, use

the `forget` option. This does not change the option; it just removes the option from the saved configuration. If no option is specified, current options are displayed. The `rdcache` option is set as the default. The *option*s are defined as follows:

`rdcache`
Data blocks are likely to be referenced again and should remain in cache.

`nordcache`
Data blocks are unlikely to be referenced again and should be treated as least recently used, so that other blocks can remain in the cache longer.

`-m` { `cd` | *diskname* | `all` }
Displays the cache descriptor and diskname map for the device specified by `cd` or *diskname* or, if you specify `all`, displays the cache descriptors and diskname map for all storage devices on the system.

Exit Status The `scmadm` command returns 0 for success, non-zero for error.

Files `/dev/sdbc`

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	storage/avs/avs-cache-management
Interface Stability	Committed

See Also `dscfg(1M)`, `attributes(5)`

Diagnostics `scmadm` fails if there is insufficient contiguous memory.

Name sconadm – register system information

Synopsis

```
/usr/sbin/sconadm register -a
     [-e softwareUpdate | -E softwareUpdate]
     [-h hostname] [-l logfile] [-N]
     [-p proxy_host[:proxy_port]]
     [-r registration_profile] [-u username]
     [-x proxy_username]

/usr/sbin/sconadm proxy [-l logfile]
     [-p proxy_host[:proxy_port]]
     [-r registration_profile] [-x proxy_username]
```

Description The `sconadm` utility is a command-line version of the Basic Registration GUI. In the first form of the command in the SYNOPSIS, `sconadm` uses the `register` subcommand to register a host with a registration server. In the second form, `sconadm` uses the `proxy` subcommand to configure all of the components for software update to use an HTTP web proxy.

The parameters specified with `-u`, `-e` (or `-E`), `-h`, `-p`, and `-x` override values specified in your registration profile. A template for this profile, owned by root, with read-only permissions, is stored in `/usr/lib/breg/data/RegistrationProfile.properties`. See `registration_profile(4)`.

For the `proxy` subcommand, the proxy password is stored in the `RegistrationProfile.properties` file, available if proxy authentication is needed. Storage in the profile prevents proxy passwords from being exposed as part of a listing of processes on a system.

Options The following options are supported:

`-a`	Accept “Terms of Use and Binary Code License”. Absence of this option means that you do not accept the license.
`-e` softwareUpdate	Enable client to be managed at the Sun-hosted Update Connection Service.
`-E` softwareUpdate	Disable client's ability to be managed at the Sun-hosted Update Connection Service.
`-h` *hostname*	Hostname of the machine you want to register.
`-l` *logfile*	Pathname of log file.
`-N`	Never register.
`-p` *proxy_host*[:*proxy_port*]	Proxy host name and optional proxy port number.
`-r` *registration_profile*	Pathname to a registration profile.
`-u` *username*	User name (a Sun Online Account).
`-x` *proxy_username*	User name on the proxy host.

Examples

Unless specified otherwise, the commands below require root privileges or privileges equivalent to root. See `privileges(5)`.

EXAMPLE 1 Registering a New System

Assume a file `registrationprofile.properties` in `/tmp` that contains the following:

```
userName=user123
password=abc123
hostName=
subscriptionKey=
portalEnabled=false
proxyHostName=
proxyPort=
proxyUserName=
proxyPassword=
```

To register a new system using the profile above, you enter:

```
/usr/sbin/sconadm register -a -r /tmp/registrationprofile.properties
```

EXAMPLE 2 Reregistering a System with a Different User

Assume a file `registrationprofile.properties` in `/tmp` with the contents shown below. Note the changed specification for `userName` and `password`.

```
userName=newuser
password=newpassword
hostName=
subscriptionKey=
portalEnabled=false
proxyHostName=
proxyPort=
proxyUserName=
proxyPassword=
```

To reregister a new system using the profile above, you enter the same command you entered to register the system:

```
/usr/sbin/sconadm register -a -r /tmp/registrationprofile.properties
```

EXAMPLE 3 Reregistering a System, Adding a Sun Subscription Key

Modify `registrationprofile.properties` as follows:

```
userName=newuser
password=newpassword
hostName=
subscriptionKey=abc12345678
portalEnabled=false
proxyHostName=
```

EXAMPLE 3 Reregistering a System, Adding a Sun Subscription Key *(Continued)*

```
proxyPort=
proxyUserName=
proxyPassword=
```

Run the command:

```
/usr/sbin/sconadm register -a -r /tmp/registrationprofile.properties
```

EXAMPLE 4 Reregistering and Enabling Access to all Update Connection Services

Modify `registrationprofile.properties` as follows:

```
userName=newuser
password=newpassword
hostName=
subscriptionKey=abc12345678
portalEnabled=false
proxyHostName=
proxyPort=
proxyUserName=
proxyPassword=
```

Note that `portalEnabled` is set to `false`. Run the command:

```
/usr/sbin/sconadm register -a -r /tmp/registrationprofile.properties \
-e softwareUpdate
```

EXAMPLE 5 Never Registering

To never register a system, enter:

```
/usr/sbin/sconadm register -N
```

EXAMPLE 6 Using a Proxy Server With Proxy Authentication

Edit `registrationprofile.properties` as follows:

```
userName=
password=
hostName=
subscriptionKey=
portalEnabled=
proxyHostName=webcache.mycompany.com
proxyPort=8080
proxyUserName=myCompanyProxyUserName
proxyPassword=myCompanyProxyPassword
```

Run the command:

EXAMPLE 6 Using a Proxy Server With Proxy Authentication *(Continued)*

```
/usr/sbin/sconadm proxy -r /tmp/registrationprofile.properties
```

EXAMPLE 7 Changing Proxy Host Settings

Edit `registrationprofile.properties` as follows:

```
userName=
password=
hostName=
subscriptionKey=
portalEnabled=
proxyHostName=webcache.mycompany.com
proxyPort=8080
proxyUserName=myCompanyProxyUserName
proxyPassword=myCompanyProxyPassword
```

Run the command:

```
/usr/sbin/sconadm proxy -r /tmp/registrationprofile.properties
```

Then, change the `proxyHostName` value by running the following command:

```
/usr/sbin/sconadm proxy -r /tmp/registrationprofile.properties \
-p newproxy.mycompany.com
```

After the preceding command all proxies use `newproxy.mycompany.com`.

EXAMPLE 8 Resetting a System Not to Use a Proxy

Edit `registrationprofile.properties` as follows:

```
userName=
password=
hostName=
subscriptionKey=
portalEnabled=
proxyHostName=
proxyPort=
proxyUserName=
proxyPassword=
```

Note that values for all proxy fields are null.

Run the command:

```
/usr/sbin/sconadm proxy -r /tmp/registrationprofile.properties
```

Exit Status

`0` Success.

`>0` An error occurred.

Files

`/usr/lib/breg/data/RegistrationProfile.properties`
Registration profile template.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWbrg
Interface Stability	Committed

See Also

`registration_profile(4)`, `attributes(5)`, `privileges(5)`

Name sendmail – send mail over the internet

Synopsis

```
/usr/lib/sendmail [-Ac] [-Am] [-ba] [-bD] [-bd] [-bi] [-bl]
     [-bm] [-bp] [-bP] [-bs] [-bt] [-bv] [-B type] [-C file]
     [-D logfile] [-d X] [-F fullname] [-f name] [-G]
     [-h N] [-L tag] [-M xvalue] [-N notifications] [-n]
     [-Ooption =value] [-o xvalue] [-p protocol]
     [-Q [reason]] [-q [time]] [-q Xstring] [-R ret]
     [-r name] [-t] [-V envid] [-v] [-X logfile]
     [address]...
```

Description The `sendmail` utility sends a message to one or more people, routing the message over whatever networks are necessary. `sendmail` does internetwork forwarding as necessary to deliver the message to the correct place.

`sendmail` is not intended as a user interface routine. Other programs provide user-friendly front ends. `sendmail` is used only to deliver pre-formatted messages.

With no flags, `sendmail` reads its standard input up to an EOF, or a line with a single dot, and sends a copy of the letter found there to all of the addresses listed. It determines the network to use based on the syntax and contents of the addresses.

Local addresses are looked up in the local `aliases(4)` file, or in a name service as defined by the `nsswitch.conf(4)` file, and aliased appropriately. In addition, if there is a `.forward` file in a recipient's home directory, `sendmail` forwards a copy of each message to the list of recipients that file contains. Refer to the `NOTES` section for more information about `.forward` files. Aliasing can be prevented by preceding the address with a backslash.

There are several conditions under which the expected behavior is for the alias database to be either built or rebuilt. This cannot occur under any circumstances unless root owns *and* has exclusive write permission to the `/etc/mail/aliases*` files.

If a message is found to be undeliverable, it is returned to the sender with diagnostics that indicate the location and nature of the failure; or, the message is placed in a `dead.letter` file in the sender's home directory.

Service Management The `sendmail` service is managed by the service management facility, `smf(5)`, under the service identifiers:

```
svc:/network/smtp:sendmail
svc:/network/sendmail-client:default
```

Administrative actions on these services, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. The services' status can be queried using the `svcs(1)` command.

These are separate services rather than instances of the same service so that other services can properly express any dependencies. In particular, here are some guidelines about which service/instance should be depended on for which purposes:

- For a service that uses `sendmail` to send mail, an optional dependency on the service `svc:/network/sendmail-client` might be in order.
- For a service that needs to receive mail in general, but does not depend on `sendmail` being the particular SMTP receiver, a dependency on the service `svc:/network/smtp` might be in order.
- For a service that needs to interact with sendmail in particular, such as a `Milter`, a dependency on the instance `svc:/network/smtp:sendmail` might be in order.

For the last two, note the difference, as the latter has the "`:sendmail`" instance specification, whereas the former does not, thus representing the more general service.

Enabling Access to Remote Clients

On an unmodified system, access to `sendmail` by remote clients is enabled and disabled through the service management facility (see `smf(5)`). In particular, remote access is determined by the value of the `local_only` SMF property:

```
svc:/network/smtp:sendmail/config/local_only = true
```

A setting of `true`, as above, disallows remote access; `false` allows remote access. The default value is `true`.

The following example shows the sequence of SMF commands used to enable `sendmail` to allow access to remote systems:

```
# svccfg -s svc:/network/smtp:sendmail setprop config/local_only = false
# svcadm refresh svc:/network/smtp:sendmail
# svcadm restart svc:/network/smtp:sendmail
```

See `svcadm(1M)` and `svccfg(1M)`.

Note, however, on a system where any of the `sendmail(4)` files have been customized, setting this property might not have the intended effect. See `sendmail(4)` for details.

Automated Rebuilding of Configuration Files

See `sendmail(4)` for details on which service properties can be set to automate (re)building of configuration files when the service is started.

Restricting Host Access

`sendmail` uses TCP Wrappers to restrict access to hosts. It uses the service name of `sendmail` for `hosts_access()`. For more information on TCP Wrappers, see `tcpd`(1M) and `hosts_access`(4) in the `security/tcp-wrapper` package. `tcpd`(1M) and `hosts_access`(4) are not part of the Solaris man pages.

Startup Options

The `/etc/default/sendmail` file stores startup options for `sendmail` so that the options are not removed when a host is upgraded. See also `sendmail(4)` for details on which service properties can be set to automate (re)building of configuration files when the service is started.

You can use the following variables in the `/etc/default/sendmail` startup file:

CLIENTOPTIONS=*string*
Selects additional options to be used with the client daemon, which looks in the client-only queue (`/var/spool/clientmqueue`) and acts as a client queue runner. No syntax checking is done, so be careful when making changes to this variable.

CLIENTQUEUEINTERVAL=#
Similar to the `QUEUEINTERVAL` option, `CLIENTQUEUEINTERVAL` sets the time interval for mail queue runs. However, the `CLIENTQUEUEINTERVAL` option controls the functions of the client daemon, instead of the functions of the master daemon. Typically, the master daemon is able to deliver all messages to the SMTP port. However, if the message load is too high or the master daemon is not running, then messages go into the client-only queue, `/var/spool/clientmqueue`. The client daemon, which checks in the client-only queue, then acts as a client queue processor.

ETRN_HOSTS=*string*
Enables an SMTP client and server to interact immediately without waiting for the queue run intervals, which are periodic. The server can immediately deliver the portion of its queue that goes to the specified hosts. For more information, refer to the `etrn`(1M) man page.

MODE=-bd
Selects the mode to start `sendmail` with. Use the `-bd` option or leave it undefined.

OPTIONS=*string*
Selects additional options to be used with the master daemon. No syntax checking is done, so be careful when making changes to this variable.

QUEUEINTERVAL=#
Sets the interval for mail queue runs on the master daemon. # can be a positive integer that is followed by either `s` for seconds, `m` for minutes, `h` for hours, `d` for days, or `w` for weeks. The syntax is checked before `sendmail` is started. If the interval is negative or if the entry does not end with an appropriate letter, the interval is ignored and `sendmail` starts with a queue interval of 15 minutes.

QUEUEOPTIONS=p
Enables one persistent queue runner that sleeps between queue run intervals, instead of a new queue runner for each queue run interval. You can set this option to `p`, which is the only setting available. Otherwise, this option is not set.

Mail Filter API

`sendmail` supports a mail filter API called “milter”. For more information, see `/usr/include/libmilter/README` and http://www.milter.org

Options

The following options are supported:

`-Ac`
Uses `submit.cf` even if the operation mode does not indicate an initial mail submission.

`-Am`
Uses `sendmail.cf` even if the operation mode indicates an initial mail submission.

`-ba`
Goes into `ARPANET` mode. All input lines must end with a RETURN-LINEFEED, and all messages are generated with a RETURN-LINEFEED at the end. Also, the `From:` and `Sender:` fields are examined for the name of the sender.

`-bd`
Runs as a daemon in the background, waiting for incoming SMTP connections.

`-bD`
Runs as a daemon in the foreground, waiting for incoming SMTP connections.

`-bi`
Initializes the `aliases(4)` database. Root must own *and* have exclusive write permission to the `/etc/mail/aliases*` files for successful use of this option.

`-bl`
Runs as a daemon (like `-bd`) but accepts only loopback SMTP connections.

`-bm`
Delivers mail in the usual way (default).

`-bp`
Prints a summary of the mail queues.

`-bP`
Prints the number of entries in the queues. This option is only available with shared memory support.

`-bs`
Uses the SMTP protocol as described in RFC 2821. This flag implies all the operations of the `-ba` flag that are compatible with SMTP.

`-bt`
Runs in address test mode. This mode reads addresses and shows the steps in parsing; it is used for debugging configuration tables.

`-bv`
Verifies names only. Does not try to collect or deliver a message. Verify mode is normally used for validating users or mailing lists.

`-B` *type*
Indicates body *type* (`7BIT` or `8BITMIME`).

`-C` *file*
Uses alternate configuration file.

`-D` *logfile*
Send debugging output to the indicated log file instead of `stdout`.

`-d` *X*
Sets debugging value to *X*.

`-f` *name*
Sets the name of the "from" person (that is, the sender of the mail).

`-F` *fullname*
Sets the full name of the sender.

`-G`
When accepting messages by way of the command line, indicates that they are for relay (gateway) submission. When this flag is set, `sendmail` might complain about syntactically invalid messages, for example, unqualified host names, rather than fixing them. `sendmail` does not do any canonicalization in this mode.

`-h` *N*
Sets the hop count to *N*. The hop count is incremented every time the mail is processed. When it reaches a limit, the mail is returned with an error message, the victim of an aliasing loop.

`-L` *tag*
Sets the identifier used in `syslog` messages to the supplied *tag*.

`-M`*x*`=`*value*
Sets macro *x* to the specified *value*.

`-n`
Does not do aliasing.

`-N` *notifications*
Tags all addresses being sent as wanting the indicated *notifications*, which consists of the word "NEVER" or a comma-separated list of "SUCCESS", "FAILURE", and "DELAY" for successful delivery, failure and a message that is stuck in a queue somwhere. The default is "FAILURE,DELAY".

`-o`*x*`=`*value*
Sets option *x* to the specified *value*. `Processing Options` are described below.

`-O`*option*`=`*value*
Sets *option* to the specified *value* (for long from names). `Processing Options` are described below.

`-p` *protocol*
Sets the sending protocol. The *protocol* field can be in form *protocol*`:`*host* to set both the sending protocol and the sending host. For example: `-pUUCP:uunet` sets the sending *protocol* to `UUCP` and the sending host to `uunet`. Some existing programs use `-oM` to set the *r* and *s* macros; this is equivalent to using `-p`.

`-q`[*time*]
Processes saved messages in the queue at given intervals. If *time* is omitted, processes the queue once. *time* is given as a tagged number, where *s* is seconds, *m* is minutes, *h* is hours, *d* is days, and *w* is weeks. For example, `-q1h30m` or `-q90m` would both set the timeout to one hour thirty minutes.

By default, sendmail runs in the background. This option can be used safely with `-bd`.

`-qp[`*time*`-]`
Similar to `-q[`*time*`]`, except that instead of periodically forking a child to process the queue, `sendmail` forks a single persistent child for each queue that alternates between processing the queue and sleeping. The sleep time (*time*) is specified as the argument; it defaults to `1` second. The process always sleeps at least `5` seconds if the queue was empty in the previous queue run.

`-qf`
Processes saved messages in the queue once and does not `fork(2)`, but runs in the foreground.

`-qG` *name*
Processes jobs in queue group called *name* only.

`-q[!]I` *substr*
Limits processed jobs to those containing *substr* as a substring of the queue ID or not when `!` is specified.

`-q[!]Q` *substr*
Limits processed jobs to those quarantined jobs containing *substr* as a substring of the quarantine *reason* or not when `!` is specified.

`-q[!]R` *substr*
Limits processed jobs to those containing *substr* as a substring of one of the recipients or not when `!` is specified.

`-q[!]S` *substr*
Limits processed jobs to those containing *substr* as a substring of the sender or not when `!` is specified.

`-Q[`*reason*`]`
Quarantines a normal queue item with the given reason or unquarantines a quarantined queue item if no reason is given. This should only be used with some sort of item matching as described above.

`-r` *name*
An alternate and obsolete form of the `-f` flag.

`-R` *ret*
Identifies the information you want returned if the message bounces. *ret* can be `HDRS` for headers only or `FULL` for headers plus body.

`-t`
Reads message for recipients. `To:`,`Cc:`, and `Bcc:` lines are scanned for people to send to. The `Bcc:` line is deleted before transmission. Any addresses in the argument list is suppressed. The `NoRecipientAction` Processing Option can be used to change the behavior when no legal recipients are included in the message.

`-v`
Goes into verbose mode. Alias expansions are announced, and so forth.

`-V` *envid*
The indicated *envid* is passed with the envelope of the message and returned if the message bounces.

`-X` *logfile*
Logs all traffic in and out of `sendmail` in the indicated *logfile* for debugging mailer problems. This produces a lot of data very quickly and should be used sparingly.

Processing Options

There are a number of "random" options that can be set from a configuration file. Options are represented by a single character or by multiple character names. The syntax for the single character names of is:

`O`*xvalue*

This sets option *x* to be *value*. Depending on the option, *value* may be a string, an integer, a boolean (with legal values `t`, `T`, `f`, or `F`; the default is `TRUE`), or a time interval.

The multiple character or long names use this syntax:

`O` *Longname=argument*

This sets the option *Longname* to be *argument*. The long names are beneficial because they are easier to interpret than the single character names.

Not all processing options have single character names associated with them. In the list below, the multiple character name is presented first followed by the single character syntax enclosed in parentheses.

`AliasFile (A`*file*`)`
Specifies possible alias files.

`AliasWait (a` *N*`)`
If set, waits up to *N* minutes for an "@:@" entry to exist in the `aliases(4)` database before starting up. If it does not appear in *N* minutes, issues a warning. Defaults to 10 minutes.

`AllowBogusHELO`
Allows a `HELO SMTP` command that does not include a host name. By default this option is disabled.

`BadRcptThrottle=`*N*
If set and more than the specified number of recipients in a single SMTP envelope are rejected, sleeps for one second after each rejected RCPT command.

`BlankSub (B`*c*`)`
Sets the blank substitution character to *c*. Unquoted spaces in addresses are replaced by this character. Defaults to SPACE (that is, no change is made).

`CACertFile`
File containing one CA cert.

`CACertPath`
Path to directory with certs of CAs.

`CheckAliases (n)`
Validates the RHS of aliases when rebuilding the `aliases(4)` database.

`CheckpointInterval (C`*N*`)`
Checkpoints the queue every *N* (default `10`) addresses sent. If your system crashes during delivery to a large list, this prevents retransmission to any but the last *N* recipients.

`ClassFactor (z`*fact*`)`
The indicated factor *fact* is multiplied by the message class (determined by the `Precedence:` field in the user header and the `P` lines in the configuration file) and subtracted from the priority. Thus, messages with a higher `Priority:` are favored. Defaults to `1800`.

`ClientCertFile`
File containing the cert of the client, that is, this cert is used when `sendmail` acts as client.

`ClientKeyFile`
File containing the private key belonging to the client cert.

`ClientPortOptions`
Sets client SMTP options. The options are key=value pairs. Known keys are:

`Addr` *Address Mask*
Address Mask defaults to `INADDR_ANY`. The address mask can be a numeric address in dot notation or a network name.

`Family`
Address family (defaults to INET).

`Listen`
Size of listen queue (defaults to 10).

`Port`
Name/number of listening port (defaults to `smtp`).

`RcvBufSize`
The size of the TCP/IP receive buffer.

`SndBufSize`
The size of the TCP/IP send buffer.

`Modifier`
Options (flags) for the daemon. Can be:

`h`
Uses name of interface for `HELO` command.

If `h` is set, the name corresponding to the outgoing interface address (whether chosen by means of the `Connection` parameter or the default) is used for the `HELO`/`EHLO` command.

`ColonOkInAddr`
If set, colons are treated as a regular character in addresses. If not set, they are treated as the introducer to the RFC 822 "group" syntax. This option is on for version 5 and lower configuration files.

`ConnectionCacheSize (k`*N*`)`
The maximum number of open connections that are to be cached at a time. The default is `1`. This delays closing the current connection until either this invocation of `sendmail` needs to connect to another host or it terminates. Setting it to `0` defaults to the old behavior, that is, connections are closed immediately.

`ConnectionCacheTimeout (K`*timeout*`)`
The maximum amount of time a cached connection is permitted to idle without activity. If this time is exceeded, the connection is immediately closed. This value should be small (on the order of ten minutes). Before `sendmail` uses a cached connection, it always sends a `NOOP` (no operation) command to check the connection. If the `NOOP` command fails, it reopens the connection. This keeps your end from failing if the other end times out. The point of this option is to be a good network neighbor and avoid using up excessive resources on the other end. The default is five minutes.

`ConnectionRateThrottle`
The maximum number of connections permitted per second. After this many connections are accepted, further connections are delayed. If not set or <= 0, there is no limit.

`ConnectionRateWindowSize`
Define the length of the interval for which the number of incoming connections is maintained. The default is 60 seconds.

`ControlSocketName`
Name of the control socket for daemon management. A running `sendmail` daemon can be controlled through this Unix domain socket. Available commands are: `help`, `restart`, `shutdown`, and `status`. The `status` command returns the current number of daemon children, the free disk space (in blocks) of the queue directory, and the load average of the machine expressed as an integer. If not set, no control socket is available. For the sake of security, this Unix domain socket must be in a directory which is accessible only by root; `/var/spool/mqueue/.smcontrol` is recommended for the socket name.

`CRLFile`
File containing certificate revocation status, useful for X.509v3 authentication.

`DaemonPortOptions (O`*options*`)`
Sets server SMTP options. The options are *key*=*value* pairs. Known keys are:

`Name`
User-definable name for the daemon (defaults to "`Daemon#`"). Used for error messages and logging.

`Addr`
Address mask (defaults `INADDR_ANY`).

The address mask may be a numeric address in dot notation or a network name.

`Family`
Address family (defaults to `INET`).

`InputMailFilters`
List of input mail filters for the daemon.

`Listen`
Size of listen queue (defaults to `10`).

`Modifier`
Options (flags) for the daemon; can be a sequence (without any delimiters) of:

`a`
Requires authentication.

`b`
Binds to interface through which mail has been received.

`c`
Performs hostname canonification (`.cf`).

`f`
Requires fully qualified hostname (`.cf`).

`h`
Uses name of interface for `HELO` command.

`u`
Allows unqualified addresses (`.cf`).

`C`
Does not perform hostname canonification.

`E`
Disallows `ETRN` (see RFC 2476).

`Name`
User-definable name for the daemon (defaults to `Daemon#`). Used for error messages and logging.

`Port`
Name/number of listening port (defaults to `smtp`).

`ReceiveSize`
The size of the TCP/IP receive buffer.

`SendSize`
The size of the TCP/IP send buffer.

`children`
Maximum number of children per daemon. See `MaxDaemonChildren`.

`DeliveryMode`
Delivery mode per daemon. See `DeliveryMode`.

`refuseLA`
`RefuseLA` per daemon.

`delayLA`
`DelayLA` per daemon.

`queueLA`
`QueueLA` per daemon.

`sendmail` listens on a new socket for each occurrence of the `DaemonPortOptions` option in a configuration file.

`DataFileBufferSize`
Sets the threshold, in bytes, before a memory-bases queue data file becomes disk-based. The default is 4096 bytes.

`DeadLetterDrop`
Defines the location of the system-wide dead.letter file, formerly hard-coded to `/var/tmp/dead.letter`. If this option is not set (the default), `sendmail` does not attempt to save to a system-wide `dead.letter` file in the event it cannot bounce the mail to the user or postmaster. Instead, it renames the `qf` file as it has in the past when the `dead.letter` file could not be opened.

`DefaultCharSet`
Sets the default character set to use when converting unlabeled 8 bit input to MIME.

`DefaultUser (g`*gid*`) or (u`*uid*`)`
Sets the default group ID for mailers to run in to *gid* or set the default userid for mailers to *uid*. Defaults to `1`. The value can also be given as a symbolic group or user name.

`DelayLA=`*LA*
When the system load average exceeds *LA*, `sendmail` sleeps for one second on most SMTP commands and before accepting connections.

`DeliverByMin=`*time*
Sets minimum time for `Deliver By SMTP Service Extension` (RFC 2852). If `0`, no time is listed, if less than `0`, the extension is not offered, if greater than `0`, it is listed as minimum time for the `EHLO` keyword `DELIVERBY`.

`DeliveryMode (d`*x*`)`
Delivers in mode *x*. Legal modes are:

`i`
Delivers interactively (synchronously).

`b`
Delivers in background (asynchronously).

`d`
Deferred mode. Database lookups are deferred until the actual queue run.

`q`
Just queues the message (delivers during queue run).

Defaults to `b` if no option is specified, `i` if it is specified but given no argument (that is, `Od` is equivalent to `Odi`).

`DHParameters`
File containing the DH parameters.

`DialDelay`
If a connection fails, waits this many seconds and tries again. Zero means "do not retry".

`DontBlameSendmail`
If set, overrides the file safety checks. This compromises system security and should not be used. See http://www.sendmail.org/tips/dontBlameSendmail for more information.

`DontExpandCnames`
If set, $[... $] lookups that do DNS-based lookups do not expand CNAME records.

`DontInitGroups`
If set, the `initgroups(3C)` routine is never invoked. If you set this, agents run on behalf of users only have their primary (`/etc/passwd`) group permissions.

`DontProbeInterfaces`
If set, `sendmail` does not insert the names and addresses of any local interfaces into the `$=w class`. If set, you must also include support for these addresses, otherwise mail to addresses in this list bounces with a configuration error.

`DontPruneRoutes (R)`
If set, does not prune route-addr syntax addresses to the minimum possible.

`DoubleBounceAddress`
If an error occurs when sending an error message, sends that "double bounce" error message to this address.

`EightBitMode (8)`
Uses 8–bit data handling. This option requires one of the following keys. The key can selected by using just the first character, but using the full word is better for clarity.

`mimify`
Does any necessary conversion of `8BITMIME` to 7–bit.

`pass`
Passes unlabeled 8–bit input through as is.

`strict`
Rejects unlabeled 8–bit input.

`ErrorHeader (E`*file/message*`)`
Appends error messages with the indicated message. If it begins with a slash, it is assumed to be the pathname of a file containing a message (this is the recommended setting). Otherwise, it is a literal message. The error file might contain the name, email address, and/or phone number of a local postmaster who could provide assistance to end users. If the option is missing or `NULL`, or if it names a file which does not exist or which is not readable, no message is printed.

`ErrorMode (e`*x*`)`
Disposes of errors using mode *x*. The values for *x* are:

`e`
Mails back errors and gives `0` exit status always.

`m`
Mails back errors.

`p`
Prints error messages (default).

`q`
No messages, just gives exit status.

`w`
Writes back errors (mail if user not logged in).

`FallbackMXhost (V`*fallbackhost*`)`
If specified, the *fallbackhost* acts like a very low priority MX on every host. This is intended to be used by sites with poor network connectivity.

`FallBackSmartHost`
If specified, the *fallBackSmartHost* is used in a last-ditch effort for each host. This is intended to be used by sites with "fake internal DNS". That is, a company whose DNS accurately reflects the world inside that company's domain but not outside.

`FastSplit`
If set to a value greater than zero (the default is one), it suppresses the MX lookups on addresses when they are initially sorted, that is, for the first delivery attempt. This usually results in faster envelope splitting unless the MX records are readily available in a local DNS cache. To enforce initial sorting based on MX records set `FastSplit` to zero. If the mail is submitted directly from the command line, then the value also limits the number of processes to deliver the envelopes; if more envelopes are created they are only queued up and must be taken care of by a queue run. Since the default submission method is by way of SMTP (either from a MUA or by way of the Message Submission Program [MSP]), the value of `FastSplit` is seldom used to limit the number of processes to deliver the envelopes.

`ForkEachJob (Y)`
If set, delivers each job that is run from the queue in a separate process. Use this option if you are short of memory, since the default tends to consume considerable amounts of memory while the queue is being processed.

`ForwardPath (J`*`path`*`)`
Sets the path for searching for users' `.forward` files. The default is `$z/.forward`. Some sites that use the automounter may prefer to change this to `/var/forward/$u` to search a file with the same name as the user in a system directory. It can also be set to a sequence of paths separated by colons; `sendmail` stops at the first file it can successfully and safely open. For example, `/var/forward/$u:$z/.forward` searches first in `/var/forward/` *username* and then in *~username*`/.forward` (but only if the first file does not exist). Refer to the NOTES section for more information.

`HeloName=`*`name`*
Sets the name to be used for `HELO`/`EHLO` (instead of `$j`).

`HelpFile (H`*`file`*`)`
Specifies the help file for SMTP.

`HoldExpensive (c)`
If an outgoing mailer is marked as being expensive, does not connect immediately.

`HostsFile`
Sets the file to use when doing "file" type access of host names.

`HostStatusDirectory`
If set, host status is kept on disk between `sendmail` runs in the named directory tree. If a full path is not used, then the path is interpreted relative to the queue directory.

`IgnoreDots (i)`
Ignores dots in incoming messages. This is always disabled (that is, dots are always accepted) when reading SMTP mail.

`LogLevel (L`*`n`*`)`
Sets the default log level to *n*. Defaults to `9`.

`(M`*`x value`*`)`
Sets the macro *x* to *value*. This is intended only for use from the command line.

`MailboxDatabase`
Type of lookup to find information about local mail boxes, defaults to `pw` which uses `getpwnam(3C)`. Other types can be introduced by adding them to the source code, see `libsm/mbdb.c` for details.

`MatchGECOS (G)`
Tries to match recipient names using the GECOS field. This allows for mail to be delivered using names defined in the GECOS field in `/etc/passwd` as well as the login name.

`MaxDaemonChildren`
The maximum number of children the daemon permits. After this number, connections are rejected. If not set or <=0, there is no limit.

`MaxHopCount (h`*N*`)`
The maximum hop count. Messages that have been processed more than *N* times are assumed to be in a loop and are rejected. Defaults to `25`.

`MaxMessageSize`
The maximum size of messages that are accepted (in bytes).

`MaxMimeHeaderLength=`*M*`[/`*N*`]`
Sets the maximum length of certain MIME header field values to *M* characters. For some of these headers which take parameters, the maximum length of each parameter is set to *N* if specified. If `/`*N* is not specified, one half of *M* is used. By default, these values are `0`, meaning no checks are done.

`MaxNOOPCommands=`*N*
Overrides the default of 20 for the number of useless commands.

`MaxQueueChildren=`*N*
When set, this limits the number of concurrent queue runner processes to *N*. This helps to control the amount of system resources used when processing the queue. When there are multiple queue groups defined and the total number of queue runners for these queue groups would exceed `MaxQueueChildren` then the queue groups are not all run concurrently. That is, some portion of the queue groups run concurrently such that `MaxQueueChildren` is not be exceeded, while the remaining queue groups are run later (in round robin order). See `MaxRunnersPerQueue`.

`MaxQueueRunSize`
If set, limits the maximum size of any given queue run to this number of entries. This stops reading the queue directory after this number of entries is reached; job priority is not used. If not set, there is no limit.

`MaxRunnersPerQueue=`*N*
This sets the default maximum number of queue runners for queue groups. Up to *N* queue runners work in parallel on a queue group's messages. This is useful where the processing of a message in the queue might delay the processing of subsequent messages. Such a delay can be the result of non-erroneous situations such as a low bandwidth connection. The can be overridden on a per queue group basis by setting the `Runners` option. The default is `1` when not set.

`MeToo (m)`
Sends to me too, even if I am in an alias expansion.

`MaxRecipientsPerMessage`
If set, allows no more than the specified number of recipients in an SMTP envelope. Further recipients receive a 452 error code and are deferred for the next delivery attempt.

`MinFreeBlocks (b`*N*`/`*M*`)`
Insists on at least *N* blocks free on the file system that holds the queue files before accepting email by way of SMTP. If there is insufficient space, `sendmail` gives a `452` response to the `MAIL` command. This invites the sender to try again later. The optional *M* is a maximum message size advertised in the `ESMTP EHLO` response. It is currently otherwise unused.

`MinQueueAge`
Specifies the amount of time a job must sit in the queue between queue runs. This allows you to set the queue run interval low for better responsiveness without trying all jobs in each run. The default value is 0.

`MustQuoteChars`
Specifies the characters to be quoted in a full name phrase. `&,;:\()[]` are quoted automatically.

`NiceQueueRun`
Specifies the priority of queue runners. See `nice(1)`.

`NoRecipientAction`
Sets action if there are no legal recipient files in the message. The legal values are:

`add-apparently-to`
Adds an `Apparently-to:` header with all the known recipients (which may expose blind recipients).

`add-bcc`
Adds an empty `Bcc:` header.

`add-to`
Adds a `To:` header with all the known recipients (which may expose blind recipients).

`add-to-undisclosed`
Adds a `To: undisclosed-recipients:` header.

`none`
Does nothing, that is, leaves the message as it is.

`OldStyleHeaders (o)`
Assumes that the headers may be in old format, that is, spaces delimit names. This actually turns on an adaptive algorithm: if any recipient address contains a comma, parenthesis, or angle bracket, it is assumed that commas already exist. If this flag is not on, only commas delimit names. Headers are always output with commas between the names.

`OperatorChars or $o`
Defines the list of characters that can be used to separate the components of an address into tokens.

`PidFile`
Specifies the filename of the `pid` file. The default is `/var/run/sendmail.pid`. The filename is macro-expanded before it is opened, and unlinked when `sendmail` exits.

`PostmasterCopy (P`*postmaster*`)`
If set, copies of error messages are sent to the named *postmaster*. Only the header of the failed message is sent. Since most errors are user problems, this is probably not a good idea on large sites, and arguably contains all sorts of privacy violations, but it seems to be popular with certain operating systems vendors.

`PrivacyOptions (p`*opt,opt,...*`)`
Sets privacy options. Privacy is really a misnomer; many of these options are just a way of insisting on stricter adherence to the SMTP protocol.

The `goaway` pseudo-flag sets all flags except `noreceipts`, `restrictmailq`, `restrictqrun`, `restrictexpand`, `noetrn`, and `nobodyreturn`. If `mailq` is restricted, only people in the same group as the queue directory can print the queue. If queue runs are restricted, only root and the owner of the queue directory can run the queue. The `restrict-expand` pseudo-flag instructs sendmail to drop privileges when the -bv option is given by users who are neither root nor the `TrustedUser` so users cannot read private aliases, forwards, or `:include:` files. It adds the `NonRootSafeAddr` to the "DontBlame-Sendmail" option to prevent misleading unsafe address warnings. It also overrides the `-v` (verbose) command line option to prevent information leakage. Authentication Warnings add warnings about various conditions that may indicate attempts to fool the mail system, such as using an non-standard queue directory.

The options can be selected from:

`authwarnings`
Puts `X-Authentication-Warning:` headers in messages.

`goaway`
Disallows essentially all SMTP status queries.

`needexpnhelo`
Insists on `HELO` or `EHLO` command before `EXPN`.

`needmailhelo`
Insists on `HELO` or `EHLO` command before `MAIL`.

`needvrfyhelo`
Insists on `HELO` or `EHLO` command before `VRFY`.

`noactualrecipient`
Do not put an X-Actual-Recipient line in a DNS that reveals the actual account to which an address is mapped.

`noetrn`
Disallows `ETRN` entirely.

`noexpn`
Disallows `EXPN` entirely.

`noreceipts`
Prevents return receipts.

`nobodyreturn`
Does not return the body of a message with DSNs.

`novrfy`
Disallows `VRFY` entirely.

`public`
Allows open access.

`restrictexpand`
Restricts `-bv` and `-v` command line flags.

`restrictmailq`
Restricts `mailq` command.

`restrictqrun`
Restricts `-q` command line flag.

`ProcessTitlePrefix` *string*
Prefixes the process title shown on "`/usr/ucb/ps auxww`" listings with *string*. The string is macro processed.

`QueueDirectory (Q`*dir*`)`
Uses the named *dir* as the queue directory.

`QueueFactor (q`*factor*`)`
Uses *factor* as the multiplier in the map function to decide when to just queue up jobs rather than run them. This value is divided by the difference between the current load average and the load average limit (`x` flag) to determine the maximum message priority to be sent. Defaults to `600000`.

`QueueFileMode=`*mode*
Defaults permissions for queue files (octal). If not set, `sendmail` uses `0600` unless its real and effective `uid` are different in which case it uses `0644`.

`QueueLA (x`*LA*`)`
When the system load average exceeds *LA*, just queues messages (that is, does not try to send them). Defaults to eight times the number of processors online when `sendmail` starts.

`QueueSortOrder=`*algorithm*
Sets the algorithm used for sorting the queue. Only the first character of the value is used. Legal values are `host` (to order by the name of the first host name of the first recipient), `filename` (to order by the name of the queue file name), `time` (to order by the submission/creation time), `random` (to order randomly), `modification` (to order by the modification time of the `qf` file (older entries first)), `none` (to not order), and `priority` (to order by message priority). Host ordering makes better use of the connection cache, but may tend to process low priority messages that go to a single host over high priority

messages that go to several hosts; it probably shouldn't be used on slow network links. Filename and modification time ordering saves the overhead of reading all of the queued items before starting the queue run. Creation (submission) time ordering is almost always a bad idea, since it allows large, bulk mail to go out before smaller, personal mail, but may have applicability on some hosts with very fast connections. Random is useful if several queue runners are started by hand which try to drain the same queue since odds are they are working on different parts of the queue at the same time. Priority ordering is the default.

`QueueTimeout (T`*rtime*`/`*wtime*`)`
Sets the queue timeout to *rtime*. After this interval, messages that have not been successfully sent are returned to the sender. Defaults to five days (`5d`). The optional *wtime* is the time after which a warning message is sent. If it is missing or `0`, then no warning messages are sent.

`RandFile`
File containing random data (use prefix `file:`) or the name of the UNIX socket if EGD is used (use prefix `egd:`). Note that Solaris supports `random(7D)`, so this does not need to be specified.

`RecipientFactor (y`*fact*`)`
The indicated factor *fact* is added to the priority (thus *lowering* the priority of the job) for each recipient, that is, this value penalizes jobs with large numbers of recipients. Defaults to `30000`.

`RefuseLA (X`*LA*`)`
When the system load average exceeds *LA*, refuses incoming SMTP connections. Defaults to 12 times the number of processors online when `sendmail` starts.

`RejectLogInterval`
Log interval when refusing connections for this long (default: 3h).

`ResolverOptions (I)`
Tunes DNS lookups.

`RetryFactor (Z`*fact*`)`
The indicated factor *fact* is added to the priority every time a job is processed. Thus, each time a job is processed, its priority is decreased by the indicated value. In most environments this should be positive, since hosts that are down are all too often down for a long time. Defaults to `90000`.

`RrtImpliesDsn`
If this option is set, a `Return-Receipt-To:` header causes the request of a DSN, which is sent to the envelope sender as required by RFC 1891, not to the address given in the header.

`RunAsUser`
If set, becomes this user when reading and delivering mail. Intended for use of firewalls where users do not have accounts.

`SafeFileEnvironment`
If set, `sendmail` does a `chroot` into this directory before writing files.

`SaveFromLine (f)`
Saves Unix-style `From` lines at the front of headers. Normally they are assumed redundant and discarded.

`SendMimeErrors (j)`
If set, sends error messages in MIME format (see RFC 2045 and RFC 1344 for details). If disabled, `sendmail` does not return the DSN keyword in response to an `EHLO` and does not do Delivery Status Notification processing as described in RFC 1891.

`ServerCertFile`
File containing the cert of the server, that is, this cert is used when `sendmail` acts as server.

`ServerKeyFile`
File containing the private key belonging to the server cert.

`ServiceSwitchFile`
Defines the path to the service-switch file. Since the service-switch file is defined in the Solaris operating environment this option is ignored.

`SevenBitInput (7)`
Strips input to seven bits for compatibility with old systems. This should not be necessary.

`SharedMemoryKey`
Specifies key to use for shared memory segment. If not set (or `0`), shared memory is not be used. If this option is set, `sendmail` can share some data between different instances. For example, the number of entries in a queue directory or the available space in a file system. This allows for more efficient program execution, since only one process needs to update the data instead of each individual process gathering the data each time it is required.

`SharedMemoryKeyFile=`*file*
If `SharedMemoryKeyFile` is set to `-1`, the automatically selected shared memory key will be stored in the specified file.

`SingleLineFromHeader`
If set, `From:` lines that have embedded newlines are unwrapped onto one line.

`SingleThreadDelivery`
If this option and the `HostStatusDirectory` option are both set, uses single thread deliveries to other hosts.

`SmtpGreetingMessage or $e`
Specifies the initial SMTP greeting message.

`SoftBounce`
If set, issue temporary errors (`4xy`) instead of permanent errors (`5xy`). This can be useful during testing of a new configuration to avoid erroneous bouncing of mail.

`StatusFile (S`*file*`)`
Logs statistics in the named *file*. By default, this is `/etc/mail/sendmail.st`. As root, you must `touch(1)` this file to enable `mailstats(1)`.

`SuperSafe (s)`
This option can be set to `True`, `False`, `Interactive`, or `PostMilter`. If set to `True`, `sendmail` is set to super-safe when running things, that is, always instantiate the queue file, even if you are going to attempt immediate delivery. `sendmail` always instantiates the queue file before returning control to the client under any circumstances. This should really always be set to `True`. The `Interactive` value has been introduced in `8.12` and can be used together with `DeliveryMode=i`. It skips some synchronization calls which are effectively doubled in the code execution path for this mode. If set to `PostMilter`, `sendmail` defers synchronizing the queue file until any milters have signaled acceptance of the message. `PostMilter` is useful only when `sendmail` is running as an SMTP server; in all other situations it acts the same as True.

`TempFileMode (F`*mode*`)`
Specifies the file mode for queue files.

`Timeout (r`*timeouts*`)`
Timeout reads after time interval. The *timeouts* argument is a list of *keyword*=*value* pairs. All but *command* apply to client SMTP. For backward compatibility, a timeout with no *keyword*= part is set all of the longer values. The recognized timeouts and their default values, and their minimum values specified in RFC 1123 section 5.3.2 are:

`aconnect`
all connections for a single delivery attempt [0, unspecified]

`command`
command read [1h, 5m]

`connect`
initial connect [0, unspecified]

`control`
complete control socket transaction [2m, none]

`datablock`
data block read [1h, 3m]

`datafinal`
reply to final `.` in data [1h, 10m]

`datainit`
reply to `DATA` command [5m, 2m]

`fileopen`
file open [60sec, none]

`helo`
reply to `HELO` or `EHLO` command [5m, none]

`hoststatus`
host retry [30m, unspecified]

`iconnect`
first attempt to connect to a host [0, unspecified]

`ident`
`IDENT` protocol timeout [5s, none]

`initial`
wait for initial greeting message [5m, 5m]

`lhlo`
wait for reply to an LMTP LHLO command [2m, unspecified]

`mail`
reply to `MAIL` command [10m, 5m]

`misc`
reply to `NOOP` and `VERB` commands [2m, none]

`queuereturn`
undeliverable message returned [5d]

`queuewarn`
deferred warning [4h]

`quit`
reply to `QUIT` command [2m, none]

`rcpt`
reply to `RCPT` command [1h, 5m]

`resolver.retrans`
Resolver's retransmission time interval (in seconds) [varies]. Sets both `Timeout.resolver.retrans.first` and `Timeout.resolver.retrans.normal`.

`resolver.retrans.first`
Resolver's retransmission time interval (in seconds) for the first attempt to deliver a message [varies].

`resolver.retrans.normal`
Resolver's retransmission time interval (in seconds) for all look-ups except the first delivery attempt [varies].

`resolver.retry`
Number of times to retransmit a resolver query [varies]. Sets both `Timeout.resolver.retry.first` and `Timeout.resolver.retry.normal`.

`resolver.retry.first`
Number of times to retransmit a resolver query for the first attempt to deliver a message [varies].

`resolver.retry.normal`
Number of times to retransmit a resolver query for all look-ups except the first delivery attempt [varies].

`rset`
reply to `RSET` command [5m, none]

`starttls`
response to an `SMTP STARTTLS` command [1h]

`TimeZoneSpec (t`*tzinfo*`)`
Sets the local time zone info to *tzinfo*, for example, "PST8PDT". Actually, if this is not set, the `TZ` environment variable is cleared (so the system default is used); if set but null, the user's `TZ` variable is used, and if set and non-null, the `TZ` variable is set to this value.

`TLSSrvOptions`
If this option is 'V', then no client verification is performed,that is, the server does not ask for a certificate.

`TrustedUser`
The user parameter can be a user name (looked up in the passwd map) or a numeric user id. Trusted user for file ownership and starting the daemon. If set, generated alias databases and the control socket (if configured) are automatically owned by this user.

`TryNullMXList (w)`
If you are the "best" (that is, lowest preference) MX for a given host, you should normally detect this situation and treat that condition specially, by forwarding the mail to a UUCP feed, treating it as local, or whatever. However, in some cases (such as Internet firewalls) you may want to try to connect directly to that host as though it had no MX records at all. Setting this option causes `sendmail` to try this. The downside is that errors in your configuration are likely to be diagnosed as "host unknown" or "message timed out" instead of something more meaningful. This option is deprecated.

`UnixFromLine or $l`
The "From " line used when sending to files or programs.

`UnsafeGroupWrites`
If set, group-writable :include: and .forward files are considered "unsafe", that is, programs and files cannot be directly referenced from such files.

`UseErrorsTo (l)`
If there is an `Errors-To`: header, sends error messages to the addresses listed there. They normally go to the envelope sender. Use of this option causes `sendmail` to violate RFC 1123. This option is not recommended and deprecated.

`UseMSP`
Uses as mail submission program, that is, allows group writable queue files if the group is the same as that of a set-group-id `sendmail` binary.

`UserDatabaseSpec (U)`
Defines the name and location of the file containing User Database information.

`Verbose (v)`
Runs in verbose mode. If this is set, `sendmail` adjusts the `HoldExpensive` and `DeliveryMode` options so that all mail is delivered completely in a single job so that you can see the entire delivery process. The `Verbose` option should *never* be set in the configuration file; it is intended for command line use only.

`XscriptFileBufferSize`
Sets the threshold, in bytes, before a memory-bases queue transcript file becomes disk-based. The default is 4096 bytes.

If the first character of the user name is a vertical bar, the rest of the user name is used as the name of a program to pipe the mail to. It may be necessary to quote the name of the user to keep `sendmail` from suppressing the blanks from between arguments.

If invoked as `newaliases`, `sendmail` rebuilds the alias database, so long as the `/etc/mail/aliases*` files are owned by root *and* root has exclusive write permission. If invoked as `mailq`, `sendmail` prints the contents of the mail queue.

Operands

address
address of an intended recipient of the message being sent.

Usage

See `largefile(5)` for the description of the behavior of `sendmail` when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

Exit Status

`sendmail` returns an exit status describing what it did. The codes are defined in `/usr/include/sysexits.h`.

`EX_OK`
Successful completion on all addresses.

`EX_NOUSER`
User name not recognized.

`EX_UNAVAILABLE`
Catchall. Necessary resources were not available.

`EX_SYNTAX`
Syntax error in address.

`EX_SOFTWARE`
Internal software error, including bad arguments.

`EX_OSERR`
Temporary operating system error, such as “cannot fork”.

`EX_NOHOST`
Host name not recognized.

`EX_TEMPFAIL`
Message could not be sent immediately, but was queued.

Environment Variables

No environment variables are used. However, sendmail's start-up script, invoked by `svcadm(1M)`, reads `/etc/default/sendmail`. In this file, if the variable `ETRN_HOSTS` is set, the start-up script parses this variable and invokes `etrn(1M)` appropriately. `ETRN_HOSTS` should be of the form:

```
"s1:c1.1,c1.2        s2:c2.1 s3:c3.1,c3.2,c3.3"
```

That is, white-space separated groups of *server:client* where *client* can be one or more comma-separated names. The *:client* part is optional. *server* is the name of the server to prod; a mail queue run is requested for each *client* name. This is comparable to running:

```
/usr/lib/sendmail -qR client
```

on the host *server*.

Files

`dead.letter`
Unmailable text

`/etc/default/sendmail`
Contains default settings. You can override some of the settings by command line options.

`/etc/mail/aliases`
Mail aliases file (ASCII)

`/etc/mail/aliases.db`
Database of mail aliases (binary)

`/etc/mail/aliases.dir`
Database of mail aliases (binary)

`/etc/mail/aliases.pag`
Database of mail aliases (binary)

`/etc/mail/sendmail.cf`
Defines environment for `sendmail`

`/etc/mail/submit.cf`
Defines environment for MSP

`/etc/mail/trusted-users`
Lists users that are “trusted”, that is, able to set their envelope from address using `-f` without generating a warning message. Note that this file is consulted by the default `sendmail.cf`, but not by the default `submit.cf`, in which the line referring to `/etc/mail/trusted-users` is commented out. See `sendmail(4)` for instructions on making changes to `submit.cf` and `sendmail.cf`.

`/var/spool/clientmqueue/*`
Temporary files and queued mail

`/var/spool/mqueue/*`
Temporary files and queued mail

`~/.forward`
List of recipients for forwarding messages

`/usr/include/libmilter/README`
Describes the steps needed to compile and run a filter

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	service/network/smtp/sendmail

See Also `svcs(1)`, `biff(1B)`, `mail(1)`, `mailq(1)`, `mailx(1)`, `nice(1)`, `check-hostname(1M)`, `check-permissions(1M)`, `etrn(1M)`, `newaliases(1M)`, `svcadm(1M)`, `svccfg(1M)`, `fork(2)`, `getpwnam(3C)getusershell(3C)resolver(3RESOLV)aliases(4)`, `hosts(4)`, `sendmail(4)`, `shells(4)`, `attributes(5)`, `largefile(5)`, `smf(5)`, `random(7D)`

`tcpd`(1M), `hosts_access`(4) in the `security/tcp-wrapper` package.

RFC 2821 *Simple Mail Transfer Protocol*, John Klensin, April 2001.

RFC 2822 *Internet Message Format*, Pete Resnick, April 2001.

sendmail, Third Edition, Bryan Costales with Eric Allman, O'Reilly & Associates, Inc., 2003.

http://www.sendmail.org

http://www.milter.org

Notes The `sendmail` program requires a fully qualified host name when starting. A script has been included to help verify if the host name is defined properly (see `check-hostname(1M)`).

The permissions and the ownership of several directories have been changed in order to increase security. In particular, access to `/etc/mail` and `/var/spool/mqueue` has been restricted.

Security restrictions have been placed users using `.forward` files to pipe mail to a program or redirect mail to a file. The default shell (as listed in `/etc/passwd`) of these users must be listed in `/etc/shells`. This restriction does not affect mail that is being redirected to another alias.

Additional restrictions have been put in place on `.forward` and `:include:` files. These files and the directory structure that they are placed in cannot be group- or world-writable. See `check-permissions(1M)`.

If you have interfaces that map to domains that have MX records that point to non-local destinations, you might need to enable the `DontProbeInterfaces` option to enable delivery to those destinations. In its default startup behavior, `sendmail` probes each interface and adds an

interface's IP addresses, as well as any domains that those addresses map to, to its list of domains that are considered local. For domains thus added, being on the list of local domains is equivalent to having a 0-preference MX record, with `localhost` as the MX value. If this is not the result you want, enable `DontProbeInterfaces`.

Name sftp-server – SFTP server subsystem

Synopsis `/usr/lib/ssh/sftp-server [-f` *log_facility*`] [-l` *log_level*`]`

Description `sftp-server` implements the server side of the SSH File Transfer Protocol as defined in the IETF `draft-ietf-secsh-filexfer`.

`sftp-server` is a subsystem for `sshd(1M)` and must not be run directly. Command-line flags to `sftp-server` should be specified in the Subsystem declaration. See `sshd_config(4)` for more information.

To enable the `sftp-server` subsystem for `sshd` add the following to `/etc/ssh/sshd_config`:

```
Subsystem   sftp     /usr/lib/ssh/sftp-server
```

To run `sftp-server` in a `chroot` configuration, use `internal-sftp` instead of `/usr/lib/ssh/sftp-server`. Otherwise, the `chroot` directory must contain the necessary files and directories to support the user's session. See the `ChrootDirectory` and `Subsystem` options in `sshd_config(4)`) for more information on how `sshd` and `sftp-server` work with `chroot(2)`.

See `sshd_config(4)` for a description of the format and contents of that file.

There is no relationship between the protocol used by `sftp-server` and the FTP protocol (RFC 959) provided by `in.ftpd`.

For logging to work, `sftp-server` must be able to access `/dev/log`. Use of `sftp-server` in a `chroot` configuration therefore requires that `syslogd(1M)` establish a logging socket inside the `chroot` directory.

Options Valid options are listed below. As stated above, these options, if used, are specified in the Subsystem declaration of `sshd_config`.

`-f` *log_facility*
Specifies the facility code that is used when logging messages from `sftp-server`. The possible values are: `DAEMON`, `USER`, `AUTH`, `LOCAL0`, `LOCAL1`, `LOCAL2`, `LOCAL3`, `LOCAL4`, `LOCAL5`, `LOCAL6`, `LOCAL7`. The default is `AUTH`.

`-l` *log_level*
Specifies which messages will be logged by `sftp-server`. The possible values are: `QUIET`, `FATAL`, `ERROR`, `INFO`, `VERBOSE`, `DEBUG`, `DEBUG1`, `DEBUG2`, and `DEBUG3`. `INFO` and `VERBOSE` log transactions that `sftp-server` performs on behalf of the client. `DEBUG` and `DEBUG1` are equivalent. `DEBUG2` and `DEBUG3` each specify higher levels of debugging output. The default is `ERROR`.

`-u` *umask*
Sets an explicit `umask(2)` to be applied to newly-created files and directories, instead of the user's default mask.

Exit Status The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

Files `/usr/lib/ssh/sftp-server` Server-side binary.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	service/network/ssh
Interface Stability	Committed

See Also `sftp(1)`, `ssh(1)`, `ssh-add(1)`, `ssh-keygen(1)`, `sshd(1M)`, `syslogd(1M)`, `chroot(2)`, `umask(2)`, `sshd_config(4)`, `attributes(5)`

Name shadowd – shadow migration daemon

Synopsis

```
/usr/lib/fs/shadowd

svc:/system/filesystem/shadowd:default
```

Description `shadowd` is a daemon that provides background worker threads to migrate data for a shadow migration. A shadow migration gradually moves data from a source file system into a new "shadow" file system. Users can access and change their data within the shadow file system while migration is occurring.

The `shadowd` service is managed by the service management facility, `smf(5)`. Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. The service's status can be queried using the `svcs(1)` command.

The `svccfg(1M)` command can be used to manage the following parameter related to `shadowd`:

```
config_params/shadow_threads
```

Number of threads to devote to background migration of data. These threads are global to the entire system, and increasing the number can increase concurrency and the overall speed of migration at the expense of increased resource consumption (network, I/O, and CPU).

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/file-system/zfs
Interface Stability	Unstable

See Also `shadowstat(1M)`, `zfs(1M)`, `attributes(5)`

Name shadowstat – report shadow migration statistics

Synopsis

```
/usr/sbin/shadowstat [count]
```

Description The `shadowstat` utility reports statistics on in-progress shadow migrations. A line of output is presented for each migrating file system; the output looks similar to the following:

```
                    EST
                BYTES   BYTES           ELAPSED
DATASET         XFRD    LEFT    ERRORS  TIME
pool/newfs      2.10M   28.8G   -       00:02:45
```

Each line identifies bytes transferred thus far, a rough estimate of bytes left to transfer, number of migration errors, and the time elapsed since the migration was started.

If *count* is unspecified, updated values are presented every few seconds until the program is terminated. If *count* is specified, updates only occur *count* times, after which the program exits.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/file-system/zfs
Interface Stability	Unstable

See Also `shadowd(1M)`, `zfs(1M)`, `attributes(5)`

Name share – display file system shares or make local file system available for mounting by remote systems

Synopsis

```
share [-F protocol] -a

share [-F protocol] [-o options] [-d description] pathname [sharename]

share [-F protocol] [-A]
```

Description The `share` command defines and publishes a file system share, which means the file system is available for mounting through a sharing protocol.

If the `-F` *protocol* option is omitted, the first file sharing protocol listed in `/etc/dfs/fstypes` is used as the default.

For a description of NFS-specific share options, see `share_nfs`(1M). For a description of SMB specific share options, see `share_smb`(1M).

Using the `share` command to define and publish an NFS or SMB share of a ZFS file system is considered a legacy operation. Consider setting the `share.nfs` property or using the `zfs share` command to define and publish an NFS or an SMB share of a ZFS file system. For more information, see `share_nfs`(1M) and `share_smb`(1M).

In the third form of `share` command, as shown in the Synopsis above, `share` displays published shares or, with the `-A` option, displays all configured (defined) shares.

Options

`-F` *protocol*
Specify the file sharing protocol.

`-o` *specific_options*

`rw`
Share *pathname* is published with read and write access to all clients. This is the default behavior.

`rw=`*client*[`:`*client*]...
Share *pathname* is published with read and write access only to the listed clients. No other systems can access the share pathname.

`ro`
Share pathname is published with read-only access to all clients.

`ro=`*client*[`:`*client*]...
Share pathname is published with read-only access only to the listed clients. No other systems can access the share pathname.

Separate multiple options with commas. Separate multiple operands for an option with colons. See EXAMPLES.

`-d`
Provide a comment that describes the file system share to be published.

`-a`
Publish all defined shares.

`-A`
Display all defined shares.

Examples

EXAMPLE 1 Publishing an NFS Share With Read-Only Access

The following command defines and publishes an NFS share of `/ufsfs` with read-only access.

```
# share -F nfs -o ro /ufsfs
```

EXAMPLE 2 Publishing an NFS Share with Multiple Share Options

The following command defines and publishes an NFS share of the `/export/manuals` file system with a `netgroup` called `users_nfs` who have read-only access and users from specified hosts who have read and write access.

```
# share -F nfs -o ro=users_nfs,rw=host1:host2:host3 /export/manuals
```

Files

`/etc/dfs/dfstab`
This file is obsolete. An SMF service publishes NFS or SMB shares at boot time.

`/etc/dfs/fstypes`
List of file-sharing protocols. NFS is the default file sharing protocol.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcs

See Also

`mountd(1M)`, `nfsd(1M)`, `share_nfs(1M)`, `share_smb(1M)`, `shareall(1M)`, `unshare(1M)`, `zfs(1M)`, `zfs(1M)attributes(5)`

Notes

If `share` commands are invoked multiple times on the same file system, the last `share` invocation supersedes the previous invocation. The options set by the last `share` command replace the old options. For example, if read-write permission was granted to `usera` on the legacy `/somefs` file system, then you want to grant read-write permission also to `userb` on `/somefs`, use the following syntax:

```
example% share -F nfs -o rw=usera:userb /somefs
```

Name shareall, unshareall – publish or unpublish multiple shares

Synopsis

```
shareall [-F FSType [,FSType]...] [-| file]

unshareall [-F FSType [,FSType]...]
```

Description When used with no arguments, `shareall` publishes shares from a specified file that contains a list of share command lines. If the operand is a hyphen (`-`), then the `share` command lines are obtained from standard input. Otherwise, if neither a file nor a hyphen is specified, then shares are published from the SMF shares repository.

Shares can be specified by identifying the file system type in a comma-separated list as an argument to `-F`.

The `unshareall` command unpublishes all published file system shares. Without an `-F` flag, it unpublishes all distributed file system shares.

Options `-F` *FSType* Specify file system type. Defaults to the first entry in `/etc/dfs/fstypes`.

Files `/etc/dfs/dfstab`

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcs

See Also `share(1M)`, `unshare(1M)`, `attributes(5)`

Name sharectl – configure and manage file sharing service

Synopsis

```
sharectl [-h]

sharectl status [-h] [protocol]

sharectl get [-h] [-p property]... protocol

sharectl set [-h] [-p property=value]... protocol
```

Description The `sharectl` command operates on file-sharing protocols, such as NFS and SMB. The command sets the client and server operational properties, takes and restores configuration snapshots, and gets status of the protocol service.

The `get` and `set` subcommands (see below) require root privileges. An authorized user can use `sharectl` to set global values for NFS and SMB properties in the Solaris server management facility. See `nfs(4)` and `smb(4)`.

Interaction with Location Profiles

The `nfsmapid_domain` property is managed in Location profiles (refer to `netcfg(1M)`) for more information about location profiles). These profiles are either fixed, meaning the network configuration is being managed in the traditional way, or reactive, meaning the network configuration is being managed automatically, reacting to changes in the network environment according to policy rules specified in the profiles.

When a fixed location (there can currently be only one, the `DefaultFixed` location) is active, changes made to the SMF repository, including those made by `sharectl`, will be applied to the location when it is disabled, and thus will be restored if that location is later re-enabled.

When a reactive location is active, changes should not be applied directly to the SMF repository; these changes will not be preserved in the location profile, and will thus be lost if the location is disabled, or if the system's network configuration, as managed by `svc:/network/physical:default` and `svc:/network/location:default`, is refreshed or restarted. Changes should instead be applied to the location itself, using the `netcfg(1M)` command; this will save the change to the location profile repository, and will also apply it to the SMF repository (if the change is made to the currently active location).

The `nfsmapid_domain` setting is stored in the `nfsv4-domain` property of a location profile.

Options The following options are supported:

`-h`
Displays usage message.

`-p` *property*[=*value*]
Specifies a property. See "Subcommands," below.

Subcommands `sharectl` supports the subcommands described below. The form of a `sharectl` command is:

```
# sharectl subcommand [option]
```

`get` [`-p` *property*] *protocol*	Get the property values for the specified protocol. If no `-p` option is provided, get all the properties for the specified protocol.
`set` [`-p` *property=value*]... *protocol*	Set properties for the specified file sharing protocol.
`status` [*protocol*]	Display status of the specified protocol, or, if no protocol is specified, of all file-sharing protocols.

Examples

EXAMPLE 1 Getting Properties

The following command gets the properties for the NFS protocol.

```
% sharectl get nfs
servers=1024
lockd_listen_backlog=32
lockd_servers=1024
lockd_retransmit_timeout=5
grace_period=90
server_versmin=2
server_versmax=4
client_versmin=2
client_versmax=4
server_delegation=on
nfsmapid_domain=oracle.com
max_connections=-1
protocol=ALL
listen_backlog=32
device=
```

The following command gets the value of the grace_period property for the NFS protocol.

```
% sharectl get -p grace_period nfs
grace_period=90
```

EXAMPLE 2 Setting a Property

Note in the preceding example that the minimum version of the server NFS protocol (`server_versmin`) is set to 2. The following command sets the minimum version number to version 3.

```
% sharectl set -p server_versmin=3 nfs
```

EXAMPLE 3 Obtaining Status

The following command obtains the status of all file-sharing protocols on a system.

```
% sharectl status
nfs      enabled
```

EXAMPLE 4 Setting Property for SMB Server

The following command sets the value of the `server_signing_required` property for the SMB protocol.

```
% sharectl set -p server_signing_required=true smb
```

EXAMPLE 5 Setting Property for SMB Client

The following command sets the value of the `client_signing_required` property for the SMB protocol.

```
% sharectl set -p client_signing_required=true smb
```

EXAMPLE 6 Setting Tracing of RPC Calls for autofs

The following command expands each RPC call to `autofs` and logs it to the location specified for that service in `automountd(1M)`.

```
# sharectl set trace=1 autofs
```

Exit Status

`0` Successful completion.

non-zero Command failed.

Files

`/usr/include/libshare.h` Error codes used for exit status.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os
Interface Stability	Committed

See Also

`automount(1M)`, `automountd(1M)`, `lockd(1M)`, `mountd(1M)`, `netcfg(1M)`, `nfsd(1M)`, `nfsmapid(1M)`, `nfs(4)`, `smb(4)`, `user_attr(4)`, `attributes(5)`, `rbac(5)`, `smf(5)`, `standards(5)`

Name share_nfs – make NFS shares available for mounting by remote systems

Synopsis

```
share -F nfs [-a [-o specific_options] [-d description]
     pathname [sharename] | [-A]]

zfs set share.nfs=on | off filesystem|share

zfs share -o share.nfs=on | off specific_options
     filesystem|filesystem%share
```

Description The `share` utility defines and publishes a NFS share, which makes a local file system available for mounting by remote systems. It starts the `nfsd`(1M) and `mountd`(1M) daemons if they are not already running.

You can use the `share` command to create and publish a ZFS file system share, but this is considered a legacy operation. See `zfs`(1M) for information about setting the `share.nfs` property or using the `zfs share` command to create and publish NFS shares.

Options The following options are supported:

`-F nfs`
Specify the NFS file sharing protocol.

`-a`
Publish all defined shares.

`-o` *specific_options*
Specify *specific_options* in a comma-separated list of keywords and attribute-value-assertions for interpretation by the NFS protocol. By default, a share is published with read-write access to all clients, unless a specific option overrides the default access. *specific_options* can be any combination of the following:

`aclok`
Allows the NFS server to do access control for NFS Version 2 clients. When `aclok` is set on the server, maximal access is given to all clients. For example, with `aclok` set, if anyone has read permissions, then everyone does. If `aclok` is not set, minimal access is given to all clients.

`anon=`*uid*
Set *uid* to be the effective user ID of unknown users. By default, unknown users are given the effective user ID `UID_NOBODY`. If *uid* is set to −1, access is denied.

charset
All clients will be assumed to be using the specified character set (see list in following description) and file and path names will be converted to UTF-8 for the server.

charset=access_list
Where *charset* is one of: `euc-cn`, `euc-jp`, `euc-jpms`, `euc-kr`, `euc-tw`, `iso8859-1`, `iso8859-2`, `iso8859-5`, `iso8859-6`, `iso8859-7`, `iso8859-8`, `iso8859-9`, `iso8859-13`, `iso8859-15`, `koi8-r`.

Clients that match the *access_list* for one of these properties will be assumed to be using that character set and file and path names will be converted to UTF-8 for the server.

`index=file`
Load `file` rather than a listing of the directory containing this file when the directory is referenced by an NFS URL.

`log[=`*tag*`]`
Enables NFS server logging for the specified file system. The optional tag determines the location of the related log files. The *tag* is defined in `/etc/nfs/nfslog.conf`. If no *tag* is specified, the default values associated with the *global tag* in `/etc/nfs/nfslog.conf` is used. Support of NFS server logging is only available for NFS Version 2 and Version 3 requests.

`noaclfab`
Allows NFS servers to not return fabricated ACLs to NFS clients. The default behavior for NFS servers is to fabricate ACLs. If `noaclfab` is set, then the NFS server does not fabricate ACLs, which is the appropriate choice if the underlying filesystem does not support the POSIX Draft ACL.

`none`
Access is disallowed to all clients. The `ro` or `rw` options can override `none`.

`none=`*access_list*
Access is not allowed to any client that matches the access list. The exception is when the access list is an asterisk (`*`), in which case `ro` or `rw` can override `none`.

`nosub`
Prevents clients from mounting subdirectories of shared directories. For example, if `/export` is shared with the `nosub` option on server *fooey* then a NFS client cannot do:

```
mount -F nfs fooey:/export/home/mnt
```

NFS Version 4 does not use the `MOUNT` protocol. The `nosub` option only applies to NFS Version 2 and Version 3 requests.

`nosuid`
By default, clients are allowed to create files on the shared file system with the setuid or setgid mode enabled. Specifying `nosuid` causes the server file system to silently ignore any attempt to enable the setuid or setgid mode bits.

`public`
Moves the location of the public file handle from `root` (`/`) to the exported directory for WebNFS-enabled browsers and clients. This option does not enable WebNFS service. WebNFS is always on. Only one file system per server may use this option. Any other option, including the `-ro=list` and `-rw=list` options can be included with the `public` option.

`ro`
Share is published with read-only access to all clients.

`ro=`*access_list*
Share is published with read-only access to the clients listed in *access_list*; overrides the `rw` suboption for the clients specified. See *access_list* below.

`root`
Root users from all hosts have root access.

`root=`*access_list*
Only root users from the hosts specified in *access_list* have root access. See *access_list* below. By default, no host has root access, so root users are mapped to an anonymous user ID (see the `anon=`*uid* option described above). Netgroups can be used if the file system shared is using UNIX authentication (`AUTH_SYS`).

`root_mapping=`*uid*
For a client that is allowed root access, map the root UID to the specified user id.

`rw`
Share is published with read and write access to all clients.

`rw=`*access_list*
Share is published with read and write access to the clients listed in *access_list*; overrides the `ro` suboption for the clients specified. See *access_list* below.

`sec=`*mode*[`:`*mode*]. . .
Publishes a share by using one or more of the specified security modes. The *mode* in the `sec=`*mode* option must be a node name supported on the client. If the `sec=` option is not specified, the default security mode used is `AUTH_SYS`. Multiple `sec=` options can be specified on the command line, although each mode can appear only once. The security modes are defined in `nfssec(5)`.

Each `sec=` option specifies modes that apply to any subsequent `window=, rw, ro, rw=, ro=` and `root=` options that are provided before another `sec=`option. Each additional `sec=` resets the security mode context, so that more `window=, rw, ro, rw=, ro=` and `root=` options can be supplied for additional modes.

`sec=`*none*
If the option `sec=`*none* is specified when the client uses `AUTH_NONE`, or if the client uses a security mode that is not one that the file system is shared with, then the credential of each NFS request is treated as unauthenticated. See the `anon=`*uid* option for a description of how unauthenticated requests are handled.

`secure`
This option has been deprecated in favor of the `sec=`*dh* option.

`window=`*value*
When a share is published with `sec=`*dh*, set the maximum life time (in seconds) of the RPC request's credential (in the authentication header) that the NFS server allows. If a credential arrives with a life time larger than what is allowed, the NFS server rejects the request. The default value is 30000 seconds (8.3 hours).

`-d` *description*
Provide a comment that describes the file system to be shared.

`-A`
Display all defined shares.

access_list The *access_list* argument is either the string "*" to represent all hosts or a colon-separated list whose components may be any number of the following:

hostname
The name of a host. With a server configured for DNS or LDAP naming in the `nsswitch hosts` entry, any hostname must be represented as a fully qualified DNS or LDAP name. The hostname specified must be the canonical name for this host and must match the hostname returned on the reverse lookup of the incoming IP address of the NFS client.

netgroup
A netgroup contains a number of hostnames. With a server configured for DNS or LDAP naming in the `nsswitch hosts` entry, any hostname in a netgroup must be represented as a fully qualified DNS or LDAP name.

domain name suffix
To use domain membership the server must use DNS or LDAP to resolve hostnames to IP addresses; that is, the `hosts` entry in the `/etc/nsswitch.conf` must specify `dns` or `ldap` ahead of `nis`, since only DNS and LDAP return the full domain name of the host. Other name services like NIS cannot be used to resolve hostnames on the server because when mapping an IP address to a hostname they do not return domain information. For example,

```
NIS   172.16.45.9 --> "myhost"
```

and:

```
DNS or LDAP   172.16.45.9 -->
     "myhost.mydomain.mycompany.com"
```

The domain name suffix is distinguished from hostnames and netgroups by a prefixed dot. For example,

```
rw=.mydomain.mycompany.com
```

A single dot can be used to match a hostname with no suffix. For example,

```
rw=.
```

matches `mydomain` but not `mydomain.mycompany.com`. This feature can be used to match hosts resolved through NIS rather than DNS and LDAP.

network
The network or subnet component is preceded by an at-sign (`@`). It can be either a name or a dotted address. If a name, it is converted to a dotted address by `getnetbyname(3SOCKET)`. For example,

`=@mynet`

would be equivalent to:

`=@172.16` or `=@172.16.0.0`

The network prefix assumes an octet-aligned netmask determined from the zeroth octet in the low-order part of the address up to and including the high-order octet, if you want to specify a single IP address (see below). In the case where network prefixes are not byte-aligned, the syntax allows a mask length to be specified explicitly following a slash (`/`) delimiter. For example,

`=@theothernet/17` or `=@172.16.132/22`

...where the mask is the number of leftmost contiguous significant bits in the corresponding IP address.

When specifying individual IP addresses, use the same `@` notation described above, without a netmask specification. For example:

`=@172.16.132.14`

Multiple, individual IP addresses would be specified, for example, as:

`root=@172.16.132.20:@172.16.134.20`

A prefixed minus sign (–) denies access to that component of *access_list*. The list is searched sequentially until a match is found that either grants or denies access, or until the end of the list is reached. For example, if host `terra` is in the `engineering` netgroup, then

`rw=-terra:engineering`

denies access to `terra` but

`rw=engineering:-terra`

grants access to `terra`.

Operands

The following operands are supported:

pathname
The pathname of the file system to be shared.

Examples

EXAMPLE 1 Define and Publish an NFS Share

The following example shows how to use the legacy `share` command to define and publish the `/export/manuals` file system share.

```
# share -F NFS /export/manuals
```

The following example shows how to use the `zfs set` command to share a ZFS file system.

```
# zfs set share.nfs=on tank/data
```

EXAMPLE 1 Define and Publish an NFS Share *(Continued)*

The following example shows how to create a named NFS share, `tank/public%pubshare`, with the `share.nfs.public` option rather than setting this option on the ZFS file system, `tank/public`, because this property is not inheritable.

```
# zfs create -o mountpoint=/pub tank/public
# zfs share -o share.nfs=on -o share.nfs.public=on tank/public%pubshare
```

Exit Status The following exit values are returned:

`0`
Successful completion.

`>0`
An error occurred.

Files `/etc/dfs/fstypes`
list of system types, NFS by default

`/etc/dfs/sharetab`
system record of shared file systems

`/etc/nfs/nfslogtab`
system record of logged file systems

`/etc/nfs/nfslog.conf`
logging configuration file

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	service/file-system/nfs

See Also `mount(1M)`, `mountd(1M)`, `nfsd(1M)`, `nfslogd(1M)`, `share(1M)`, `unshare(1M)`, `zfs_share(1M)`, `getnetbyname(3SOCKET)`, `nfslog.conf(4)`, `netgroup(4)`, `attributes(5)`, `nfssec(5)`

Notes Creating and publishing an NFS share with the `share` command is permanent until the share is unshared. Publishing NFS shares is managed by the following SMF service:

```
$ svcs | grep share
online         Mar_07   svc:/network/shares:default
```

If the file system being shared is a symbolic link to a valid pathname, the canonical path (the path which the symbolic link follows) are shared. For example, if `/export/foo` is a symbolic link to `/export/bar` (`/export/foo -> /export/bar`), the following `share` command results in `/export/bar` as the shared pathname (and not `/export/foo`).

```
# share -F nfs /export/foo
```

An NFS mount of `server:/export/foo` results in `server:/export/bar` really being mounted.

The `mountd`(1M) process allows the processing of a path name the contains a symbolic link. This allows the processing of paths that are not themselves explicitly shared with `share_nfs`. For example, `/export/foo` might be a symbolic link that refers to `/export/bar` which has been specifically shared. When the client mounts `/export/foo` the `mountd` processing follows the symbolic link and responds with the `/export/bar`. The NFS Version 4 protocol does not use the `mountd` processing and the client's use of `/export/foo` does not work as it does with NFS Version 2 and Version 3 and the client receives an error when attempting to mount `/export/foo`.

Name share_smb – make SMB shares available for mounting by remote systems

Synopsis

```
share  -F smb [-a [-o specific-options] [-d description]
     pathname sharename  | [-A]]

zfs set share.smb=on | off filesystem|filesystem%share

zfs share -o share.smb=on | off specific_options
     filesystem|filesystem%share
```

Description The `share` command defines and publishes a SMB share, which makes a local file system available for mounting by remote systems.

You can modify the behavior of SMB shares by setting property values with either the `share`, the `zfs set` command or the `zfs share` command. See the `share(1M)` and `zfs(1M)` man pages.

The `share` command has the following options:

`-F smb`
Share SMB file sharing protocol.

`-a`
Publish all defined shares.

`-o` *specific-options*
Specify *specific-options* in a comma-separated list of keywords and attribute-value-assertions for interpretation by the SMB protocol. By default, a share is published with read-write access to all clients, unless a specific option overrides the default access. *specific-options* can be any combination of the properties supported by a given file system.

`-d` *description*
Provide a comment that describes the file system to be shared.

`-A`
Display all defined shares.

Share Properties The following SMB share properties are supported and can be set by the `zfs` and `share` commands:

`abe=`*boolean*
Sets the access-based enumeration (ABE) policy for a share. When set to `true`, ABE filtering is enabled on this share and directory entries to which the requesting user has no access will be omitted from directory listings returned to the client. When set to `false` or not defined, ABE filtering will not be performed on this share. This property is not defined by default.

`false`
Disable ABE for this share.

`true`
Enable ABE for this share.

`ad-container`
Specifies the AD container in which to publish shares.

The AD container is specified as a comma-separated list of attribute name-value pairs using the LDAP distinguished name (DN) or relative distinguished name (RDN) format.

The following example uses the `share` command to specify the AD container:

```
$ share -F smb -o abe=true,ad-container=cn=sales,ou=mycompany,dc=com /export/home
```

The following example uses the `zfs sare` command to specify the AD container:

```
$ zfs share -o share.smb=on -o share.smb.ad-container=cn=sales,ou=mycompany,dc=com -o share.s
```

The DN or RDN must be specified in LDAP format using the `cn=`, `ou=`, and `dc=` prefixes:

- `cn` represents the common name
- `ou` represents the organizational unit
- `dc` represents the domain component

`cn=`, `ou=` and `dc=` are attribute types. The attribute type used to describe an object's RDN is called the naming attribute, which, for ADS, includes the following object classes:

- `cn` for the `user` object class
- `ou` for the organizational unit (`OU`) object class
- `dc` for the `domainDns` object class

`catia=`*boolean*
Specifies whether to perform CATIA character substitution. CATIA V4 uses characters in file names that are considered to be invalid by Windows. A CATIA V4 file could be inaccessible to Windows clients if the file name contains any of the characters that are considered illegal in Windows. By default, CATIA character substitution is not performed. See *Managing SMB File Sharing and Windows Interoperability in Oracle Solaris 11.1*.

If the `catia` property is set to `true`, the following character substitution is applied to file names.

```
CATIA    CATIA
V4 UNIX  V5 Windows
  "      \250   0x00a8  Dieresis
  *      \244   0x00a4  Currency Sign
  /      \370   0x00f8  Latin Small Letter O with Stroke
  :      \367   0x00f7  Division Sign
  <      \253   0x00ab  Left-Pointing Double Angle Quotation Mark
  >      \273   0x00bb  Right-Pointing Double Angle Quotation Mark
  ?      \277   0x00bf  Inverted Question Mark
  \      \377   0x00ff  Latin Small Letter Y with Dieresis
  |      \246   0x00a6  Broken Bar
```

`csc=`*value*
Sets the client-side caching policy for a share. Client-side caching is a client feature and offline files are managed entirely by the clients.

The following are valid values for the `csc` property:

- `manual` – Clients are permitted to cache files from the specified share for offline use as requested by users. However, automatic file-by-file reintegration is not permitted. `manual` is the default value.
- `auto` – Clients are permitted to automatically cache files from the specified share for offline use and file-by-file reintegration is permitted.
- `vdo` – Clients are permitted to automatically cache files from the specified share for offline use, file-by-file reintegration is permitted, and clients are permitted to work from their local cache even while offline.
- `disabled` – Client-side caching is not permitted for this share.

`dfsroot=`*boolean*
Marks a share as a distributed file system (DFS) root share to distinguish it from a regular share. By default, `dfsroot` is not defined. If `dfsroot` is `false` or not defined, the share is not a DFS root share.

`guestok=`*boolean*
Sets the guest access policy for the share. When set to `true` guest access is allowed on this share. When set to `false` or not defined guest access is not allowed on this share. This property is not defined by default.

An `idmap(1M)` name-based rule can be used to map `guest` to any local user name, such as `guest` or `nobody`. If the local account has a password in `/var/smb/smbpasswd` the guest connection will be authenticated against that password. Any connection made using an account that maps to the local guest account will be treated as a guest connection.

The following name-based rule maps the Windows `Guest` user to the UNIX `guest` user:

```
# idmap add winname:Guest unixuser:guest
```

`none=`*access-list*
Specifies that access is not allowed to any client that matches the access list. The exception is when the access list is an asterisk (`*`), in which case `ro` or `rw` can override `none`.

`ro=`*access-list*
Specifies that sharing is read-only to the clients listed in *access-list*. Overrides the `rw` suboption for the clients specified. See *access-list*.

`rw=`*access-list*
Specifies that sharing is read-write to the clients listed in *access-list*. Overrides the `ro` suboption for the clients specified. See *access-list*.

Access List Argument

The *access-list* argument is either the string "*" to represent all hosts or a colon-separated list whose components may be any number of the following:

hostname
Specifies the name of a host. *hostname* must be a fully qualified DNS or LDAP name when the host specifies these naming schemes in the `hosts` portion of the `nsswitch.conf` file.

netgroup
A netgroup contains a number of host names. Any *hostname* in a netgroup must be a fully qualified DNS or LDAP name when the host specifies these naming schemes in the `hosts` portion of the `nsswitch.conf` file.

domainname.suffix
To use domain membership, the server must use DNS or LDAP to resolve host names to IP addresses. This means that the `hosts` entry of the `/etc/nsswitch.conf` file must specify `dns` or `ldap` before `nis`. You must do this because only DNS and LDAP return the full domain name of the host.

Other naming services, such as NIS, cannot be used to resolve host names on the server because these naming services do not return domain information. For example, the following shows how NIS, DNS, and LDAP return host name information for the `172.16.45.9` IP address:

NIS	Returns: `myhost`
DNS or LDAP	Returns: `myhost.mydomain.mycompany.com`

The domain name suffix is distinguished from host names and netgroups by a prefixed dot. For example, `rw=.mydomain.mycompany.com` matches all host names in `mydomain.mycompany.com`.

The `rw=.` notation uses a single dot to match a host name that has no suffix. This notation matches `mydomain` but not `mydomain.mycompany.com`. This feature can be used to match hosts that are resolved by NIS rather than by DNS and LDAP.

network
The network or subnet component is preceded by an at-sign character (`@`). It can be either a network name or a dotted address.

A network name is converted to a dotted address by using `getnetbyname(3SOCKET)`. For example, `=@mynet` is equivalent to `=@172.16` or `=@172.16.0.0`.

The network prefix assumes an octet-aligned netmask. The netmask is determined from the zeroth octet in the low-order part of the address up to and including the high-order octet. If network prefixes are not byte-aligned, the syntax permits a mask length to be explicitly specified following a slash delimiter (`/`). For example, `=@theothernet/17` or `=@172.16.132/22` where the mask is the number of leftmost contiguous significant bits in the corresponding IP address.

When specifying individual IP addresses, use the same `@` notation described previously, but do not use a netmask specification. For example, `=@172.16.132.14`.

You can use a colon character (`:`) to separate multiple, individual IP addresses. For example, `root=@172.16.132.20:@172.16.134.20`.

A prefixed minus sign (–) denies access to that component of *access-list*. The list is searched sequentially until a match is found that either grants or denies access, or until the end of the list is reached. For example, if host `terra` is in the `engineering` netgroup, specifying `rw=-terra:engineering` denies access to `terra`. However, specifying `rw=engineering:-terra` grants access to `terra`.

Examples

EXAMPLE 1 Setting a Share Property

The following examples use the `zfs share` and `share` commands to create and publish an SMB share.

- The following example shows how to use the `zfs share` command to create and publish an SMB share that also enables guest access:

```
# zfs share -o share.smb=on -o share.smb.guestok=on tank/home%hshare
```

- The following example shows how to use the `share` command to enable guest access on a share:

```
# share -F smb -o guestok=true /tank/home
```

EXAMPLE 2 Viewing the Share Properties

The following examples show how to use the `zfs get` command and the `/etc/dfs/sharetab` file to view share properties:

- The `zfs get` command enables you to view share properties on the `tank/home` dataset:

```
# zfs get share.smb tank/home%hshare
NAME              PROPERTY   VALUE  SOURCE
tank/home%hshare  share.smb  on     local
```

- The `/etc/dfs/sharetab` file shows all the active shares on the system. The entry for each share shows the properties set and their values:

```
# grep home /etc/dfs/sharetab
/tank/home      hshare  smb     guestok
```

Files

`/etc/dfs/sharetab`
System record of shared file systems

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	service/file-system/smb
Interface Stability	Committed

See Also idmap(1M), share(1M), zfs(1M), zfs(1M), getnetbyname(3SOCKET), netgroup(4), attributes(5)

Name showmount – show remote mounts

Synopsis

```
/usr/sbin/showmount [-ade] [hostname]
```

Description `showmount` lists the clients that have remotely mounted a filesystem from *host*. This information is maintained by the `mountd(1M)` server on *host*, and is saved across crashes in the file `/etc/rmtab`. The default value for *host* is the value returned by `hostname(1)`.

The `showmount` command does not display the names of NFS Version 4 clients.

Options

`-a` Print all remote mounts in the format:

```
hostname : directory
```

where `hostname` is the name of the client, and `directory` is the root of the file system that has been mounted.

`-d` List directories that have been remotely mounted by clients.

`-e` Print the list of shared file systems.

Files `/etc/rmtab`

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/file-system/nfs

See Also `hostname(1)`, `mountd(1M)`, `attributes(5)`

Installing Oracle Solaris 11.1 Systems

Bugs If a client crashes, its entry is removed from the list of remote mounts on the server.

Name shutdown – shut down system, change system state

Synopsis

```
/usr/sbin/shutdown [-y] [-g grace-period] [-r | -i init-state]
     [message]
```

Description `shutdown` is executed by the super user to change the state of the machine. In most cases, it is used to change from the multi-user state (state 2) to another state.

By default, `shutdown` brings the system to a state where only the console has access to the operating system. This state is called single-user.

Before starting to shut down daemons and killing processes, `shutdown` sends a warning message and, by default, a final message asking for confirmation. *message* is a string that is sent out following the standard warning message:

```
The system will be shut down in . . .
```

If the string contains more than one word, it should be contained within single (') or double (") quotation marks.

The warning message and the user provided *message* are output when there are 7200, 3600, 1800, 1200, 600, 300, 120, 60, and 30 seconds remaining before `shutdown` begins. See `EXAMPLES`.

System state definitions are:

state 0 — Stop the operating system.

state 1 — State 1 is referred to as the administrative state. In state 1 file systems required for multi-user operations are mounted, and logins requiring access to multi-user file systems can be used. When the system comes up from firmware mode into state 1, only the console is active and other multi-user (state 2) services are unavailable. Note that not all user processes are stopped when transitioning from multi-user state to state 1.

state s, S — State s (or S) is referred to as the single-user state. All user processes are stopped on transitions to this state. In the single-user state, file systems required for multi-user logins are unmounted and the system can only be accessed through the console. Logins requiring access to multi-user file systems cannot be used.

state 5 — Shut the machine down so that it is safe to remove the power. Have the machine remove power, if possible. The `rc0` procedure is called to perform this task.

state 6 — Stop the operating system and reboot to the state defined by the `initdefault` entry in `/etc/inittab`. The `rc6` procedure is called to perform this task.

Options `-y`

Pre-answer the confirmation question so the command can be run without user intervention.

`-g` *grace-period*
Allow the super user to change the number of seconds from the 60-second default.

`-i` *init-state*
If there are warnings, *init-state* specifies the state `init` is to be in. By default, system state 's' is used.

`-r`
Equivalent to specifying `-i6`.

Examples

EXAMPLE 1 Using `shutdown`

In the following example, `shutdown` is being executed on host `foo` and is scheduled in 120 seconds. The warning message is output 2 minutes, 1 minute, and 30 seconds before the final confirmation message.

```
example# shutdown -i S -g 120 "===== disk replacement ====="
Shutdown started.   Tue Jun   7  14:51:40 PDT  1994

Broadcast Message from root (pts/1) on foo Tue Jun  7 14:51:41. . .
The system will be shut down in 2 minutes
===== disk replacement =====
Broadcast Message from root (pts/1) on foo Tue Jun  7 14:52:41. . .
The system will be shut down in 1 minutes
===== disk replacement =====
Broadcast Message from root (pts/1) on foo Tue Jun  7 14:53:41. . .
The system will be shut down in 30 seconds
===== disk replacement =====
Do you want to continue? (y or n):
```

Files

`/etc/inittab` controls process dispatching by `init`

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also

`boot(1M)`, `halt(1M)`, `init(1M)`, `killall(1M)`, `reboot(1M)`, `ufsdump(1M)`, `init.d(4)`, `inittab(4)`, `nologin(4)`, `attributes(5)`

Notes

When a system transitions down to the `S` or `s` state, the `/etc/nologin` file (see `nologin(4)`) is created. Upon subsequent transition to state 2 (multi-user state), this file is removed by a script in the `/etc/rc2.d` directory.

Name slpd – Service Location Protocol Daemon

Synopsis `/usr/lib/inet/slpd [-f` *configuration-file*`]`

Description The `slpd` daemon provides common server functionality for the Service Location Protocol ("SLP") versions 1 and 2, as defined by IETF in *RFC 2165* and *RFC 2608*. SLP provides a scalable framework for the discovery and selection of network services.

`slpd` provides the following framework services:

Directory Agent	This service automatically caches service advertisements from service agents to provide them to user agents, and makes directory agent advertisements of its services. This service is optional. `slpd` does not provide directory agent service by default. Directory agents are not databases, and they do not need to be maintained.
Service Agent Server	All service agents on the local host register and deregister with this server. This service responds to all requests for services, and forwards registrations to directory agents. By default, `slpd` is a service agent server.
Passive Directory Agent Discovery	This service listens for directory agent advertisements and maintains a table of active directory agents. When a user agent wishes to discover a directory agent, it can simply query `slpd`, obviating the need to perform discovery by means of multicast. By default, `slpd` performs this service.
Proxy Registration	This service can act as a proxy service agent for services that cannot register themselves. `slpd` reads the proxy registration file for information on services it is to proxy. By default, no services are registered by proxy.

All configuration options are available from the configuration file. `slpd` reads its configuration file upon startup.

Stop and start the slpd daemon using `svcadm(1M)`. Use the command `svcadm enable network/slp` to start the slpd daemon. Use the command `svcadm disable network/slp` to stop it.

The file `/etc/inet/slp.conf` must exist before the `slp` service can start the daemon. Only the example file `/etc/inet/slp.conf.example` is present by default. To enable SLP, copy `/etc/inet/slp.conf.example` to `/etc/inet/slp.conf`.

Options The following options are supported:

`-f` *configuration-file* Specify an alternate configuration file

Examples **EXAMPLE 1** Stopping the `slpd` daemon

The following command stops the `slpd` daemon:

```
example# svcadm disable network/slp
```

EXAMPLE 2 Restarting the `slpd` daemon

The following command restarts the `slpd` daemon:

```
example# svcadm restart network/slp
```

Files `/etc/inet/slp.conf` The default configuration file

`slpd.reg` The proxy registration file

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	service/network/slp, service/network/slp
CSI	Enabled
Interface Stability	Committed

See Also `svcs(1)`, `svcadm(1M)`, `slp_api(3SLP)`, `slp.conf(4)`, `slpd.reg(4)`, `attributes(5)`, `smf(5)`, `slp(7P)`

Oracle Solaris Administration: Network Services

Guttman, E., Perkins, C., Veizades, J., and Day, M., *RFC 2608, Service Location Protocol, Version 2*, The Internet Society, June 1999.

Notes The `slpd` service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/network/slp
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. The service's status can be queried using the `svcs(1)` command.

Name smattrpop – populate security attribute databases in a name service

Synopsis

```
smattrpop [-c ] [-f] [-m] [-p policy] [-r] -s scope -t scope
     [-v] database
```

Description The `smattrpop` command updates the `auth_attr(4)`, `exec_attr(4)`, `prof_attr(4)`, and `user_attr(4)` role-based access control databases in a target NIS, LDAP, or local `/etc` files name service from the corresponding databases in a source name service or files.

This command processes the table entries from the source database and merges each source entry field into the same field in the corresponding table entry in the target database. If a source entry does not exist in the target database, the entry is created. If the source entry exists in the target database, the fields are merged or replaced according to the command options.

Any errors encountered while updating the target entry are reported to `stdout`, and the command continues with the next source database entry.

Options The following options are supported:

`-c` Performs cross-table checking. If you specify this option and a check error occurs, a message identifying the check error is written to `stdout`.

The target entry values are checked against entries in related databases:

- `auths` values — Each value must exist as the name of an authorization in the `auth_attr(4)` database.
- `profiles` values — Each value must exist as a name of a profile in the `prof_attr(4)` database.
- `roles` values — Each value must exist as the name of a role identity in the `user_attr(4)` database.
- For each `exec_attr(4)` entry in the source database, the name must exist as the name of a `profile` in the `prof_attr`(4) database.

`-f` Specifies that the value in each field in the source entry replaces the value in the corresponding field in the target entry, if the source entry field has a non-empty value.

`-m` For the `auths`, `profiles`, and `roles` attributes, specifies that the values in each field in the source entry are merged with the values in the corresponding target entry field. If a source value does not exist in the target field, the value is appended to the set of target values. If the target field is empty, the source values replace the target field. The attribute values that merge depend on the database being updated:

- `prof_attr`(4) — the `auths` and `profiles` attribute values are merged.
- `user_attr`(4) — the `auths`, `profiles`, and `roles` attribute values are merged.
- `exec_attr`(4) — the `uid`, `gid`, `euid`, and `egid` values are merged.

`-p` *policy* Specifies the value of the policy field in the `exec_attr`(4) database. Valid values are `suser` (standard Solaris superuser) and `tsol` (Trusted Solaris). If you specify this option, only the entries in the source `exec_attr` database with the specified policy are processed. If you omit this option, all entries in the source `exec_attr` database are processed.

`-r` Specifies that role identities in the `user_attr`(4) database in the source name service are processed. If you omit this option, only the normal user entries in the `user_attr` source database are processed.

`-s` *scope* Specifies the source name service or local file directory for database updates, using the following syntax:

type`:/`*server*`/`*domain*

where *type* indicates the type of name service. Valid values for *type* are:

- `file` — local files
- `nis` — NIS name service
- `ldap` — LDAP name service

server indicates the local host name of the Solaris system on which the `smattrpop` command is executed, and on which both the source and target databases exist.

domain specifies the management domain name for the name service.

You can use two special cases of *scope* values:

- To indicate the databases in the `/etc/security` local system directory, use the scope `file`:/*server*, where *server* is the name of the local system.
- To load from databases in an arbitrary directory on the Solaris server, use the scope `file`:/*server*/*pathname*, where where *server* is the name of the local system and *pathname* is the fully-qualified directory path name to the database files.

`-t` *scope* Specifies the target name service or local file directory for database updates, using the following syntax:

type`:/`*server*`/`*domain*

where *type* indicates the type of name service. Valid values for *type* are:

- `file` — local files
- `nis` — NIS name service
- `ldap` — LDAP name service

server indicates the local host name of the Solaris system on which the `smattrpop` command is executed, and on which both the source and target databases exist.

domain specifies the management domain name for the name service.

You can use two special cases of *scope* values:

- To indicate the databases in the `/etc/security` local system directory, use the scope `file:`/*server*, where *server* is the name of the local system.
- To update to databases in an arbitrary directory on the Solaris server, use the scope `file:`/*server*/*pathname*, where where *server* is the name of the local system and *pathname* is the fully-qualified directory path name to the database files.

`-v` Specifies that verbose messages are written. A message is written to `stdout` for each entry processed.

Operands The following operands are supported:

database Populates one or all databases. You can specify either the name of the database you want to process (for example, `auth_attr`), or `all` to process all databases. If you specify `all`, the databases are processed in the following order:

1. `auth_attr`(4)
2. `prof_attr`(4)
3. `exec_attr`(4)
4. `user_attr`(4)

Examples **EXAMPLE 1** Populating all tables in the NIS name service

The following example merges the values from all four attribute databases in the `/etc/security` directory of the local system into the corresponding tables in the NIS domain, `east.example.com`. The command is executed on the master server, `hoosier`, for the NIS domain and the source files are in the `/etc` and `/etc/security` directories on the NIS master server. No cross-table checking is performed. A summary message indicating the number of entries processed and updated for each table is written to `stdout`.

```
/usr/sadm/bin/smattrpop -s file:/hoosier \
          -t nis:/hoosier/east.example.com all
```

Environment Variables See `environ(5)` for a description of the `JAVA_HOME` environment variable, which affects the execution of the `smattrpop` command. If this environment variable is not specified, the `/usr/java` location is used.

Exit Status Any errors encountered while updating the target entry are reported to `stdout`. The following exit values are returned:

`0` The specified tables were updated. Individual entries may have encountered checking errors.

`1` A syntax error occurred in the command line.

`2` A fatal error occurred and the tables were not completely processed. Some entries may have been updated before the failure.

Files

`/etc/security/auth_attr`	Authorization description database. See `auth_attr`(4).
`/etc/security/exec_attr`	Execution profiles database. See `exec_attr`(4).
`/etc/security/prof_attr`	Profile description database. See `prof_attr`(4).
`/etc/user_attr`	Extended user attribute database. See `user_attr`(4).

Attributes See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWmga

See Also `auth_attr`(4), `exec_attr`(4), `prof_attr`(4), `user_attr`(4), `attributes`(5), `environ`(5)

Name smbadm – configure and manage SMB local groups and users, manage domain membership, manage persistent password information, and issue various commands

Synopsis

```
smbadm add-key [-u username]

smbadm add-member -m member [[-m member] ...] group

smbadm create-group [-d description] group

smbadm crypt

smbadm delete-group group

smbadm disable-user username

smbadm enable-user username

smbadm get-group [[-p property] ...] group

smbadm join -u username [-o organizational-unit] domain

smbadm join -w workgroup

smbadm lookup-server //server

smbadm lookup-user [-u username] name | SID

smbadm print [-u username] //server/share {print_file|-}

smbadm remove-key [-u username]

smbadm remove-member -m member [[-m member] ...] group

smbadm rename-group group new-group

smbadm set-group -p property=value [[-p property=value] ...] group

smbadm show-connections [-t] [-u username] [-c computername | -s sharename]
    server

smbadm show-domains

smbadm show-files [-t] [-u username] server

smbadm show-groups [-m] [-p] [group]

smbadm show-sessions [-t] [-u username] server

smbadm show-shares [-t] [-A | -u username] server
```

Description The `smbadm` command is used to configure SMB local groups, to manage domain membership, to manage persistent password information, and issue various commands. You can also use the `smbadm` command to enable or disable SMB password generation for individual local users.

SMB local groups can be used when Windows accounts must be members of some local groups and when Windows-style privileges must be granted. Solaris local groups cannot provide these functions.

There are two types of local groups: user defined and built-in. Built-in local groups are predefined local groups to support common administration tasks.

In order to provide proper identity mapping between SMB local groups and Solaris groups, an SMB local group must have a corresponding Solaris group. This requirement has two consequences: first, the group name must conform to the intersection of the Windows and Solaris group name rules. Thus, an SMB local group name can be up to eight (8) characters long and contain only lowercase characters and numbers. Second, a Solaris local group has to be created before an SMB local group can be created.

Built-in groups are standard Windows groups and are predefined by the SMB service. The built-in groups cannot be added, removed, or renamed, and these groups do not follow the SMB local group naming conventions.

When the SMB server is started, the following built-in groups are available:

Administrators
: Group members can administer the system.

Backup Operators
: Group members can bypass file access controls to back up and restore files.

Power Users
: Group members can share directories.

Solaris local users must have an SMB password for authentication and to gain access to SMB resources. This password is created by using the `passwd(1)` command when the `pam_smb_password` module is added to the system's PAM configuration. See the `pam_smb_passwd(5)` man page.

The `disable-user` and `enable-user` subcommands control SMB password-generation for a specified local user. When disabled, the user is prevented from connecting to the Solaris SMB service. By default, SMB password-generation is enabled for all local users.

To reenable a disabled user, you must use the `enable-user` subcommand and then reset the user's password by using the `passwd` command. The `pam_smb_passwd.so.1` module must be added to the system's PAM configuration to generate an SMB password.

Escaping Backslash Character

For the `add-member`, `remove-member`, and `join` (with `-u`) subcommands, the backslash character (`\`) is a valid separator between member or user names and domain names. The backslash character is a shell special character and must be quoted. For example, you might escape the backslash character with another backslash character: *domain*`\\`*username*. For more information about handling shell special characters, see the man page for your shell.

Operands

The `smbadm` command uses the following operands:

domain
: Specifies the name of an existing Windows domain to join.

group
Specifies the name of the SMB local group.

username
Specifies the name of a Windows user. *username* can be specified in any of the following formats:

*domain**username*[+*password*]
domain/*username*[+*password*]
username@*domain*
username

...where *domain* can be the NetBIOS or DNS domain name.

server
Specifies the name or IP address of the local host.

Sub-commands The `smbadm` command includes these subcommands:

`add-key` [`-u` *username*]
Specifies persistent password information to be used for an SMB server user account. When you specify this information, mounts can be done without a password prompt in non-Kerberos configurations. Kerberos sites should use Kerberos automatically, not prompt for a password. If a default domain is available in SMF, the domain can be omitted. If a user name is not specified, the Solaris user account name is used. An encrypted (hashed) password can also be used, see the `crypt` subcommand. The command can also read a password from standard input, prompting if standard input is a TTY.

Passwords can also be stored for a specific server by using a server name in place of the domain name.

The persistent password information will also be stored in `/var/smb/smbfspasswd` for the user running the command.

`add-member -m` *member* [[`-m` *member*] ...] *group*
Adds the specified member to the specified SMB local group. The `-m` *member* option specifies the name of an SMB local group member. The member name must include an existing user name and an optional domain name.

Specify the member name in either of the following formats:

[*domain*\]*username*
[*domain*/]*username*

For example, a valid member name might be `sales\terry` or `sales/terry`, where `sales` is the Windows domain name and `terry` is the name of a user in the `sales` domain.

`create-group` [`-d` *description*] *group*
Creates an SMB local group with the specified name. You can optionally specify a description of the group by using the `-d` option.

`crypt`
Creates a hash of a password. This subcommand prompts for a password and writes the hash to standard output. This hash value is suitable for use as a value for the encrypted password option for `smbfs` mount and various `smbadm` subcommands.

`delete-group` *group*
Deletes the specified SMB local group. The built-in groups cannot be deleted.

`disable-user` *username*
Disables SMB password-generation capabilities for the specified local user. A disabled local user is prevented from accessing the system by means of the SMB service. When a local user account is disabled, you cannot use the `passwd` command to modify the user's SMB password until the user account is reenabled.

`enable-user` *username*
Enables SMB password-generation capabilities for the specified local user. After the password-generation capabilities are reenabled, you must use the `passwd` command to generate the SMB password for the local user before he can connect to the SMB service.

The `passwd` command manages both the Solaris password and SMB password for this user if the `pam_smb_passwd` module has been added to the system's PAM configuration.

`get-group` [[`-p` *property=value*] …] *group*
Retrieves property values for the specified group. If no property is specified, all property values are shown.

`join -u` *username* [`-o` *organizational-unit*] *domain*
Joins a Windows domain or a workgroup.

The default mode for the SMB service is workgroup mode, which uses the default workgroup name, `WORKGROUP`.

An authenticated user account is required to join a domain, so you must specify a Windows user name with the `-u` option. If the password is not specified on the command line, the user is prompted for it. The following users are allowed to perform domain join:

- Domain administrator. Can join any number of systems to the domain with machine trust accounts placed in any containers.
- Delegated administrator with authority over one or more OUs. Can join any number of systems to a domain with machine account location designated in the OUs they are responsible for.
- Normal user with machine accounts pre-staged by administrator. Can join a system to the domain as pre-authorized by an administrator.
- Normal user. Normally authorized to join a limited number of systems. For more details see the Active Directory documentation and consult the Active Directory domain administrator.

username and *domain* can be entered in any of the following formats:

username[+*password*] *domain*
*domain**username*[+*password*]
domain/*username*[+*password*]
username@*domain*

...where *domain* can be the NetBIOS or DNS domain name.

By default, a machine trust account for the system will be automatically created in the default container for computer accounts (`cn=Computers`) as part of the domain join operation if the account does not already exist in Active Directory.

The `-o` *organizational-unit* option specifies an alternative organizational unit in which the system's machine trust account will be created.

If the system's computer account already exists, you do not need to specify the `-o` option. A warning will be given if the OU specified is not the one that the account is in.

The *organizational-unit* is specified as a comma-separated list of one or more name-value pairs using the domain-relative distinguished name (DN) format, for example, `'ou=innerOU,ou=outerOU'`.

The following reserved characters when specified in an attribute value must be escaped using the backslash character (\). The backslash character is a shell special character and so distinguished names that contain the following reserved characters must be quoted. It is recommended to use single quotes as opposed to double quotes because backslash enclosed by double quotes can retain its special meaning in some cases. For more information about handling shell special characters, see the man page for your shell.

```
---------+---------------------
Reserved |
Character|    Description
---------+---------------------
 ,            comma
 +            plus sign
 "            double quote
 \            backslash
 <            left angle bracket
 >            right angle bracket
 ;            semicolon
 =            equals sign
 #            # character at the beginning of a string
```

For example, in the following hierarchy:

```
dc=com
            dc=mycompany
               ou=Departments
                  ou=Engineering
                  ou=Payables,Receivables,and Payroll
```

:
:

If the machine trust account is intended to be created in the sub-OU named `engineering`, the organizational-unit should be specified as:

```
ou=Engineering,ou=Departments
```

If the machine trust account is intended to be created in the second sub-OU, the *organizational-unit* should be specified with backslashes and enclosed in single quotes as follows:

```
ou=Payables\,Receivables\,and Payroll,ou=Departments'
```

`join -w` *workgroup*
Joins a Windows domain or a workgroup.

The `-w` *workgroup* option specifies the name of the workgroup to join when using the `join` subcommand.

`lookup-server` //*server*
Resolves the specified server to IP address, NetBIOS domain, and NetBIOS server name. *server* can be one of the following:

- NetBIOS hostname
- DNS hostname
- IP address

`lookup-user` [`-u` *username*] *name* | *SID*
Resolves information about the name or SID of an account in the current domain or any trusted domain.

`print` [`-u` *username*] //*server*/*share* {*print_file*|-}
Print file to the specified remote printer. If *print_file* is a hyphen (`-`), read standard input. If a default domain is available in SMF, the domain can be omitted. If a user name is not specified, the Solaris user account name is used. An encrypted (hashed) password can also be used, see `crypt` subcommand. The command can also take a password through redirection.

`remove-key` [`-u` *username*]
Erases the passwords for the user running the command. The passwords in `/var/smb/smbfspasswd` will also be deleted for the user running the command.

The username and domain name portions of the name are optional. If a default domain is available in SMF, the domain can be omitted. If a *username* is not specified, all of the keys that are stored for the user who is running the command will be deleted.

If the user's password is stored for a specific server, the server name should be specified in place of the domain name.

`remove-member -m` *member* [[`-m` *member*] …] *group*
Removes the specified member from the specified SMB local group. The `-m` *member* option specifies the name of an SMB local group member. The member name must include an existing user name and an optional domain name.

Specify the member name in either of the following formats:

[*domain*`\`]*username*
[*domain*`/`]*username*

For example, a valid member name might be `sales\terry` or `sales/terry`, where `sales` is the Windows domain name and `terry` is the name of a user in the `sales` domain.

`rename-group` *group new-group*
Renames the specified SMB local group. The group must already exist. The built-in groups cannot be renamed.

`set-group -p` *property=value* [[`-p` *property=value*] …] *group*
Sets configuration properties for an SMB local group. The description and the privileges for the built-in groups cannot be changed.

The `-p` *property=value* option specifies the list of properties to be set on the specified group.

The group-related properties are as follows:

`backup=[on|off]`
Specifies whether members of the SMB local group can bypass file access controls to back up file system objects.

`description=`*description-text*
Specifies a text description for the SMB local group.

`restore=[on|off]`
Specifies whether members of the SMB local group can bypass file access controls to restore file system objects.

`take-ownership=[on|off]`
Specifies whether members of the SMB local group can take ownership of file system objects.

`show-domains`
Shows information about the current workgroup or domain. The information typically includes the workgroup name or the primary domain name. When in domain mode, the information includes domain controller names and trusted domain names.

Each entry in the output is identified by one of the following tags:

- `[*]` Primary domain
- `[.]` Local domain
- `[-]` Other domains
- `[+]` Selected domain controller

`show-groups` [`-m`] [`-p`] [*group*]
Shows information about the specified SMB local group or groups. If no group is specified, information is shown for all groups. If the `-m` option is specified, the group members are also shown. If the `-p` option is specified, the group privileges are also shown.

The following set of subcommands shows information about the user shares, sessions, connections, and open files on a local or a remote server.

An authenticated user account is required to show the types of data listed above, so you must specify the Windows administrative user name with the `-u` option. If the password is not specified on the command line, the user is prompted for it. This user should be the domain administrator or a user who has administrative privileges for the target domain. If a user name is not specified, the Solaris user account name is used.

The *username* can be in any of the formats described under "Operands".

`show-connections` [`-t`] [`-u` *username*] [`-c` *computername* | `-s` *sharename*] *server*
Shows information about the SMB tree connections made on the server. The `-c` option specifies the computer name for connections of interest to the client. The `-s` option specifies the share name for connections of interest to the client. The `-t` option specifies the command header to be displayed.

`show-files` [`-t`] [`-u` *username*] *server*
Shows information about the files open over SMB on the server. The `-t` option specifies the command header to be displayed.

`show-sessions` [`-t`] [`-u` *username*] *server*
Shows information about the SMB user sessions on the server. The `-t` option specifies command header to be displayed.

`show-shares` [`-t`] [`-A` | `-u` *username*] *server*
Shows information about the SMB shares on the server. The `-t` option specifies the command header to be displayed. The `-A` option specifies anonymous user.

Exit Status The following exit values are returned:

0 Successful completion.

>0 An error occurred.

Attributes See the `attributes(5)` man page for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	service/file-system/smb
Utility Name and Options	Uncommitted
Utility Output Format	Not-An-Interface

ATTRIBUTE TYPE	ATTRIBUTE VALUE
`smbadm join`	Obsolete

See Also `passwd(1)`, `groupadd(1M)`, `idmap(1M)`, `idmapd(1M)`, `kclient(1M)`, `mount_smbfs(1M)`, `share(1M)`, `sharectl(1M)`, `smbd(1M)`, `smbstat(1M)`, `smb(4)`, `smbautohome(4)`, `attributes(5)`, `pam_smb_passwd(5)`, `smf(5)`

Name smbd – SMB server daemon

Synopsis `/usr/lib/smbd`

Description The `smbd` daemon handles CIFS/SMB requests from SMB clients, such as Windows clients. Only processes with `{PRIV_SYS_SMB}` and sufficient privileges to write the `/var/run` directory can run this daemon.

The `smbd` daemon is automatically invoked by using the `share(1M)` command or the `zfs(1M) set share` command over all available transports. By default, `smbd` starts over the NetBIOS-Over-TCP (NBT) and TCP transports.

When `smbd` is started over NBT, the following services are started:

- The NetBIOS name service is started on UDP port 137.
- The NetBIOS datagram service is started on UDP port 138.
- The NetBIOS session service is started on TCP port 139.

When the `smbd` daemon is started over TCP, the SMB service is started on TCP port 445.

Only one instance of `smbd` may be running at a time.

Exit Status The following exit values are returned:

0 Daemon exited cleanly.

95 Daemon exited with a fatal error.

96 Daemon exited with a configuration error.

Attributes See the `attributes(5)` man page for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	service/file-system/smb
Interface Stability	Uncommitted

See Also `ps(1)`, `svcs(1)`, `share(1M)`, `sharectl(1M)`, `smbadm(1M)`, `smbstat(1M)`, `svcadm(1M)`, `zfs(1M)`, `smb(4)`, `smbautohome(4)`, `system(4)`, `attributes(5)`, `smf(5)`

Notes Use the `svcadm` command to perform administrative actions on the `smbd` service, such as enabling, disabling, or restarting the service. Use the `svcs` command to query the service status.

The `smbd` service is managed by the service management facility under the service identifier `svc:/network/smb/server`.

If the `smbd` service is disabled, it will be enabled by the `share(1M)` command or the `zfs set share` command, unless its `auto_enable` property is set to false.

Name smbiod, smbiod-svc – SMB client I/O daemon

Synopsis

```
/usr/lib/smbfs/smbiod

/usr/lib/smbfs/smbiod-svc
```

Description `smbiod` and `smbiod-svc` are internal components of the SMB client. They have no external, customer-accessible interfaces. For more information about the SMB client, see the `mount_smbfs(1M)` man pages.

`smbiod-svc` is run by the SMF service, `svc:/network/smb/client:default`, and starts and manages the `smbiod` processes that are required by the SMB client.

`smbiod` is started by `smbiod-svc`, which runs as a separate process for each user who requests SMB client connections. These `smbiod` processes continue to run until the owning user no longer has any SMB client connections.

Files

`/usr/lib/smbfs/smbiod`
SMB I/O daemon.

`/usr/lib/smbfs/smbiod-svc`
SMF service program.

`/var/run/smbiod/.svc`
File on which `smbiod-svc` instantiates a door.

`/var/run/smbiod/$`*UID*
File on which a user's `smbiod` process instantiates a door. $*UID* is the numeric representation of the owner's user ID.

Attributes See the `attributes(5)` man page for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/file-system/smb
Interface Stability	Uncommitted

See Also `ps(1)`, `mount_smbfs(1M)`, `attributes(5)`, `smbfs(7FS)`

Name smbios – display the contents of a System Management BIOS image

Synopsis `smbios [-BeOsx] [-i` *id*`] [-t` *type*`] [-w` *file*`] [`*file*`]`

Description The `smbios` utility displays the contents of the System Management BIOS (SMBIOS) image exported by the current system or stored in a file. SMBIOS is an industry-standard mechanism for low-level system software to export hardware configuration information to higher-level system management software. The SMBIOS data format itself is defined by the Distributed Management Task Force (DMTF). Refer to http://www.dmtf.org for more information about SMBIOS and to obtain a copy of the SMBIOS specification and implementation guidelines.

The SMBIOS image consists of a table of structures, each describing some aspect of the system software or hardware configuration. By default, `smbios` displays the entire contents of the current SMBIOS image. If the `-s` option is specified, `smbios` displays a summary of the structures that are present in the image. If the `-w` option is specified, `smbios` writes a copy of the SMBIOS image to the specified file. `smbios` can then be applied to the resulting file to display its content.

`smbios` attempts to display each structure and its content in a human- readable fashion. If `smbios` does not recognize a structure's type or content, the raw hexadecimal data for the structure is displayed.

Options The following options are supported:

`-B` Disable header validation for broken BIOSes.

By default, `smbios` attempts to validate the SMBIOS header by verifying the anchor strings, header checksums, and version number. This option might be necessary when a BIOS has a non-compliant header.

`-e` Display the contents of the SMBIOS entry point rather than the contents of the SMBIOS structure table.

`-i` *id* Display only the specified structure, named by its integer id.

`-O` Display obsolete structure types.

By default, `smbios` elides output for structures whose type is marked as obsolete in the DMTF SMBIOS specification.

`-s` Display only a summary listing of the structure identifiers and types, instead of the content of each selected structure.

`-t` *type* Display only those structures whose type matches the specified integer type, as defined the DMTF SMBIOS specification.

`-w`*file* Write a copy of the SMBIOS image to the specified file and exit.

The SMBIOS entry point is written to the start of the file with its structure table address set to the file offset of the structure table, and a new entry point checksum is computed.

`-x` Display raw hexadecimal data for the selected structures in addition to human-readable output.

By default, hexadecimal data is only displayed if `smbios` cannot display human-readable output for the selected structures.

Operands The following operands are supported:

file Specifies an alternate SMBIOS image to display instead of the current system's SMBIOS image.

Exit Status The following exit values are returned:

`0` Successful completion. All structures in the SMBIOS image were examined successfully.

`1` A fatal error occurred, such as failure to open the specified file or device, or corruption in the image.

`2` Invalid command-line options were specified.

Files `/dev/smbios` Kernel SMBIOS image device. This device special file is used to export a snapshot of the current system SMBIOS image.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os
Interface Stability	See below.

The command-line options are Committed. The human-readable output is Uncommitted.

See Also `prtdiag(1M)`, `attributes(5)`, `smbios(7D)`

System Management BIOS Reference Specification (see http://www.dmtf.org)

Notes The implementation of a System Management BIOS image is entirely at the discretion of the system and BIOS vendors. Not all systems export an SMBIOS. The SMBIOS structure content varies widely between systems and BIOS vendors and frequently does not comply with the guidelines included in the specification. Some structure fields might not be filled in by the BIOS at all, and others might be filled inwith non-conforming values.

Name smbstat – show Solaris SMB file server statistics

Synopsis

```
smbstat [-r [-n [-a | -z]] [-t] [-u] [-c] interval
```

Description The `smbstat` command shows statistical information for the `smbd`(1M) server. `smbstat` has a number of options, described below, and a single operand, *interval*. If *interval* is specified, the first display captures statistics since the server started, up to the moment the command was entered. Subsequent displays capture statistics for the last *interval*.

By default, the `smbstat` command shows all statistics.

Options The `smbstat` command includes the following options:

`-c`
Display counters.

`-r`
Display the statistics of the requests. You can combine `-r` with the following options.

`-a`
Display statistics for all the types of server requests, whether valid or not. Note that there are 256 types of server requests. The `-a` and `-z` options are mutually exclusive.

`-n`
Display in alphabetic order.

`-z`
Display statistics for requests actually received. The `-a` and `-z` options are mutually exclusive.

`-t`
Display the throughput of the SMB server.

`-u`
Display the utilization of the SMB server.

Extended Description The `smbstat` command displays the headings shown below. The headings displayed for a given command vary according to which option(s) are specified. There are four categories of headings, corresponding to categories of statistics: counters, throughput, utilization, and (server) requests.

Counters

`nbt`
Number of SMB NetBIOS-over-TCP (NBT) sessions.

`tcp`
Number of SMB TCP sessions.

`users`
Number of users logged in.

`trees`
Number of trees connected.

`files`
Number of open files.

`pipes`
Number of open pipes.

Throughput `rbytes/s`
Number of bytes received per second.

`tbytes/s`
Number of bytes transmitted per second.

`reqs/s`
Number of requests handled per second.

`reads/s`
Number of read requests per second. This would be an aggregation of the following requests: `SMB_COM_READ`, `SMB_COM_LOCK_AND_READ`, `SMB_COM_READ_RAW`, and `SMB_COM_READ_ANDX`.

`writes/s`
Number of write requests per second. This would ba an aggregation of the following requests: `SMB_COM_WRITE`, `SMB_COM_WRITE_AND_UNLOCK`, `SMB_COM_WRITE_RAW`, and `SMB_COM_WRITE_AND_CLOSE`.

Utilization `wcnt`
Average number of requests received but waiting for an SMB worker thread to execute them.

`rcnt`
Average number of requests being simultaneously executed by an SMB worker thread.

`wtime`
Average time a request waits before an SMB worker thread starts executing it.

`rtime`
Average execution time of a request.

`w%`
Percentage of the time during which at least one request was waiting.

`r%`
Percentage of the time during which at least one request was being executed.

`u%`
Percentage of utilization of the SMB server. This number is defined as: `rcnt` / (*max_worker_threads*).

`sat`
Flag indicating if the server has been saturated in the past. Saturation is defined as: `u%` == 100%.

`usr%`
Percentage of the time the processor(s) spent in user space.

`sys%`
Percentage of the time the processor(s) spent in kernel space.

`idle%`
Percentage of the time the processor(s) was(were) idle.

Requests The following headings are displayed for each type of request.

`code`
Code of the request, in hexadecimal.

`%`
Percentage of a given type of request.

`count`
Number of requests received.

`rbytes/s`
Number of bytes received per second.

`tbytes/s`
Number of bytes received per second.

`req/s`
Number of requests handled per second.

`rt-mean`
Average response time in seconds.

`rt-stddev`
Standard deviation of the response time.

Examples

EXAMPLE 1 Combining Options

The following command combines the `-c`, `-t`, and `-u` options.

```
% smbstat -ctu
nbt   tcp users trees files pipes
0     1     1     2    20     0

rbytes/s   tbytes/s    reqs/s     reads/s   writes/s
1.036e+02  1.298e+00  0.000e+00  0.000e+00  0.000e+00

wcnt       rcnt       wtime      rtime       w%   r%  u% sat usr% sys% idle%
4.317e-01  7.410e+00  2.461e-05  4.224e-04   31  100  0  no  0    76   24
```

Exit Status The following exit values are returned:

0 Successful completion.

>0 An error occurred.

Attributes See the attributes(5) man page for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	service/file-system/smb
Interface Stability	Uncommitted
Utility Output Format	Not-an-Interface

See Also sharectl(1M), smbadm(1M), smbd(1M), attributes(5)

Name smrsh – restricted shell for sendmail

Synopsis `smrsh -c` *command*

Description The `smrsh` program is intended as a replacement for the `sh` command in the `prog` mailer in `sendmail(1M)` configuration files. The `smrsh` program sharply limits commands that can be run using the `|program` syntax of `sendmail`. This improves overall system security. `smrsh` limits the set of programs that a programmer can execute, even if `sendmail` runs a program without going through an `alias` or `forward` file.

Briefly, `smrsh` limits programs to be in the directory `/var/adm/sm.bin`, allowing system administrators to choose the set of acceptable commands. It also rejects any commands with the characters: `,`, `<`, `>`, `|`, `;`, `&`, `$`, `\r` (RETURN), or `\n` (NEWLINE) on the command line to prevent end run attacks.

Initial pathnames on programs are stripped, so forwarding to `/usr/ucb/vacation`, `/usr/bin/vacation`, `/home/server/mydir/bin/vacation`, and `vacation` all actually forward to`/var/adm/sm.bin/vacation`.

System administrators should be conservative about populating `/var/adm/sm.bin`. Reasonable additions are utilities such as `vacation(1)` and `procmail`. Never include any shell or shell-like program (for example, `perl`) in the `sm.bin` directory. This does not restrict the use of `shell` or `perl` scrips in the `sm.bin` directory (using the `#!` syntax); it simply disallows the execution of arbitrary programs.

Options The following options are supported:

`-c` *command* Where *command* is a valid command, executes *command*.

Files `/var/adm/sm.bin` directory for restricted programs

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also `sendmail(1M)`, , `attributes(5)`

Name smtp-notify – email notification daemon for software events

Synopsis

```
/usr/lib/fm/notify/smtp-notify
```

Description `smtp-notify` is a daemon that subscribes to software events and Fault Management lifecycle events and produces email notifications based on a set of notification preferences that are stored in the SMF service configuration repository.

Email notification preferences are set using `svccfg(1M)`.

`smtp-notify` is managed by the service management facility, `smf(5)`, under the service FMRI:

```
svc:/system/fm/smtp-notify:default
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. The service's status can be queried using the `svcs(1)` command.

Properties The following service properties can be set:

`config/rootdir`
: This is an `astring` property that defaults to `/`. When set, the specified root directory will be used for all pathnames evaluated by `smtp-notify`.

By default, the body of the email event notification will be based on a set of localized message templates that are delivered with the Oracle Solaris operating system. `smtp-notify` will lookup the appropriate template (based on the event class) and fill in the template using elements from the event payload.

A user-supplied message body template can be substituted by specifying the `msg_template` notification preference. Refer to `svccfg(1M)` for more information about setting notification preferences on a per-service or per-event-class basis.

The message body template can contain the expansion macros listed below. These macros will be expanded by `smtp-notify` before sending the message.

```
Macro          Description
-----          -----------
%%              expands to a literal % character
%<HOSTNAME>     expands to the hostname on which the event
                occurred
%<URL>          expands to the URL of the knowledge article
                associated with this event
%<CLASS>        expands to the event class
%<UUID>         expands to the UUID of the event
%<CODE>         expands to the knowledge article message ID
%<SEVERITY>     expands to the severity of the event
```

For SMF service state transition events, the following additional macros can be specified in a message template:

```
%<FMRI>          expands to FMRI of the affected service
%<FROM-STATE>    expands to the previous state of the service
%<TO-STATE>      expands to the new state of the service
```

To facilitate email filtering, `smtp-notify` will create X-headers for the following components of the event payload for all event classes:

```
Header name             Value
-----------             -----
X-FMEV-HOSTNAME         the name of the host on which the
                        event occurred
X-FMEV-CLASS            the event class
X-FMEV-CODE             the Knowledge article message ID
X-FMEV-SEVERITY         the severity of the event
X-FMEV-UUID             the UUID of the event
```

SMF service state transition events will also include the following additional X-headers:

```
Header name             Value
-----------             -----
X-FMEV-FMRI             the FMRI of the affected SMF service
X-FMEV-FROM-STATE       the previous state of the service
X-FMEV-TO-STATE         the new state of the service
```

Examples

EXAMPLE 1 Configuring Notification Preferences

The following command configures notification preferences for SMF service state transition events.

```
# svccfg -s svc:/system/svc/global:default setnotify -g \
from-online,to-maintenance mailto:admin@somehost.com
```

EXAMPLE 2 Enabling Email Notifications

The following command enables email notifications for Fault Management problem diagnosis events.

```
# svccfg setnotify problem-diagnosed mailto:admin@somehost.com
```

EXAMPLE 3 Listing Notification Settings

The following command lists all notification settings for Fault Management problem diagnosis events.

```
# svccfg listnotify problem-diagnosed
```

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/fault-management/smtp-notify

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Unstable

See Also svcs(1), svcadm(1M), svccfg(1M), syslogd(1M), attributes(5), smf(5)

Name sndradm – control Sun StorageTek Availability Suite Remote Mirror operations

Synopsis

```
sndradm -I a master shadow bitmap

sndradm -I d master shadow bitmap

sndradm -h usage message

sndradm -v version information

sndradm [options] -e [sndr_set]

sndradm [options] -E [sndr_set]

sndradm [options] -d [sndr_set]

sndradm [options] -D block [sndr_set]

sndradm [options] -D noblock [sndr_set]

sndradm [options] -l [sndr_set]

sndradm [options] -m [sndr_set]

sndradm [options] -m -r [sndr_set]

sndradm [options] -u [sndr_set]

sndradm [options] -u -r [sndr_set]

sndradm [options] -w [sndr_set]

sndradm [options] -H [sndr_set]

sndradm [options] -p [sndr_set]

sndradm [options] -P [sndr_set]

sndradm [options] -q a volume [sndr_set]

sndradm [options] -q d [sndr_set]

sndradm [options] -q r volume [sndr_set]

sndradm [options] -i [sndr_set]

sndradm [options] -a value [sndr_set]

sndradm [options] -A value [sndr_set]

sndradm [options] -F value [sndr_set]

sndradm [options] -W value [sndr_set]

sndradm [options] -R

sndradm [options] -R b p bitmap [sndr_set]

sndradm [options] -R b s bitmap [sndr_set]

sndradm [options] -R C tag [sndr_set]
```

```
sndradm [options] -R g io_groupname [sndr_set]

sndradm [options] -R m sync [sndr_set]

sndradm [options] -R m async [sndr_set]

sndradm [options] -R -f volset-file

sndradm [options] -R r [sndr_set] *
```

Description

The `/usr/sbin/sndradm` command is the administrative command line interface for the Sun StorageTek Availability Suite Remote Mirror software. Remote Mirror enables you to replicate disks between different physically-separate Sun servers in real time. Remote Mirror is conceptually similar to the local disk mirroring scheme of RAID 1 but it performs its replication operations over longer distances.

If you do not specify a Remote Mirror set (*sndr_set*) on the command line, `sndradm` operates on all configured Remote Mirror sets.

The `sndradm` command generates an entry in the Availability Suite log file, `/var/adm/ds.log` (see `ds.log(4)`), for all operations except print (`-p`, `-P` and `-i`), help (`-h`), and version (`-v`).

Options

The `sndradm` utility supports the following options:

`-f` *volset-file*
: Specifies a file containing the *sndr_set* information for one or more Remote Mirror sets in the same format as the fully specified command line *sndr_set* documented below.

`-g` *io_groupname*
: Limits operations to only those Remote Mirror sets belonging to *io_groupname*.

  The *io_groupname* for a given set must be consistent across both the primary and the secondary hosts.

`-C` *tag*
: On a clustered node, limits operations to only those Remote Mirror sets belonging to the cluster resource group or disk group name specified by *tag*. This option is not valid on a system that is not clustered.

`-n`
: Does not prompt the user after starting a Remote Mirror operation using `sndradm`. For all but the printing, help, and version options, the default behavior is to prompt for a response. For example, after starting a full synchronization from the primary to the secondary volume, Remote Mirror prompts: "`Overwrite secondary with primary? (Y/N) [N]`".

sndr_set
: Specifies the Remote Mirror set. For a set that has already been enabled, this can be a `set_name` in the format *shost*:*sdev*. You can supply a fully specified Remote Mirror set in the same format as a configuration file:

  phost pdev pbitmap shost sdev sbitmap `ip {sync | async} \`
  `[g` *io_groupname*`] [C` *tag*`]`

These parameters are described as follows:

phost
Specifies the server on which the primary volume resides.

pdev
Specifies the primary volume partition to be replicated. Specify full pathnames only (for example, `/dev/rdsk/c0t1d0s2`).

pbitmap
Specifies the volume partition on which the bitmap (scoreboard log) of the primary partition is stored. Specify full pathnames only (for example, `/dev/rdsk/c0t1d0s3`).

shost
Specifies the server on which the secondary volume resides.

sdev
Specifies the secondary volume partition. Specify full path names only (for example, `/dev/rdsk/c0t1d0s4`).

sbitmap
Specifies the volume partition on which the bitmap (scoreboard log) of the secondary partition is stored. Specify full path names only (for example, `/dev/rdsk/c0t1d0s5`).

`ip`
Specifies the network transfer protocol.

`sync | async`
Specifies the Remote Mirror operating mode. `sync` is the Remote Mirror mode where the I/O operation is not confirmed as complete until the remote volume has been updated. `async` is the Remote Mirror mode where the primary host I/O operation is confirmed as complete before updating the remote volume.

io_groupname
Specifies the name of the Remote Mirror consistency group to which the Remote Mirror set belongs. In asynchronous mode, write ordering must be preserved across all replicating volumes in a Remote Mirror consistency group. This ensures that the secondary volumes belonging to the group contains a valid point-in-time copy of the corresponding primary volumes.

When adding an existing set to a consistency group or when enabling a set to be in a group, the set must be configured with the same group name on both the primary and the secondary hosts.

tag
For operation within a cluster, this specifies the disk group name or resource tag of the local data and bitmap volumes in cases where this is not implied by the name of the volume (for example, `/dev/rdsk/md/dg/vol` and `/dev/vx/rdsk/dg/vol` both indicate a disk group name of `dg`). It is the responsibility of the user to ensure that the cluster tag specified to the Remote Mirror matches the appropriate cluster resource group tag, and

to keep all the Availability Suite services up to date in the event of cluster resource group reconfigurations. It is illegal to specify the cluster resource tag on a system that is not clustered.

Parameters A valid `sndradm` command must specify one of the parameters listed below.

`-I a` *master shadow bitmap*
Add an `ndr_ii` entry with the specified master, shadow, and bitmap to the Availability Suite configuration file. See `sndrsyncd(1M)`. If the corresponding Point-in-Time Copy set does not exist, it is enabled when the next `sync` command is issued on the related volume(s). When no longer required, this Point-in-Time Copy set can be disabled by `iiadm -d`. See `iiadm(1M)`

`-I d` *master shadow bitmap*
Delete the `ndr_ii` entry with the specified master, shadow, and bitmap from the Availability Suite configuration file. Use the `dscfg` command to list `ndr_ii` configuration entries.

`-a` *value*
Specifies the value, on or off, of the automatic sync variable for the set. Once `autosync` has been requested for a set, the functionality is active from the time a sync operation is requested until the set is manually put into logging mode. Once the set is manually put into logging mode, the `autosync` functionality is not active and remains inactive until the next time a `sync` request is made. To check whether `autosync` is active, use `sndradm -P`. To check whether autosync has been requested for a set, look for the"`auto=on;`" tag for the set in the output of `dscfg -l`. See `sndrsyncd(1M)`.

`-A` *value*
Specifies the maximum number of threads that can be created to process the asynchronous queue when a set is replicating in asynchronous mode. The default is `2`.

`-W` *value*
Specifies the maximum number of writes that can be queued to a set replicating in asynchronous mode. The default is `4096`. For example, set this value to 1 to ensure that the secondary volume is never more than one write operation behind the primary volume.

Tuning the maximum number of writes is only valid for sets using memory-based async I/O queues. This value is ignored when disk based I/O queues are used.

`-D {block | noblock}`
Toggles the `block|noblock` attribute of a disk-based queue. The default setting is `block`. If the I/O fill rate is larger than the drain rate for enough time for the queue to fill, incoming I/O is blocked until there is adequate space on the queue for it. This is to preserve write ordering whether it is one volume or across many volumes in the same consistency group. If `noblock` is set, and incoming I/O fills the queue, the I/O is not blocked. Instead, the set is put into logging and the disk queue contents are disregarded. An ensuing update sync synchronizes the latest data to the secondary site.

`-F` *value*
Specifies the maximum number of 512–byte FBAs that can be queued in kernel memory to a set replicating in asynchronous mode. The default is `16384`.

Tuning the maximum number of FBAs is valid only for sets using memory–based async I/O queues. This value isignored when disk-based I/O queues are used.

`-h`
Prints the `sndradm` usage summary.

`-v`
Prints the Remote Mirror version number.

`-e`
Enables Remote Mirror for the set and enables scoreboard logging. The scoreboard is set to indicate that a full synchronization is required. Details of the set are saved in the current configuration. See `dscfg`(1M). The local volume and the bitmap volume are enabled for the Storage Volume driver (see `sv`(7D)).

`-E`
Enables Remote Mirror for the set and enables scoreboard logging. The scoreboard is cleared to indicate that the primary and secondary volumes are already guaranteed to be fully synchronized. Details of the set are saved in the current configuration. See `dscfg`(1M). The local volume and the bitmap volume are enabled for the Storage Volume driver (see `sv`(7D)).

`-d`
Disables Remote Mirror for the set and halts any current synchronization operations. `sndradm -d` also discards any active scoreboards that track temporary differences between primary and secondary volumes.

`-l`
Stops Remote Mirror replication and copy operations between primary and secondary volumes and starts independent Remote Mirror scoreboard logging on these volumes. When all the sets in a consistency group are replicating, it means that the secondary volumes contain a valid point-in-time copy of the corresponding primary volumes. Under this condition, as soon as one Remote Mirror set drops into logging mode, the `rdc` kernel module drops all the other sets in the group into logging mode automatically. This ensures that the secondary volumes still contains a valid point-in-time copy. To resume the Remote Mirror after using the `-l` parameter, use the `-m` parameter to perform a full resynchronization or the `-u` parameter to perform an update resynchronization (based on the scoreboard).

This option does not work on the secondary for any volumes that are currently synchronizing.

`-w`
Waits for a synchronization copy to complete or abort, or returns immediately if invoked on the secondary system.

`-H`
Reports on the health of the network link used by the specified volume set. The health of the link is reported as active or inactive. Active means that the network link is actively being used for replicating or resynchronizing data, and is therefore in good health. Inactive means that the network link is not actively being used for replicating or resynchronizing data, which might indicate a problem with the link.

`-p`
Displays a list of configured Remote Mirror volumes or sets.

`-P`
Displays a list of configured Remote Mirror volumes or sets with extra details. (See state descriptions, below.)

`-q a` *volume*
Add a disk queue to a set or group. This operation is valid when the set or group is in logging mode.

`-q d`
Remove a disk queue from a set or group. This operation is valid when the set or group is in logging mode.

`-q r` *volume*
Replace a disk queue for a group or set. The queue is removed from the set or group as in the queue-disable operation and the new disk queue is added as in the queue-add operation. This operation is valid when the set or group is in logging mode.

`-i`
Displays a list of configured Remote Mirror volumes or sets in the same format as the *volset-file*.

`-R`
Attempt to reset a Remote Mirror set's error condition such as failed bitmaps.

`-R b p` *bitmap*
Reconfigure a Remote Mirror set's primary bitmap. This command should be entered on both primary and secondary servers. It is only possible to reconfigure the primary bitmap for one set at a time.

`-R b s` *bitmap*
Reconfigure a Remote Mirror set's secondary bitmap. This command should be entered on both primary and secondary servers. It is only possible to reconfigure the secondary bitmap for one set at a time.

`-R C` *tag*
Reconfigure the cluster tag, or disk group name, of a Remote Mirror set's local volumes, in those cases where this is not indicated by the pathname. This does not affect the remote volumes. This parameter cannot be used on a system that is not clustered.

`-R m {sync | async}`
Reconfigure the replication mode of a Remote Mirror set. The sets belonging to a consistency group must be either all synchronous or all asynchronous. It is not possible to mix modes within a group.

`-R g` *group*
Reconfigure the consistency group of a Remote Mirror set. This command should be entered with the same group name on both primary and secondary servers.

To remove a set from a consistency group, specify the null string (" ") when reconfiguring the consistency group.

The following parameters can be issued only from the primary server:

`-m`
Starts a full volume copy from the primary volume to the secondary volume, and concurrently enables Remote Mirror replication of new updates from the primary volume to the secondary volume. Use this parameter when the primary and secondary volumes might be different and no logging information exists to incrementally resynchronize the volumes. See EXIT STATUS.

`-r`
Reverses the direction of the synchronization so the primary volume is synchronized from the secondary volume. Use this parameter with the `-m` or `-u` parameter. `-m -r` starts a full volume copy from the secondary (source) volume to the primary (target) volume but concurrently enables Remote Mirror replication of new updates from the primary (source) volume to the secondary (target) volume, ensuring the volume sets remain synchronized. Use `-m -r` when the primary and secondary volume content might differ and the secondary has the desired contents, yet no logging information exists to incrementally resynchronize the volumes (using `-u`). `-u -r` resynchronizes the primary (target) volume from the secondary (source) volume, using the Remote Mirror scoreboard logs maintained while replication was suspended. It then resumes Remote Mirror replication of new updates from the primary volume to the secondary volume so that the volume sets remain synchronized. Quiesce the workload to the volume sets during the restore/refresh operation. This action ensures that the primary and secondary volumes match before replication of new updates resumes.

`-u`
Updates a Remote Mirror volume set. This parameter resynchronizes a Remote Mirror volume set. Only the blocks logged as changed in the Remote Mirror scoreboards are updated. Enables Remote Mirror replication for the primary volume and also uses the Remote Mirror scoreboard logs to start the resynchronization process so that the corresponding secondary volume matches the primary volume.

States Returned from `sndradm -P`

The following are the states that can be returned from `sndradm -P`.

`volume failed`
An I/O operation to the local data volume has failed

`bitmap failed`
An I/O operation to the local bitmap volume has failed

`disk queue failed`
An I/O operation to disk queue volume has failed

`need sync`
A sync to this volume has been interrupted. It needs to be completed (or restored via Point-in-Time Copy). The direction of the data flow must not be changed until one or the other is done.

`need reverse sync`
A reverse sync to this volume has been interrupted. It needs to be completed (or restored via Point-in-Time Copy). The direction of the data flow must not be changed until one or the other is done.

`logging`
Incoming writes are logged in the bitmap only. Data is not replicated to the remote site. `need sync`, `need reverse sync`, and `queuing` are all substates of logging such that writes are logged in the bitmap, but not replicated. Queuing mode (described below) logs the writes to the bitmap, and queues the request for later replication by the async flushers.

`reverse syncing`
A secondary to primary copy is in progress.

`syncing`
A primary to secondary copy is in progress.

`queuing`
During normal async replication using disk queues, i/o is placed on the disk queue to be replicated by the async flusher threads. In the event of a temporary link failure, the set transitions to queuing mode. The queue is not discarded, as it would be with memory based queues. Instead, data is logged in the bitmap and placed on the queue. When the link comes up, and `sndradm -u` is issued, (automated by turning autosync on for the set) the flushers restarts. This preserves write ordering through a temporary link outage. If write ordering is not necessary, and only the latest data is needed, the set can be put into logging manually (`sndradm -l`) and an update sync issued (`sndradm -u`). This action discards the data on the queue, and fast resyncs using the bitmap. If the queue fills before the link comes back and the update sync is issued, the queue is discarded and the set put into logging mode to avoid application hangs.

Examples

EXAMPLE 1 Enabling a Remote Mirror Set

The following command enables a Remote Mirror asynchronous set on host `example`, where `example` is the primary host and `example-remote` is the secondary host.

```
example% sndradm -e example /dev/rdsk/c1t0d0s1 /dev/rdsk/c1t1d0s3 \
example-remote /dev/rdsk/c2t3d0s5 /dev/rdsk/c2t4d0s5 ip async
```

EXAMPLE 2 Adding a Disk Queue to an Asynchronous Set

The following command adds a disk queue volume to an asynchronous set.

```
example% sndradm -q a /dev/rdsk/c1t2d0s3 \
example-remote:/dev/rdsk/c2t3d0s5
```

EXAMPLE 3 Removing a Disk Queue from an Asynchronous Set

The following command removes the disk queue volume from a set with a disk queue volume attatched to it.

```
example% sndradm -q d example-remote:/dev/rdsk/c2t3d0s5
```

EXAMPLE 4 Disabling a Remote Mirror Set

The following command disables a Remote Mirror set enabled on host `example`.

```
example% sndradm -d example-remote:/dev/rdsk/c2t3d0s5
```

Exit Status

`0` Command completed successfully.

`>0` An error occurred.

When the `-m` or `-u` option is executed in a script, the exit status following one of these options always returns success, regardless of the current status of the Remote Mirror set.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	storage/avs/avs-remote-mirror
Interface Stability	Committed

See Also

`dscfg(1M)`, `sndrd(1M)`, `sndrsyncd(1M)`, `ds.log(4)`, `rdc.cf(4)`, `attributes(5)`, `sv(7D)`

Name sndrd – Remote Mirror daemon

Synopsis `/usr/lib/sndrd [-c` *max_connections*`] [-l` *listen_backlog*`]`

Description The `sndrd` daemon processes client Remote Mirror requests. Only the root user or a user with equivalent privileges can run this daemon. The daemon is automatically invoked in run level 2. `sndrd` restarts the TCP transport layers.

Administrators wanting to change startup parameters for `sndrd` should, as root or equivalent, make changes in the `/etc/default/sndr` file rather than editing the `/lib/svc/method/svc-rdcsyncd` file. See `sndr(4)`.

Options The `sndrd` daemon supports the following options:

`-c` *max_connections*
Sets the maximum number of connections allowed to the server over connection-oriented transports. By default, the number of connections is 16.

`-l` *listen_backlog*
Sets connection queue length for the RDC TCP over a connection-oriented transport. The default value is 10 entries.

Exit Status

`0`	Daemon started successfully.
`>0`	Daemon failed to start.

Error information is reported to `syslog` at level `LOG_ERR`.

Files

`/lib/svc/method/svc-rdcsyncd`
Shell script for starting `sndrd`.

`/lib/svc/method/svc-rdc`
Shell script for stopping `sndrd`.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	storage/avs/avs-remote-mirror
Interface Stability	Committed

See Also `svcadm(1M)`, `syslogd(1M)`, `ds.log(4)`, `attributes(5)`

Notes Do not manually stop the sndrd daemon. If you need to manually stop sndrd perform these steps. This stops both the `sndrd` and `sndrsyncd` daemons.

```
# svcadm disable svc:/system/nws_rdc

# svcadm disable svc:/system/nws_rdcsyncd
```

Do not manually start or restart the sndrd daemon. If you need to manually start sndrd perform these steps. This starts both the sndrd and sndrsyncd daemons.

```
# svcadm enable svc:/system/nws_rdc

# svcadm enable svc:/system/nws_rdcsyncd
```

See svcadm(1M) for additional information.

Name sndrsyncd – Availability Suite Remote Mirror update resynchronization daemon

Synopsis

```
/usr/lib/sndrsyncd
```

Description The `sndrsyncd` daemon automates update resynchronization after a network or machine failure and invokes Point-in-Time Copy copies when needed to protect the data volumes being updated during a resynchronization.

The daemon is notified by the kernel when a network link being used by Remote Mirror goes down and invokes the `sndradm`(1M) command with the `-u` option to resynchronize all Remote Mirror sets which have autosync switched on and are using the network link. See `sndradm`(1M) for details on how to configure autosync for a Remote Mirror set.

The daemon is also notified when any Remote Mirror resynchronization starts or completes. The daemon takes Point-in-Time Copy snapshots, if configured in the Availability Suite configuration file. On a secondary server, the daemon checks if a file system is currently mounted on the secondary volume and informs the kernel not to allow the synchronization to start if the file system is currently mounted.

Exit Status

`0` Daemon started successfully.

`>0` Daemon failed to start.

Files

`/lib/svc/method/svc-rdcsyncd`
Shell script for starting `sndrsyncd`.

`/lib/svc/method/svc-rdc`
Shell script for stopping `sndrsyncd`.

Attributes See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	storage/avs/avs-remote-mirror
Interface Stability	Committed

See Also `iiadm`(1M), `sndradm`(1M), `sndrd`(1M), `svcadm`(1M), `ds.log`(4), `rdc.cf`(4), `attributes`(5)

Notes Do not manually stop the sndrsyncd daemon. If you need to manually stop sndrsyncd perform these steps. This stops both the `sndrd` and `sndrsyncd` daemons.

```
# svcadm disable svc:/system/nws_rdc

# svcadm disable svc:/system/nws_rdcsyncd
```

Do not manually start or restart the `sndrsyncd` daemon. If you need to manually start `sndrsyncd` perform these steps. This starts both the `sndrd` and `sndrsyncd` daemons.

```
# svcadm enable svc:/system/nws_rdc

# svcadm enable svc:/system/nws_rdcsyncd
```

See svcadm(1M) for additional information.

Name snmpdx – Sun Solstice Enterprise Master Agent

Synopsis

```
/usr/lib/snmp/snmpdx [-hy] [-a filename] [-c config-dir]
     [-d debug-level] [-i filename] [-m GROUP -m SPLIT]
     [-o filename] [-p port] [-r filename]
```

Description The Master Agent, `snmpdx`, is the main component of Solstice Enterprise Agent (SEA) technology. It runs as a daemon process and listens to User Datagram Protocol (UDP) port 161 for SNMP requests. The Master Agent also opens another port to receive SNMP trap notifications from various subagents. These traps are forwarded to various managers, as determined by the configuration file.

Upon invocation, `snmpdx` reads its various configuration files and takes appropriate actions by activating subagents, determining the subtree Object Identifier (OID) for various subagents, populating its own Management Information Bases (MIBs), and so forth. The Master Agent invokes subagents, registers subagents, sends requests to subagents, receives responses from subagents, and traps notifications from subagents.

The Master Agent is invoked by the service management facility `smf(5)` at boot time if `svc:/application/management/snmpdx` is enabled (see NOTES) and contents of the resource configuration file `/etc/snmp/conf/snmpdx.rsrc` are non-trivial.

Note – The SMA (Systems Management Agent) is the default SNMP agent in the Solaris operating system. `snmpdx` is Obsolete and may not be supported in a future release of Solaris.

Options The following options are supported:

`-a`*filename* Specify the full path of the access control file used by the Master Agent. The default access control file is `/etc/snmp/conf/snmpdx.acl`.

`-c`*config-dir* Specify the full path of the directory containing the Master Agent configuration files. The default directory is `/etc/snmp/conf`.

`-d`*debug-level* Debug. Levels from 0 to 4 are supported, giving various levels of debug information. The default is 0 which means no debug information is given.

`-h` Help. Print the command line usage.

`-i`*filename* Specify the full path of the enterprise-name OID map. This file contains the PID used by the Master Agent for recovery after a crash. It contains tuples of the UNIX process ID, port number, resource name, and agent name. The default file is `/var/snmp/snmpdx.st`.

`-m GROUP | -m SPLIT` Specify the mode to use for forwarding of SNMP requests.

`GROUP` Multiple variables can be included in each request from the Master Agent to the subagents. This results in, at mose, one send-request per agent.

`SPLIT` Each variable in the incoming request results in one send-request to each subagent.

The default is `GROUP`.

`-o`*filename* Specify the full path of the file containing the tuple (enterprise-name, OID). For example, (Sun Microsystems, 1.3.1.6.1.4.32). The Master Agent uses this file as a base for look-up in the trap-filtering and forwarding process. The default file is `/etc/snmp/conf/enterprises.oid`.

`-p`*port* Specify the port number. The default port number is 161.

`-r`*filename* Specify the full path of the resource file to be used by the Master Agent. This file stores information about the subagents that the Master Agent invokes and manages. The default resource file is `/etc/snmp/conf/snmpdx.rsrc`.

`-y` Set a recovery indicator to invoke the recovery module. The recovery process discovers which subagents in the previous session are still active; those subagents not active are re-spawned by the Master Agent.

Files

`/etc/snmp/conf/enterprises.oid`	Enterprise-name OID map
`/etc/snmp/conf/snmpdx.acl`	Access control file
`/etc/snmp/conf/snmpdx.rsrc`	Resource configuration file
`/var/snmp/snmpdx.st`	Master Agent status file
`/var/snmp/mib/snmpdx.mib`	Master Agent MIB file

Exit Status The following error values are returned:

`0`	Successful completion.
non-zero	An error occurred.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/management/snmp/sea
Interface Stability	Obsolete

See Also `attributes(5)`, `smf(5)`

Notes The `snmpdx` service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/application/management/snmpdx
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. The service's status can be queried using the `svcs(1)` command.

Name snmp-notify – SNMP notification daemon for software events

Synopsis `/usr/lib/fm/notify/snmp-notify`

Description `snmp-notify` is a daemon that subscribes to software events and FMA diagnosis and repair events and generates SNMP trap notifications based on a set of notification preferences that are stored in the SMF service configuration repository.

SNMP notification preferences are set using `svccfg(1M)`.

`snmp-notify` is managed by the service management facility, `smf(5)`, under the service FMRI:

`svc:/system/fm/snmp-notify:default`

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. The service's status can be queried using the `svcs(1)` command.

Properties The following service properties can be set:

`config/rootdir`
This is an `astring` property that defaults to `/`. When set, the specified root directory will be used for all pathnames evaluated by `snmp-notify`.

Examples **EXAMPLE 1** Configuring Notification Preferences

The following command configures notification preferences for SMF service state transition events.

```
# svccfg setnotify -g from-online,to-maintenance snmp:active
```

EXAMPLE 2 Enabling SNMP Notifications

The following command enables notifications for Fault Management diagnosis events.

```
# svccfg setnotify problem-diagnosed snmp:active
```

EXAMPLE 3 Listing Notification Settings

The following command lists all notification settings for Fault Management diagnosis events.

```
# svccfg listnotify fma-diagnosis
```

Refer to `svccfg(1M)` for more information on configuring SNMP notification preferences.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/fault-management/snmp-notify
Interface Stability	Unstable

See Also svcs(1), svcadm(1M), svccfg(1M), syslogd(1M), attributes(5), smf(5)

Name snmpXwbemd – SNMP Adapter Subagent for WBEM

Synopsis

```
/usr/sadm/lib/wbem/snmpXwbemd [-d] [-h] [-p port]
```

Description The `snmpXwbemd` daemon is a subagent in the Web-Based Enterprise Management (WBEM) services package.

This daemon maps the Simple Network Management Protocol (SNMP) requests forwarded by the Solstice Enterprise Agents (SEA) Master Agent `snmpdx(1M)` into one or more equivalent WBEM Common Information Model (CIM) properties and instances. Further, it remaps the response from the CIM Object Manager into a SNMP response, which it passes back to `snmpdx`(1M).

A mapping file contains the corresponding Object Identifier (OID), class name, property name, and Abstract Syntax Notation 1 (ASN.1) type for each object. You can also create your own mapping file.

Options The following options are supported:

`-d` Displays all debug information.

`-h` Displays help by printing the correct command line usage.

`-p` Specifies the port number to use.

Operands The following operand is supported:

port Specifies the port number you want to use.

Examples **EXAMPLE 1** An Example of a `050SUNWwbcou.map` File

This mapping file that Sun Microsystems provides contains definitions of objects, in this format:

```
#
#pragma ident    "@(#)050SUNWwbcou.map    1.0    01/04/03 SMI"
#
# Copyright (c) 2001 by Sun Microsystems, Inc.
# All rights reserved.
#
# *** Description of contents ***
#
# First non-commented non-blank line contains required Version label.
# Remaining non-commented non-blank lines are considered map entries
# used as described below:
#
# Column 1 - SNMP OID - Uniquely describes an SNMP variable
# Column 2 - CIM Class Name - CIM class associated with this variable
# Column 3 - CIM Property Name - CIM property that maps to SNMP OID variable
# Column 4 - ASN.1 type - SNMP datatype that dictates how data is mapped
#            to/from SNMP requests.  Supported types are: SnmpString, SnmpOid,
```

EXAMPLE 1 An Example of a `050SUNWwbcou.map` File *(Continued)*

```
#            SnmpTimeticks, SnmpCounter, SnmpInt, SnmpGauge, SnmpIpAddress,
#            SnmpOpaque)
# Column 5 and greater are ignored
#
Version 1.0

1.3.6.1.2.1.1.1.0 Solaris_ComputerSystem Description SnmpString
1.3.6.1.2.1.1.3.0 Solaris_OperatingSystem LastBootUpTime SnmpTimeticks
1.3.6.1.2.1.1.4.0 Solaris_ComputerSystem PrimaryOwnerContact SnmpString
1.3.6.1.2.1.1.5.0 Solaris_ComputerSystem Name SnmpString

1.3.6.1.2.1.25.1.5.0 Solaris_OperatingSystem NumberOfUsers SnmpGauge
1.3.6.1.2.1.25.1.6.0 Solaris_OperatingSystem NumberOfProcesses SnmpGauge
1.3.6.1.2.1.25.1.7.0 Solaris_OperatingSystem MaxNumberOfProcesses SnmpGauge
1.3.6.1.2.1.25.1.2.0 Solaris_OperatingSystem LocalDateTime SnmpString
```

Each definition of an object in this file contains an OID, its corresponding CIM class name, its corresponding CIM property name, and its corresponding ASN.1 type. Each of these elements is separated by a space character.

Files

`/var/sadm/wbem/snmp/map/050SUNWwbcou.map` — The SNMP Adapter Subagent for WBEM MIB–2 mapping file that Sun Microsystems provides contains SNMP Management Information Base (MIB) definitions for the CIM instrumentation that SNMP manages.

Exit Status

The following exit values are returned:

`0` Successful completion.

`1` An error occurred.

Attributes

See `attributes(5)` for descriptions of the following attributes.

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWwbco
CSI	Enabled
Interface Stability	Committed
MT-Level	Safe

See Also snmpdx(1M), attributes(5)

Name snoop – capture and inspect network packets

Synopsis

```
snoop [-aqrCDINPSvV] [-t [r | a | d]] [-c maxcount]
     [-d device] [-I IP_interface] [-i filename] [-n filename]
     [-o filename] [-p first [, last]] [-s snaplen]
     [-x offset [, length]] [expression]
```

Description From a datalink or IP interface, `snoop` captures packets and displays their contents. If the datalink or IP interface is not specified, `snoop` will pick a datalink to use, giving priority to datalinks that have been plumbed for IP traffic. `snoop` uses the `pfmod(7M)` and `bufmod(7M)` STREAMS modules to provide efficient capture of packets from the network. Captured packets can be displayed as they are received or saved to a file (which is *RFC 1761*-compliant) for later inspection.

`snoop` can display packets in a single-line summary form or in verbose multi-line forms. In summary form, with the exception of certain VLAN packets, only the data pertaining to the highest level protocol is displayed. If a packet has a VLAN header and its VLAN ID is non-zero, then `snoop` will show that the packet is VLAN tagged. For example, an NFS packet will have only NFS information displayed. Except for VLAN information under the condition just described, the underlying RPC, UDP, IP, and Ethernet frame information is suppressed, but can be displayed if either of the verbose options are chosen.

In the absence of a name service, such as LDAP or NIS, `snoop` displays host names as numeric IP addresses.

`snoop` requires an interactive interface.

Note – The `snoop` command is Obsolete. It is recommended that you use Wireshark or its non-GUI version, `Tshark` instead. These programs are available in the following packages:

- `pkg://solaris/diagnostic/wireshark`
- `pkg://solaris/diagnostic/wireshark/tshark`

These packages are available in the Solaris package repository.

Options

`-C`
List the code generated from the filter expression for either the kernel packet filter, or `snoop`'s own filter.

`-D`
Display number of packets dropped during capture on the summary line.

`-N`
Create an IP address-to-name file from a capture file. This must be set together with the `-i` *filename* option that names a capture file. The address-to-name file has the same name as the capture file with `.names` appended. This file records the IP address to hostname mapping at the capture site and increases the portability of the capture file. Generate a `.names` file if the capture file is to be analyzed elsewhere. Packets are not displayed when this flag is used.

`-I` *IP_interface*
Capture IP packets from the network using the IP interface specified by *IP_interface*, for example, `lo0`. The `ipadm(1M)` command can be used to list available IP interfaces. The `-I` and `-d` options are mutually exclusive.

`-P`
Capture packets in non-promiscuous mode. Only broadcast, multicast, or packets addressed to the host machine will be seen.

`-S`
Display size of the entire link layer frame in bytes on the summary line.

`-V`
Verbose summary mode. This is halfway between summary mode and verbose mode in degree of verbosity. Instead of displaying just the summary line for the highest level protocol in a packet, it displays a summary line for each protocol layer in the packet. For instance, for an NFS packet it will display a line each for the ETHER, IP, UDP, RPC and NFS layers. Verbose summary mode output may be easily piped through `grep` to extract packets of interest. For example, to view only RPC summary lines, enter the following: `example#` **`snoop -i rpc.cap -V | grep RPC`**

`-a`
Listen to packets on `/dev/audio` (warning: can be noisy).

`-c` *maxcount*
Quit after capturing *maxcount* packets. Otherwise keep capturing until there is no disk space left or until interrupted with Control-C.

`-d` *datalink*
Capture link-layer packets from the network using the DLPI datalink specified by *datalink*, for example, `bge0` or `net0`. The `dladm(1M)` `show-link` subcommand can be used to list available datalinks. The `-d` and `-I` options are mutually exclusive.

`-i` *filename*
Display packets previously captured in *filename*. Without this option, `snoop` reads packets from the first network interface. If a *filename*`.names` file is present, it is automatically loaded into the `snoop` IP address-to-name mapping table (See `-N` flag).

`-n` *filename*
Use *filename* as an IP address-to-name mapping table. This file must have the same format as the `/etc/hosts` file (IP address followed by the hostname).

`-o` *filename*
Save captured packets in *filename* as they are captured. (This *filename* is referred to as the "capture file".) The format of the capture file is RFC 1761–compliant. During packet capture, a count of the number of packets saved in the file is displayed. If you wish just to count packets without saving to a file, name the file `/dev/null`.

`-p` *first* [, `last`]
Select one or more packets to be displayed from a capture file. The *first* packet in the file is packet number 1.

`-q`
When capturing network packets into a file, do not display the packet count. This can improve packet capturing performance.

`-r`
Do not resolve the IP address to the symbolic name. This prevents `snoop` from generating network traffic while capturing and displaying packets. However, if the `-n` option is used, and an address is found in the mapping file, its corresponding name will be used.

`-s` *snaplen*
Truncate each packet after *snaplen* bytes. Usually the whole packet is captured. This option is useful if only certain packet header information is required. The packet truncation is done within the kernel giving better utilization of the streams packet buffer. This means less chance of dropped packets due to buffer overflow during periods of high traffic. It also saves disk space when capturing large traces to a capture file. To capture only IP headers (no options) use a *snaplen* of 34. For UDP use 42, and for TCP use 54. You can capture RPC headers with a *snaplen* of 80 bytes. NFS headers can be captured in 120 bytes.

`-t` [`r` | `a` | `d`]
Time-stamp presentation. Time-stamps are accurate to within 4 microseconds. The default is for times to be presented in `d` (delta) format (the time since receiving the previous packet). Option `a` (absolute) gives wall-clock time. Option `r` (relative) gives time relative to the first packet displayed. This can be used with the `-p` option to display time relative to any selected packet.

`-v`
Verbose mode. Print packet headers in lots of detail. This display consumes many lines per packet and should be used only on selected packets.

`-x`*offset* [, *length*]
Display packet data in hexadecimal and ASCII format. The *offset* and *length* values select a portion of the packet to be displayed. To display the whole packet, use an *offset* of 0. If a *length* value is not provided, the rest of the packet is displayed.

Operands *expression*
Select packets either from the network or from a capture file. Only packets for which the expression is true will be selected. If no expression is provided it is assumed to be true.

Given a filter expression, `snoop` generates code for either the kernel packet filter or for its own internal filter. If capturing packets with the network interface, code for the kernel packet filter is generated. This filter is implemented as a streams module, upstream of the buffer module. The buffer module accumulates packets until it becomes full and passes the packets on to `snoop`. The kernel packet filter is very efficient, since it rejects unwanted packets in the kernel before they reach the packet buffer or `snoop`. The kernel packet filter

has some limitations in its implementation; it is possible to construct filter expressions that it cannot handle. In this event, `snoop` tries to split the filter and do as much filtering in the kernel as possible. The remaining filtering is done by the packet filter for `snoop`. The `-C` flag can be used to view generated code for either the packet filter for the kernel or the packet filter for `snoop`. If packets are read from a capture file using the `-i` *filename* option, only the packet filter for `snoop` is used.

A filter *expression* consists of a series of one or more boolean primitives that may be combined with boolean operators (`AND`, `OR`, and `NOT`). Normal precedence rules for boolean operators apply. Order of evaluation of these operators may be controlled with parentheses. Since parentheses and other filter expression characters are known to the shell, it is often necessary to enclose the filter expression in quotes. Refer to Example 2 for information about setting up more efficient filters.

The primitives are:

`host` *hostname*
: True if the source or destination address is that of `hostname`. The *hostname* argument may be a literal address. The keyword `host` may be omitted if the name does not conflict with the name of another expression primitive. For example, `pinky` selects packets transmitted to or received from the host `pinky`, whereas `pinky and dinky` selects packets exchanged between hosts `pinky AND dinky`.

  The type of address used depends on the primitive which precedes the `host` primitive. The possible qualifiers are `inet`, `inet6`, `ether`, or none. These three primitives are discussed below. Having none of the primitives present is equivalent to "inet host hostname or inet6 host hostname". In other words, snoop tries to filter on all IP addresses associated with hostname.

inet or *inet6*
: A qualifier that modifies the `host` primitive that follows. If it is *inet*, then `snoop` tries to filter on all IPv4 addresses returned from a name lookup. If it is *inet6*, `snoop` tries to filter on all IPv6 addresses returned from a name lookup.

ipaddr, *atalkaddr*, or *etheraddr*
: Literal addresses, IP dotted, AppleTalk dotted, and Ethernet colon are recognized. For example,

  - "`172.16.40.13`" matches all packets with that IP
  - "`2::9255:a00:20ff:fe73:6e35`" matches all packets with that IPv6 address as source or destination;
  - "`65281.13`" matches all packets with that AppleTalk address;
  - "`8:0:20:f:b1:51`" matches all packets with the Ethernet address as source or destination.

An Ethernet address beginning with a letter is interpreted as a hostname. To avoid this, prepend a zero when specifying the address. For example, if the Ethernet address is `aa:0:45:23:52:44`, then specify it by add a leading zero to make it `0aa:0:45:23:52:44`.

`from` or `src`
A qualifier that modifies the following `host`, `net`, *ipaddr*, *atalkaddr*, *etheraddr*, `port` or `rpc` primitive to match just the source address, port, or RPC reply.

`to` or `dst`
A qualifier that modifies the following `host`, `net`, *ipaddr*, *atalkaddr*, *etheraddr*, `port` or `rpc` primitive to match just the destination address, port, or RPC call.

`ether`
A qualifier that modifies the following `host` primitive to resolve a name to an Ethernet address. Normally, IP address matching is performed. This option is not supported on media such as IPoIB (IP over InfiniBand).

`ethertype` *number*
True if the Ethernet type field has value *number*. If *number* is not 0x8100 (VLAN) and the packet is VLAN tagged, then the expression will match the encapsulated Ethernet type.

`ip`, `ip6`, `arp`, `rarp`, `pppoed`, `pppoes`
True if the packet is of the appropriate ethertype.

`vlan`
True if the packet has `ethertype` VLAN and the VLAN ID is not zero.

`vlan-id` *id*
True for packets of ethertype VLAN with the id *id*.

`pppoe`
True if the ethertype of the packet is either `pppoed` or `pppoes`.

`broadcast`
True if the packet is a broadcast packet. Equivalent to `ether[2:4] = 0xffffffff` for Ethernet. This option is not supported on media such as IPoIB (IP over InfiniBand).

`multicast`
True if the packet is a multicast packet. Equivalent to "`ether[0] & 1 = 1`" on Ethernet. This option is not supported on media such as IPoIB (IP over InfiniBand).

`bootp`, `dhcp`
True if the packet is an unfragmented IPv4 UDP packet with either a source port of `BOOTPS (67)` and a destination port of `BOOTPC (68)`, or a source port of `BOOTPC (68)` and a destination of `BOOTPS (67)`.

`dhcp6`
True if the packet is an unfragmented IPv6 UDP packet with either a source port of `DHCPV6-SERVER` (547) and a destination port of `DHCPV6-CLIENT` (546), or a source port of `DHCPV6-CLIENT` (546) and a destination of `DHCPV6-SERVER` (547).

`apple`
True if the packet is an Apple Ethertalk packet. Equivalent to "`ethertype 0x809b or ethertype 0x80f3`".

`decnet`
True if the packet is a DECNET packet.

`greater` *length*
True if the packet is longer than *length*.

`less` *length*
True if the packet is shorter than *length*.

`udp`, `tcp`, `icmp`, `icmp6`, `ah`, `esp`
True if the IP or IPv6 protocol is of the appropriate type.

`net` *net*
True if either the IP source or destination address has a network number of *net*. The `from` or `to` qualifier may be used to select packets for which the network number occurs only in the source or destination address.

`port` *port*
True if either the source or destination port is *port*. The *port* may be either a port number or name from `/etc/services`. The `tcp` or `udp` primitives may be used to select TCP or UDP ports only. The `from` or `to` qualifier may be used to select packets for which the *port* occurs only as the source or destination.

`rpc` *prog* [, *vers* [, `proc`]]
True if the packet is an RPC call or reply packet for the protocol identified by *prog*. The *prog* may be either the name of an RPC protocol from `/etc/rpc` or a program number. The *vers* and `proc` may be used to further qualify the program *version* and *procedure* number, for example, `rpc nfs,2,0` selects all calls and replies for the NFS null procedure. The `to` or `from` qualifier may be used to select either call or reply packets only.

`zone` *zoneid*
True if *zoneid* matches either the source or destination *zoneid* of a packet received on an `ipnet` device.

`ldap`
True if the packet is an LDAP packet on port 389.

`gateway` *host*
True if the packet used *host* as a gateway, that is, the Ethernet source or destination address was for *host* but not the IP address. Equivalent to "`ether host` *host* and not host *host*".

`nofrag`
True if the packet is unfragmented or is the first in a series of IP fragments. Equivalent to `ip[6:2] & 0x1fff = 0`.

expr relop expr

True if the relation holds, where *relop* is one of >, <, >=, <=, =, `!=`, and `expr` is an arithmetic expression composed of numbers, packet field selectors, the `length` primitive, and arithmetic operators +, –, *, &, |, ^, and %. The arithmetic operators within `expr` are evaluated before the relational operator and normal precedence rules apply between the arithmetic operators, such as multiplication before addition. Parentheses may be used to control the order of evaluation. To use the value of a field in the packet use the following syntax:

base`[expr [: size ] ]`

where `expr` evaluates the value of an offset into the packet from a *base* offset which may be `ether`, `ip`, `ip6`, `udp`, `tcp`, or `icmp`. The `size` value specifies the size of the field. If not given, 1 is assumed. Other legal values are 2 and 4. For example,

```
ether[0] & 1 = 1
```

is equivalent to `multicast`

```
ether[2:4] = 0xffffffff
```

is equivalent to `broadcast`.

```
ip[ip[0] & 0xf * 4 : 2] = 2049
```

is equivalent to `udp[0:2] = 2049`

```
ip[0] & 0xf > 5
```

selects IP packets with options.

```
ip[6:2] & 0x1fff = 0
```

eliminates IP fragments.

```
udp and ip[6:2]&0x1fff = 0 and udp[6:2] != 0
```

finds all packets with UDP checksums.

The `length` primitive may be used to obtain the length of the packet. For instance "`length > 60`" is equivalent to "`greater 60`", and "`ether[length – 1]`" obtains the value of the last byte in a packet.

`and`

Perform a logical AND operation between two boolean values. The AND operation is implied by the juxtaposition of two boolean expressions, for example "`dinky pinky`" is the same as "`dinky AND pinky`".

`or` or `,`

Perform a logical OR operation between two boolean values. A comma may be used instead, for example, "`dinky,pinky`" is the same as "`dinky OR pinky`".

`not` or `!`
Perform a logical NOT operation on the following boolean value. This operator is evaluated before AND or OR.

`slp`
True if the packet is an SLP packet.

`sctp`
True if the packet is an SCTP packet.

`ospf`
True if the packet is an OSPF packet.

Examples

EXAMPLE 1 Using the `snoop` Command

Capture all packets and display them as they are received:

```
example# snoop
```

Capture packets with host `funky` as either the source or destination and display them as they are received:

```
example# snoop funky
```

Capture packets between `funky` and `pinky` and save them to a file. Then inspect the packets using times (in seconds) relative to the first captured packet:

```
example# snoop -o cap funky pinky
example# snoop -i cap -t r | more
```

To look at selected packets in another capture file:

```
example# snoop -i pkts -p 99,108
 99   0.0027   boutique -> sunroof     NFS C GETATTR FH=8E6
100   0.0046   sunroof -> boutique     NFS R GETATTR OK
101   0.0080   boutique -> sunroof NFS C RENAME FH=8E6C MTra00192 to .nfs08
102   0.0102   marmot -> viper        NFS C LOOKUP FH=561E screen.r.13.i386
103   0.0072   viper -> marmot       NFS R LOOKUP No such file or directory
104   0.0085   bugbomb -> sunroof    RLOGIN C PORT=1023 h
105   0.0005   kandinsky -> sparky    RSTAT C Get Statistics
106   0.0004   beeblebrox -> sunroof  NFS C GETATTR FH=0307
107   0.0021   sparky -> kandinsky    RSTAT R
108   0.0073   office -> jeremiah      NFS C READ FH=2584 at 40960 for 8192
```

To look at packet 101 in more detail:

```
example# snoop -i pkts -v -p101
ETHER:  ----- Ether Header -----
ETHER:
ETHER:  Packet 101 arrived at 16:09:53.59
ETHER:  Packet size = 210 bytes
ETHER:  Destination = 8:0:20:1:3d:94, Sun
```

EXAMPLE 1 Using the snoop Command *(Continued)*

```
ETHER:  Source      = 8:0:69:1:5f:e,  Silicon Graphics
ETHER:  Ethertype = 0800 (IP)
ETHER:
IP:   ----- IP Header -----
IP:
IP:   Version = 4, header length = 20 bytes
IP:   Type of service = 00
IP:         ..0. .... = routine
IP:         ...0 .... = normal delay
IP:         .... 0... = normal throughput
IP:         .... .0.. = normal reliability
IP:   Total length = 196 bytes
IP:   Identification 19846
IP:   Flags = 0X
IP:   .0.. .... = may fragment
IP:   ..0. .... = more fragments
IP:   Fragment offset = 0 bytes
IP:   Time to live = 255 seconds/hops
IP:   Protocol = 17 (UDP)
IP:   Header checksum = 18DC
IP:   Source address = 172.16.40.222, boutique
IP:   Destination address = 172.16.40.200, sunroof
IP:
UDP:  ----- UDP Header -----
UDP:
UDP:  Source port = 1023
UDP:  Destination port = 2049 (Sun RPC)
UDP:  Length = 176
UDP:  Checksum = 0
UDP:
RPC:  ----- SUN RPC Header -----
RPC:
RPC:  Transaction id = 665905
RPC:  Type = 0 (Call)
RPC:  RPC version = 2
RPC:  Program = 100003 (NFS), version = 2, procedure = 1
RPC:  Credentials: Flavor = 1 (Unix), len = 32 bytes
RPC:     Time = 06-Mar-90 07:26:58
RPC:     Hostname = boutique
RPC:     Uid = 0, Gid = 1
RPC:     Groups = 1
RPC:  Verifier   : Flavor = 0 (None), len = 0 bytes
RPC:
NFS:  ----- SUN NFS -----
NFS:
NFS:  Proc = 11 (Rename)
```

EXAMPLE 1 Using the snoop Command *(Continued)*

```
NFS:  File handle = 0000164300000001000800000305A1C47
NFS:                597A0000000800002046314AFC450000
NFS:  File name = MTra00192
NFS:  File handle = 0000164300000001000800000305A1C47
NFS:                597A0000000800002046314AFC450000
NFS:  File name = .nfs08
NFS:
```

To view just the NFS packets between `sunroof` and `boutique`:

```
example# snoop -i pkts rpc nfs and sunroof and boutique
1   0.0000   boutique -> sunroof    NFS C GETATTR FH=8E6C
2   0.0046    sunroof -> boutique   NFS R GETATTR OK
3   0.0080   boutique -> sunroof   NFS C RENAME FH=8E6C MTra00192 to .nfs08
```

To save these packets to a new capture file:

```
example# snoop -i pkts -o pkts.nfs rpc nfs sunroof boutique
```

To view encapsulated packets, there will be an indicator of encapsulation:

```
example# snoop ip-in-ip
sunroof -> boutique ICMP Echo request    (1 encap)
```

If -V is used on an encapsulated packet:

```
example# snoop -V ip-in-ip
sunroof -> boutique  ETHER Type=0800 (IP), size = 118 bytes
sunroof -> boutique  IP D=172.16.40.222 S=172.16.40.200 LEN=104, ID=27497
sunroof -> boutique  IP  D=10.1.1.2 S=10.1.1.1 LEN=84, ID=27497
sunroof -> boutique  ICMP Echo request
```

EXAMPLE 2 Setting Up A More Efficient Filter

To set up a more efficient filter, the following filters should be used toward the end of the expression, so that the first part of the expression can be set up in the kernel: `greater`, `less`, `port`, `rpc`, `nofrag`, and `relop`. The presence of `OR` makes it difficult to split the filtering when using these primitives that cannot be set in the kernel. Instead, use parentheses to enforce the primitives that should be `OR`'d.

To capture packets between `funky` and `pinky` of type `tcp` or `udp` on `port` 80:

```
example# snoop funky and pinky and port 80 and tcp or udp
```

Since the primitive `port` cannot be handled by the kernel filter, and there is also an `OR` in the expression, a more efficient way to filter is to move the `OR` to the end of the expression and to use parentheses to enforce the `OR` between `tcp` and `udp`:

```
example# snoop funky and pinky and (tcp or udp) and port 80
```

Exit Status

`0` Successful completion.

`1` An error occurred.

Files

`/dev/audio`	Symbolic link to the system's primary audio device.
`/dev/null`	The null file.
`/etc/hosts`	Host name database.
`/etc/rpc`	RPC program number data base.
`/etc/services`	Internet services and aliases.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	service/network/network-clients
Interface Stability	Obsolete

See Also

`dladm(1M)`, `ipadm(1M)`, `netstat(1M)`, `hosts(4)`, `rpc(4)`, `services(4)`, `attributes(5)`, `audio(7I)`, `ipnet(7D)`, `bufmod(7M)`, `dlpi(7P)`, `pfmod(7M)`

Callaghan, B. and Gilligan, R. *RFC 1761, Snoop Version 2 Packet Capture File Format*. Network Working Group. February 1995.

Warnings

The processing overhead is much higher for real-time packet interpretation. Consequently, the packet drop count may be higher. For more reliable capture, output raw packets to a file using the `-o` option and analyze the packets offline.

Unfiltered packet capture imposes a heavy processing load on the host computer, particularly if the captured packets are interpreted real-time. This processing load further increases if verbose options are used. Since heavy use of `snoop` may deny computing resources to other processes, it should not be used on production servers. Heavy use of `snoop` should be restricted to a dedicated computer.

`snoop` does not reassemble IP fragments. Interpretation of higher level protocol halts at the end of the first IP fragment.

`snoop` may generate extra packets as a side-effect of its use. For example it may use a network name service (NIS) to convert IP addresses to host names for display. Capturing into a file for later display can be used to postpone the address-to-name mapping until after the capture session is complete. Capturing into an NFS-mounted file may also generate extra packets.

Setting the `snaplen` (`-s` option) to small values may remove header information that is needed to interpret higher level protocols. The exact cutoff value depends on the network and

protocols being used. For NFS Version 2 traffic using UDP on 10 Mb/s Ethernet, do not set `snaplen` less than 150 bytes. For NFS Version 3 traffic using TCP on 100 Mb/s Ethernet, `snaplen` should be 250 bytes or more.

`snoop` requires information from an RPC request to fully interpret an RPC reply. If an RPC reply in a capture file or packet range does not have a request preceding it, then only the RPC reply header will be displayed.

Name soconfig – configure transport providers for use by sockets

Synopsis

```
/usr/sbin/soconfig -d dir

/usr/sbin/soconfig -f file

/usr/sbin/soconfig -F name [[-h hint] [-m modname]
     -t family:type:protocol[,...]]

/usr/sbin/soconfig family type protocol [module | path]

/usr/sbin/soconfig -l [-np]
```

Description The `soconfig` utility configures the transport provider driver for use with sockets. It specifies how the family, type, and protocol parameters in the `socket(3SOCKET)` call are mapped to the name of a transport provider such as `/dev/tcp`. This utility can be used to add an additional mapping or remove a previous mapping.

The `soconfig` utility is also used to configure socket filters.

The `init(1M)` utility uses `soconfig` with the `sock2path.d(4)` file during the booting sequence.

Options The following options are supported:

`-d` *dir*
: Set up the `soconfig` configuration for each driver according to the information stored in the files in *dir*.

`-f` *file*
: Set up the `soconfig` configuration for each driver according to the information stored in *file*. A `soconfig` file consists of lines of at least the first three fields listed below, separated by spaces:

  family type protocol [*module* | *path*]

  These fields are described in the `OPERANDS` section below.

  An example of *file* can be found in the `EXAMPLES` section below.

`-F` *name* [[`-h` *hint*] [`-m` *modname*] `-t` *family*:*type*:*protocol*[,...]]
: Update socket filter configuration. If only the *name* is specified, then the filter is unconfigured. Otherwise, the filter will be configured.

  The following are options associated with the `-F` option.

  `-h` *hint*
  : Placement hint for the filter. If the placement cannot be satisfied, then the configuration request will fail. The hint can be one of the following:

    `top`
    : The filter should be placed on top, closest to the socket.

`bottom`
The filter should be placed on the bottom, closest to the protocol.

`above:`*name*
The filter should be placed above the filter specified by *name*. The filter will lie closer to the socket than *name*.

`below:`*name*
The filter should be placed below the filter specified by *name*. The filter will lie closer to the socket than *name*.

`-m` *modname*
Name of the filter module. The module must be located in the `socketmod` namespace. If unspecified, then the module name is expected to be the same as the filter name.

`-t` *family*:*type*:*protocol*[,...]
Socket tuple for the filter. The fields making up the tuple are described in the `OPERANDS` section. The configuration will fail if any of the socket tuples refer to a socket that uses a transport provider that does not support socket filters. STREAMS-based transport providers are not supported by socket filters.

`-l` [`-np`]
Dump the in-kernel socket configuration table. This option has the following suboptions:

`-n`
Do not convert socket family, type, and protocol to string representation.

`-p`
Display machine-parseable output.

Operands The following operands are supported:

family
The protocol family as listed in the `/usr/include/sys/socket.h` file, expressed as an integer. This file is contained in the `system/header` package, which might not be part of your Solaris distribution.

type
The socket type as listed in the `/usr/include/sys/socket.h` file, expressed as an integer. This file is contained in the `system/header` package, which might not be part of your Solaris distribution.

protocol
The protocol number as specified in the family-specific `include` file, expressed as an integer. For example, for `AF_INET` this number is specified in `/usr/include/netinet/in.h`. An unspecified protocol number is denoted with the value zero.

module | *path*
The module name or path name of a device that corresponds to the transport provider, such as `tcp` or `/dev/tcp`. Modules must reside in `kernel/socketmod`. A device name must begin with `/dev`. If this parameter is specified, the configuration will be added for the specified family, type, and protocol. If this parameter is not specified, the configuration will be removed.

Examples

EXAMPLE 1 Setting Up a Module

The following example sets up a module for family `AF_INET` and type `SOCK_STREAM`:

```
example# soconfig 2 2 0 tcp
```

The following example sets up `/dev/tcp` for family `AF_INET` and type `SOCK_STREAM`:

```
example# soconfig 2 2 0 /dev/tcp
```

The following is a sample file used with the `-f` option. Comment lines begin with a hash mark (`#`):

```
#   Family  Type  Protocol   Module | Path
      2       2       0          tcp
      2       2       6          tcp

      2       1       0          udp
      2       1       17         udp

      1       2       0          /dev/ticotsord
      1       1       0          /dev/ticlts

      2       4       0          icmp
```

EXAMPLE 2 Configuring a Socket Filter

The following command configures a socket filter, `foo`, which has the loadable module named `foomod`. The filter will automatically be attached to all `AF_INET` TCP sockets.

```
example# soconfig -F foo -m foomod -t 2:2:0,2:2:6
```

The following command unconfigures the filter `foo`.

```
example# soconfig -F foo
```

Files

`/etc/sock2path.d`
Directory containing files with mappings from sockets to transport providers.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also `init`(1M), `sock2path.d`(4), `attributes`(5)

Network Interface Guide

Name soladdapp – add an application to the Solstice application registry

Synopsis

```
/usr/snadm/bin/soladdapp [-r registry] -n name -i icon -e executable
    [args]
```

Description `soladdapp` adds an application to the Solstice application registry. After it is added, the application is displayed in the Solstice Launcher main window (see `solstice(1M)`).

Options

`-r` *registry*	Define the full path name of the Solstice registry file.
`-n` *name*	Define the name of the tool to be registered.
`-i` *icon*	Define the full path name of the tool icon.
`-e` *executable*	Define the full path name of the tool.
args	Specify any arguments to use with the tool.

When executed without options, `soladdapp` uses `/opt/SUNWadm/etc/.solstice_registry` (the default registry path).

Return Values

`0`	on success
`1`	on failure
`2`	if the registry is locked
`3`	if the entry is a duplicate.

Examples **EXAMPLE 1** A sample display of the `soladdapp` command.

The following adds an application called Disk Manager to the Solstice application registry for display in the Solstice Launcher main window.

```
# soladdapp -r /opt/SUNWadm/etc/.solstice_registry -n "Disk Manager"
-i /opt/SUNWdsk/etc/diskmgr.xpm -e /opt/SUNWdsk/bin/diskmgr
```

Files

`/opt/SUNWadm/etc/.solstice_registry`	The default registry path.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsadml

See Also `soldelapp(1M)`, `solstice(1M)`, `attributes(5)`

Notes Globally registered applications are used by local and remote users sharing the software in a particular /opt directory. They can be added only using `soladdapp`.

Name soldelapp – remove an application from the Solstice application registry

Synopsis

```
/usr/snadm/bin/soldelapp [-r registry] -n name
```

Description `soldelapp` removes an application from the Solstice application registry. After removal, the application is no longer displayed in the Solstice Launcher main window (see `solstice(1M)`).

Options

`-r` *registry* Define the full path name of the Solstice registry file.

`-n` *name* Define the name of the tool to be removed.

When executed without options, `soldelapp` uses `/opt/SUNWadm/etc/.solstice_registry` (the default registry path).

Return Values

`0` on success

`1` on failure

`2` if the registry is locked

`3` if *name* is not found in the registry

`4` if the named registry or default registry is not found

Examples **EXAMPLE 1** A sample display of the `soldelapp` command.

The following removes an application called Disk Manager from the Solstice application registry and the Solstice Launcher main window.

```
# soldelapp -r /opt/SUNWadm/etc/.solstice_registry -n "Disk Manager"
```

Files `/opt/SUNWadm/etc/.solstice_registry` The default registry file.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsadml

See Also `soladdapp(1M)`, `solstice(1M)`, `attributes(5)`

Notes Globally registered applications are used by local and remote users sharing the software in a particular `/opt` directory. They can be removed only using `soldelapp`.

Name solstice – access system administration tools with a graphical user interface

Synopsis `/bin/solstice`

Description `solstice` used on a system presents the `Solstice Launcher`, a graphical user interface that provides access to the Solstice AdminSuite product family of system administration tools. The tools that appear in the launcher depend on what Solstice products you installed on your system.

Help is available by using the Help button.

Usage The `Solstice Launcher` allows you to do the following tasks:

Launch applications	Use the `Solstice Launcher` to launch system administration tools.
Register applications	Use the `Solstice Launcher` to add and register applications locally with the launcher.
Remove applications	Use the `Solstice Launcher` to remove locally registered applications.
Customize application properties	Use the `Solstice Launcher` to show, hide, or remove applications in the launcher, reorder the icons, change the launcher window width, modify applications properties, and add applications.

Files `/$HOME/.solstice_registry` Local registry information.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsadml

See Also `soladdapp(1M)`, `soldelapp(1M)`, `attributes(5)`

Notes The `Solstice Launcher` adds or removes local applications that are private to the user (not local to the system) only. The properties of globally registered applications that are used by local and remote users sharing the software from a particular `/opt` directory cannot be modified from the `Solstice Launcher`. To register global applications for use by local and remote users, use the `soladdapp(1M)` command. To remove globally registered applications, use the `soldelapp(1M)` command.

Name sppptun – PPP tunneling driver utility

Synopsis

```
sppptun plumb

sppptun plumb [-s sap] protocol device

sppptun unplumb interface

sppptun query
```

Description The `sppptun` utility is used to configure and query the Solaris PPP tunneling device driver, `/dev/sppptun`. Currently, only PPP over Ethernet (PPPoE) is supported, so the `plumb` and `unplumb` arguments are used to specify Ethernet interfaces that are to be used for PPPoE, and the `query` option lists the plumbed interfaces.

The use of `sppptun` to add interfaces is similar to the use of `ifconfig(1M)` to add interfaces to IP. The plumbing is done once for each interface, preferably at system start-up time, and is not normally manipulated on a running system. If multiple instances of PPP are run over a single interface, they share the plumbing to that interface. Plumbing for each session is not required (and not possible for PPPoE).

The proper way to plumb interfaces for PPPoE is to list the interfaces, one per line, in the `/etc/ppp/pppoe.if` file. If alternate Ethertype (SAP) values are necessary, then include the PPPoE Session and Discovery Stage values as hexadecimal on the same line. The line format is:

interface [*session* [*discovery*]]

The defaults are the Ethertypes specified in RFC 2516, and most users should not need to set these values. See the examples for one possible use.

Usage

`sppptun plumb` — When specified with no additional arguments, the plumb argument lists the protocols that are supported by the utility. These are the strings that are used as the *protocol* argument below.

`sppptun plumb [-s sap] protocol device` — This plumbs a new interface into the driver. The *protocol* parameter is `pppoe` for the PPP-carrying "Session Stage" connection or `pppoed` for the PPPoE "Discovery Stage" connection. Both connections must be present for each Ethernet interface that is to be used for PPPoE. The *device* parameter is the path name of the Ethernet interface to use (use `ifconfig(1M)` to list available devices). If the path begins with `/dev/`, then this portion may be omitted.

The `-s` *sap* option can be specified to use an alternate Ethertype (SAP) for the selected protocol. The *sap* value must be given in

hexadecimal. Some access servers use Ethertypes for PPPoE different from those in RFC 2516. The defaults are 8864 for `pppoe` and 8863 for `pppoed`.

`spppptun unplumb` *interface*
: This removes an existing interface from the driver and terminates any PPP sessions that were using the interface. The *interface* parameter is the name of the interface as reported when the interface was plumbed.

`spppptun query`
: Displays the canonical names of all interfaces plumbed into the `/dev/spppptun` device driver.

Examples

EXAMPLE 1 Setting up to Use PPPoE on hme0

Plumb the `hme0` interface.

```
# spppptun plumb pppoed hme0
hme0:pppoed
# spppptun plumb pppoe hme0
hme0:pppoe
```

Remove the `hme0` interface.

```
# spppptun unplumb hme0:pppoed
# spppptun unplumb hme0:pppoe
```

EXAMPLE 2 Script to Remove All Plumbed Interfaces

```
#!/bin/sh
for intf in ‘spppptun query‘
do
        spppptun unplumb $intf
done
```

EXAMPLE 3 Interoperating with 3COM HomeConnect Dual Link ADSL

```
# dladm show-link
LINK        CLASS    MTU    STATE    OVER
nge0        phys     1500   up       --
# echo nge0 3c13 3c12 > /etc/ppp/pppoe.if
```

Exit Status

The following exit values are returned:

`0` Successful completion.

`1` One or more errors occurred.

Files

`/etc/ppp/pppoe.if`	list of Ethernet interfaces to be plumbed at boot time
`/usr/sbin/sppptun`	executable command
`/dev/sppptun`	Solaris PPP tunneling device driver

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/network/ppp/tunnel

See Also `pppd(1M)`, `pppoec(1M)`, `pppoed(1M)`, `sppptun(7M)`

RFC 2516, Method for Transmitting PPP Over Ethernet (PPPoE), Mamakos et al, February 1999

Name spray – spray packets

Synopsis

```
/usr/sbin/spray [-c count] [-d delay] [-l length]
     [-t nettype] host
```

Description `spray` sends a one-way stream of packets to *host* using RPC, and reports how many were received, as well as the transfer rate. The *host* argument can be either a name or an Internet address.

`spray` is not useful as a networking benchmark, as it uses unreliable connectionless transports, UDP for example. `spray` can report a large number of packets dropped when the drops were caused by `spray` sending packets faster than they can be buffered locally, that is, before the packets get to the network medium.

Options

`-c` *count* Specify how many packets to send. The default value of *count* is the number of packets required to make the total stream size 100000 bytes.

`-d` *delay* Specify how many microseconds to pause between sending each packet. The default is `0`.

`-l` *length* The *length* parameter is the numbers of bytes in the Ethernet packet that holds the RPC call message. Since the data is encoded using XDR, and XDR only deals with 32 bit quantities, not all values of *length* are possible, and `spray` rounds up to the nearest possible value. When *length* is greater than 1514, then the RPC call can no longer be encapsulated in one Ethernet packet, so the *length* field no longer has a simple correspondence to Ethernet packet size. The default value of *length* is `86 bytes`, the size of the RPC and UDP headers.

`-t` *nettype* Specify class of transports. Defaults to `netpath`. See `rpc(3NSL)` for a description of supported classes.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	service/network/network-clients

See Also `rpc(3NSL)`, `attributes(5)`

Name srptadm – administer SRP targets

Synopsis

```
srptadm modify-target [-e, --enable] [-d, --disable]
     [-r, --reset] hca_guid

srptadm list-target [hca_guid]

srptadm modify-defaults [-e, --enable] [-d, --disable]

srptadm list-defaults
```

Description The `srptadm` command manages SCSI RDMA Protocol (SRP) target ports within the SCSI Target Mode Framework described in `stmfadm(1M)` and `libstmf(3LIB)`. This allows SRP initiators to access SCSI Target Mode Framework (STMF) logical units using the SRP protocol.

`srptadm` is implemented as a set of subcommands with options and operands for each subcommand. These subcommands are described in their own section, below.

Sub-commands

`list-defaults`
Lists information about the default properties. This subcommand has no options.

`list-target` [*hca_guid*]
If *hca_guid* is specified, lists the properties of the target HCA. Otherwise, properties are listed for all HCAs.

`modify-defaults` [`-e`, `--enable`] [`-d`, `--disable`]
Modify default parameters.

`-e`, `--enable`
Enable SRP target creation for all HCAs that have not been explicitly disabled with `modify-target`.

`-d`, `--disable`
Disable SRP target creation for all HCAs that have not been explicitly enabled with `modify-target`.

`modify-target` [`-e`, `--enable`] [`-d`, `--disable`] [`-r`, `--reset` *hca_guid*
Sets SRP Target properties for the specified HCA.

`-e`, `--enable`
Enables SRP target creation on this HCA.

`-d`, `--disable`
Disables SRP target creation on this HCA.

`-r`, `--reset`
Clears HCA-specific information and resets to defaults. The SRP Target, if any, associated with this HCA will not be modified as a result of this option until the SRP Target SMF service is restarted.

Operands The list-target and modify-target subcommands have the following operand.

hca_guid
GUID of the InfiniBand Host Channel Adapter (HCA) on this system for which SRP Target Services can be provided. The GUID must be in one of the following forms:

`3BA000100CD18`	Base hex form.
`0003BA000100CD18`	Base hex form with leading zeroes.
`hca:3BA000100CD18`	Form from `cfgadm(1M)`.
`eui.0003BA000100CD18`	EUI form.

Usage If the default state is changed when the SRP service is `online`, the state of existing targets is not changed until the service is restarted.

Changing the target state takes effect immediately if the SRP target service is `online`. Targets set to `disabled` will be offlined and removed; targets set to `enabled` will be immediately created.

Examples **EXAMPLE 1** Listing Default Properties

The following command lists the default SRP Target Service properties.

```
# srptadm list-defaults
```

EXAMPLE 2 Changing Default Behavior

The following command changes the default behavior of the SRP Target service to not create SRP Targets when the service is enabled.

```
# srptadm modify-defaults -d
```

EXAMPLE 3 Listing Properties for Specific HCA

The following command lists SRP Target properties for a specific HCA.

```
# cfgadm | grep hca
hca:3BA000100CD18              IB-HCA       connected    configured   ok
hca:3BA000100D030              IB-HCA       connected    configured   ok

# srptadm list-target hca:3BA000100CD18
Target HCA 3BA000100CD18:
    Enabled             : true
    SRP Target Name     : eui.0003BA000100CD18
    Operational Status  : online
```

EXAMPLE 4 Disabling Services for Specific HCA

The following command disables SRP Target services for a specific HCA.

```
# srptadm modify-target -d 3BA000100CD18
```

Use the `list-target` command to see the changes:

```
# srptadm list-target hca:3BA000100CD18
Target HCA 3BA000100CD18:
    Enabled             : false
    SRP Target Name     : eui.0003BA000100CD18
    Operational Status  : -
```

EXAMPLE 5 Re-enabling Target Services

The following command sequence re-enables SRP Target services and displays the SRP and STMF target properties.

```
# srptadm modify-target -e 3BA000100CD18
# srptadm list-target hca:3BA000100CD18
Target HCA 3BA000100CD18:
    Enabled             : true
    SRP Target Name     : eui.0003BA000100CD18
    Operational Status  : online

# stmfadm list-target -v eui.0003BA000100CD18
Target: eui.0003BA000100CD18
    Operational Status: Online
    Provider Name     : srpt
    Alias             : -
    Protocol          : SRP
    Sessions          : 0
```

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/storage/scsi-rdma/scsi-rdma-target
Interface Stability	Committed

See Also

`cfgadm(1M)`, `stmfadm(1M)`, `libsrpt(3LIB)`, `libstmf(3LIB)`, `attributes(5)`, `srpt(7D)`

Name sshd – secure shell daemon

Synopsis

```
sshd [-deiqtD46] [-b bits] [-f config_file]
     [-g login_grace_time] [-h host_key_file]
     [-h PKCS#11 URI]
     [-p port] [-V client_protocol_id]
```

Description The `sshd` (Secure Shell daemon) is the daemon program for `ssh(1)`. Together these programs replace `rlogin` and `rsh`, and provide secure encrypted communications between two untrusted hosts over an insecure network. The programs are intended to be as easy to install and use as possible.

`sshd` is the daemon that listens for connections from clients. It forks a new daemon for each incoming connection. The forked daemons handle key exchange, encryption, authentication, command execution, and data exchange.

This implementation of `sshd` supports SSH protocol version 2 only.

SSH Protocol Version 1

This protocol version is no longer supported in the daemon program. It is supported only in the client. See `ssh(1)` manual page for more information.

SSH Protocol Version 2

Each host has a host-specific DSA/RSA key used to identify the host. Forward security is provided through a Diffie-Hellman key agreement. This key agreement results in a shared session key. The rest of the session is encrypted using a symmetric cipher, currently 128/192/256-bit AES CBC or CTR, 128/256-bit ARCFOUR, Blowfish, or 3DES.. The client selects the encryption algorithm to use from those offered by the server. Additionally, session integrity is provided through a cryptographic message authentication code (`hmac-sha1`, `hmac-sha2-256`, `hmac-sha2-256-96`, `hmac-sha2-512`, `hmac-sha2-512-96`, or `hmac-md5`).

Protocol version 2 provides a public key based user authentication method (PubKeyAuthentication) GSS–API based user authentication, conventional password authentication, and a generic prompt/reply protocol for password-based authentication.

Command Execution and Data Forwarding

If the client successfully authenticates itself, a dialog for preparing the session is entered. At this time the client can request things like allocating a pseudo-tty, forwarding X11 connections, forwarding TCP/IP connections, or forwarding the authentication agent connection over the secure channel.

Finally, the client either requests a shell or execution of a command. The sides then enter session mode. In this mode, either side may send data at any time, and such data is forwarded to/from the shell or command on the server side, and the user terminal on the client side.

When the user program terminates and all forwarded X11 and other connections have been closed, the server sends command exit status to the client, and both sides exit.

`sshd` can be configured using command-line options or the configuration file `/etc/ssh/ssh_config`, described in `ssh_config(4)`. Command-line options override values specified in the configuration file.

`sshd` rereads its configuration file when it receives a hangup signal, `SIGHUP`, by executing itself with the name it was started as, that is, `/usr/lib/ssh/sshd`.

Host Access Control

The `sshd` daemon uses TCP Wrappers to restrict access to hosts. It uses the service name of `sshd` for `hosts_access()`. For more information on TCP Wrappers see `tcpd(1M)` and `hosts_access(3)` man pages, which are part of the `SUNWsfman` package (they are not SunOS man pages). TCP wrappers binaries, including `libwrap`, are in `security/tcp-wrapper`, a required package for `service/network/ssh`, the package containing `sshd`.

Options

The options for `sshd` are as follows:

`-b` *bits*
Specifies the number of bits in the server key (the default is 768).

`-d`
Debug mode. The server sends verbose debug output to the system log, and does not put itself in the background. The server also will not fork and will only process one connection. This option is only intended for debugging for the server. Multiple `-d` options increase the debugging level. Maximum is 3.

`-e`
When this option is specified, `sshd` will send the output to standard error instead of to the system log.

`-f` *configuration_file*
Specifies the name of the configuration file. The default is `/etc/ssh/sshd_config`. `sshd` refuses to start if there is no configuration file.

`-g` *login_grace_time*
Gives the grace time for clients to authenticate themselves (the default is 300 seconds). If the client fails to authenticate the user within this number of seconds, the server disconnects and exits. A value of zero indicates no limit.

`-h` *host_key_file*
Specifies a file from which a host key is read. This option must be given if `sshd` is not run as root (as the normal host key files are normally not readable by anyone but root). The default is `/etc/ssh/ssh_host_key` for protocol version 1, and `/etc/ssh/ssh_host_rsa_key` and `/etc/ssh/ssh_host_dsa_key` for protocol version 2. It is possible to have multiple host key files for the different protocol versions and host key algorithms.

`-h` *PKCS#11 URI*
Instead of working with a host key file, work with a certificate and a private key stored in the PKCS#11 token. See also `Using X.509 Certificates` section below.

`-i`
Specifies that `sshd` is being run from `inetd`. `sshd` is normally not run from `inetd` because it needs to generate the server key before it can respond to the client, and this may take tens of

seconds. Clients would have to wait too long if the key was regenerated every time. However, with small key sizes (for example, 512) using `sshd` from `inetd` may be reasonable.

`-o` *option*
Can be used to specify options in the format used in the configuration file. This is useful for specifying options for which there are no separate command-line flags.

`-p` *port*
Specifies the port on which the server listens for connections (the default is 22).

`-q`
Quiet mode. Nothing is sent to the system log. Normally the beginning, authentication, and termination of each connection is logged.

`-t`
Test mode. Check only the validity of the configuration file and the sanity of the keys. This is useful for updating sshd reliably as configuration options might change.

`-D`
When this option is specified `sshd` does not detach and does not become a daemon. This allows easy monitoring of `sshd`.

`-4`
Forces `sshd` to use IPv4 addresses only.

`-6`
Forces `sshd` to use IPv6 addresses only.

Extended Description

`authorized_keys` File Format

The `$HOME/.ssh/authorized_keys` file lists the public keys that are permitted for public key authentication (`PubkeyAuthentication`) in protocol version 2. The `AuthorizedKeysFile` configuration option can be used to specify an alternative file.

Each line of the file contains one key (empty lines and lines starting with a hash mark [#] are ignored as comments).

For each RSA key for protocol version 1, the file consists of the following space-separated fields:

options bits exponent modulus comment

For the public key for protocol version 2, the file consists of the following space-separated fields:

options key-type base64-encoding-key comment

For protocol version 2, *key-type* is one of `ssh-rsa` or `ssh-dsa`.

The *options* field is optional; its presence is determined by whether the line starts with a number. (The *options* field never starts with a number.) The bits, exponent, and modulus fields give the RSA key; the comment field is a convenient place for you to identify the key.

Lines in this file are usually several hundred bytes long (because of the size of the key modulus). You will find it very inconvenient to type them in; instead, copy the public key file and edit it.

Permissions of this file must be set so that it is not world or group writable. See the `StrictModes` option of `sshd_config(4)`.

The options (if present) consist of comma-separated option specifications. No spaces are permitted, except within double quotes. The following option specifications are supported:

`from="`*pattern-list*`"`
: Specifies that, in addition to public key authentication, the canonical name of the remote host must be present in the comma-separated list of patterns (‘`*`' and ‘`?`' serve as wildcards). The list can also contain negated patterns by prefixing the patterns with ‘`!`'. If the canonical host name matches a negated pattern, the key is not accepted.

  The purpose of this option is to give you the option of increasing security: public key authentication by itself does not trust the network or name servers or anything but the key. However, if someone manages to steal the key, possession of the key would permit the intruder to log in from anywhere in the world. This option makes using a stolen key more difficult, because name servers and routers would have to be compromised, in addition to just the key.

`command="`*command*`"`
: Specifies that the *command* is executed whenever this key is used for authentication. The command supplied by the user (if any) is ignored. The command is run on a `pty` if the client requests a `pty`; otherwise it is run without a `tty`. If an 8-bit clean channel is required, one must not request a `pty` or should specify `no-pty`. You can include a quote in the command by escaping it with a backslash. This option might be useful to restrict certain public keys from performing a specific operation. An example is a key that permits remote backups but nothing else. Note that the client can specify TCP/IP and/or X11 forwarding unless they are explicitly prohibited from doing so. Also note that this option applies to shell, command, or subsystem execution.

`environment="`*NAME=value*`"`
: Specifies that the string *NAME=value* is to be added to the environment when logging in using this key. Environment variables set this way override other default environment values. Multiple options of this type are permitted. Environment processing is disabled by default and is controlled via the `PermitUserEnvironment` option.

`no-port-forwarding`
: Forbids TCP/IP forwarding when this key is used for authentication. Any port forward requests by the client will return an error. This might be used, for example, in connection with the `command` option.

`no-X11-forwarding`
Forbids X11 forwarding when this key is used for authentication. Any X11 forward requests by the client will return an error.

`no-agent-forwarding`
Forbids authentication agent forwarding when this key is used for authentication.

`no-pty`
Prevents `tty` allocation (a request to allocate a `pty` will fail).

`permitopen="`*host*`:`*port*`"`
Limit local `ssh -L` port forwarding such that it can connect only to the specified host and port. IPv6 addresses can be specified with an alternative syntax: *host*/*port*. You can invoke multiple `permitopen` options, with each instance separated by a comma. No pattern matching is performed on the specified hostnames. They must be literal domains or addresses.

`ssh_known_hosts` File Format

The `/etc/ssh/ssh_known_hosts` and `$HOME/.ssh/known_hosts` files contain host public keys for all known hosts. The global file should be prepared by the administrator (optional), and the per-user file is maintained automatically: whenever the user connects from an unknown host its key is added to the per-user file.

For the RSA key for protocol version 1, these files consist of the following space-separated fields:

hostnames bits exponent modulus comment

For the public key for protocol version 2, these files consist of the following space-separated fields:

hostnames key-type base64-encoding-key comment

For protocol version 2, *key-type* is one of `ssh-rsa` or `ssh-dsa`.

Hostnames is a comma-separated list of patterns (`*` and `?` act as wildcards); each pattern in turn is matched against the canonical host name (when authenticating a client) or against the user-supplied name (when authenticating a server). A pattern can also be preceded by `!` to indicate negation: if the host name matches a negated pattern, it is not accepted (by that line) even if it matched another pattern on the line.

Alternately, hostnames can be stored in a hashed form, which hides host names and addresses should the file's contents be disclosed. Hashed hostnames start with a vertical bar (`|`) character. Only one hashed hostname can appear on a single line and none of the above negation or wildcard operators may be applied.

Bits, exponent, and modulus are taken directly from the RSA host key; they can be obtained, for example, from `/etc/ssh/ssh_host_rsa_key.pub`. The optional comment field continues to the end of the line, and is not used.

Lines starting with a hash mark (`#`) and empty lines are ignored as comments.

When performing host authentication, authentication is accepted if any matching line has the proper key. It is thus permissible (but not recommended) to have several lines or different host keys for the same names. This will inevitably happen when short forms of host names from different domains are put in the file. It is possible that the files contain conflicting information; authentication is accepted if valid information can be found from either file.

The lines in these files are typically hundreds of characters long. You should definitely not type in the host keys by hand. Rather, generate them by a script or by taking `/etc/ssh/ssh_host_rsa_key.pub` and adding the host names at the front.

Environment Variables

`sshd` sets the following environment variables for commands executed by `ssh` users:

`DISPLAY`
: Indicates the location of the X11 server. It is automatically set by `sshd` to point to a value of the form *hostname*:*n*, where *hostname* indicates the host where the shell runs, and *n* is an integer greater than or equal to 1. `ssh` uses this special value to forward X11 connections over the secure channel. Unless you have important reasons to do otherwise, you should not set `DISPLAY` explicitly, as that will render the X11 connection insecure and will require you to manually copy any required authorization cookies.

`HOME`
: Set to the path of the user's home directory.

`LANG`, `LC_ALL`, `LC_COLLATE`, `LC_CTYPE`, `LC_MESSAGES`, `LC_MONETARY`, `LC_NUMERIC`, `LC_TIME`
: A locale setting. The locale defaults to that of `sshd` (usually the system-wide default locale), or is negotiated between the client and server during initial key exchange (as per RFC 4253).

  Following initial key exchange, each of the variables can be overriden in the following sequence:

  1. If a locale setting is set in a client's environment and that client supports "Environment Variable Passing" (see RFC 4254), then the setting will be passed over to the server side.
  2. If the public key authentication method was used to authenticate the server and the `PermitUserEnvironment` variable in `sshd_config(4)` is set to `yes` on the server side, then the setting can be changed through the use of the `environment` option in the client's `AuthorizedKeysFile` file.
  3. The setting can be changed in the client's `~/.ssh/environment` file on the server.

  See `PermitUserEnvironment` in `sshd_config(4)` as to when the `AuthorizedKeysFile` and `~/.ssh/environment` files are processed and used for setting the user environment.

`LOGNAME`
: Synonym for `USER`. Set for compatibility with systems that use this variable.

`MAIL`
: Set to point to the user's mailbox.

`SSH_AUTH_SOCK`
Indicates the path of a `unix-domain` socket used to communicate with the agent.

`SSH_CONNECTION`
Identifies the client and server ends of the connection. The variable contains four space-separated values: client IP address, client port number, server IP address and server port number.

`SSH_CLIENT`
Identifies the client end of the connection. The variable contains three space-separated values: client IP address, client port number, and server port number.

`SSH_TTY`
Set to the name of the `tty` (path to the device) associated with the current shell or command. If the current session has no `tty`, this variable is not set.

`TZ`
Indicates the present timezone, if `TIMEZONE` is set in `/etc/default/login` or if `TZ` was set when the daemon was started.

`HZ`
If set in `/etc/default/login`, the daemon sets it to the same value.

`SHELL`
The user's shell, if `ALTSHELL=YES` in `/etc/default/login`.

`PATH`
Set to the value of `PATH` or `SUPATH` (see `login(1)`) in `/etc/default/login`, or, if not set, to `/usr/bin:/bin`.

`USER`
Set to the name of the user logging in.

Additionally, `sshd` reads `$HOME/.ssh/environment` and adds lines of the format `VARNAME=`*value* to the environment.

Examples

In the following examples, certain lines might wrap due to line length limits in your display. You should nevertheless consider the wrapped line as a single line.

EXAMPLE 1 `authorized_key` File Entries

The following are examples of `authorized_key` file entries for protocol 1:

```
1024 33 12121...312314325 ylo@foo.bar

from="*.niksula.hut.fi,!pc.niksula.hut.fi" 1024 35 23...2334 ylo@niksula

command="dump /home",no-pty,no-port-forwarding 1024 33 23...2323 backup.hut.fi
```

EXAMPLE 2 `authorized_key` File Entries for Protocol 2

The following are examples of `authorized_key` file entries for protocol 2:

```
ssh-rsa AAAAB3NzaC1y.....EU88ovYKg4GfclWGCFYTuw8= ylo@foo.bar
from="*.niksula.hut.fi" ssh-rsa AAAAB3NzaC...uw8= ylo@niksula
command="dump /home",no-pty,no-port-forwarding ssh-rsa AA..8= backup.hut.fi
```

EXAMPLE 3 `ssh_known_hosts` File Entries for Protocol 1

The following are examples of `ssh_known_hosts` file entries for protocol 1:

```
closenet,closenet.hut.fi,...,130.233.208.41 1024 37 159...93 closenet.hut.fi
```

EXAMPLE 4 `ssh_known_hosts` File Entries for Protocol 2

The following are examples of `ssh_known_hosts` file entries for protocol 2:

```
closenet,closenet.hut.fi,...,130.233.208.41 ssh-rsa AA..8= closenet.hut.fi
```

EXAMPLE 5 Using X.509 Public Key Authentication

The following example of user authentication should help you understand how the X.509 public key authentication works. Steps to use X.509 certificates/keys for host authentication are very similar. Note that we use the same token (`softtoken`) for both the TA certificate and the user certificate, which would not normally be the case.

Create a self-signed trusted anchor certificate and a user certificate signed by the trusted anchor private key. Configure the SSH daemon for certificate validation, and run the SSH client with the user certificate generated in the previous step as its user identity.

Replace user name "*PUT-YOUR-USERNAME-HERE*" with an existing (your) user name on the server side.

```
# Generate a trusted anchor certificate/key pair in the
# default token (softtoken).
pktool gencert keystore=pkcs11 label=authority \
    subject="CN=authority" serial=0x01

# Export the trusted anchor certificate (and, after that,
# put it to /etc/ssh/cert directory).
pktool export keystore=pkcs11 outfile=ta.cert objtype=cert
    label=authority outformat=pem

# Generate a user certificate signed by the trusted anchor
# private key and then import the certificate into the
# token.
pktool gencsr keystore=pkcs11 label=user \
    outcsr=user.csr subject="CN=<PUT-YOUR-USERNAME-HERE>"
pktool signcsr keystore=pkcs11 signkey=authority \
    csr=user.csr serial=0x02 issuer="CN=authority" \
```

EXAMPLE 5 Using X.509 Public Key Authentication *(Continued)*

```
     outcert=user.cert
pktool import keystore=pkcs11 infile=user.cert label=user

# Create a new policy file.
kmfcfg create dbfile=/etc/ssh/policy.xml policy=ssh \
    ta-name=search mapper-name=cn

# Start the SSH daemon in a debug mode to test the
# configuration. Use a different port so that you do not
# clash with the already running daemon.
/usr/lib/ssh/sshd -p 2222 -ddd -o \
    -o TrustedAnchorKeystore=/etc/ssh/cert \
    -o KMFPolicyDatabase=/etc/ssh/policy.xml \
    -o KMFPolicyName=ssh

# Run the client in a debug mode to test the configuration
# (will ask for a PIN for the token, use the 'pinfile'
# attribute if you do not want to set PIN for each
# invocation).
# Note that the space and the hash-sign in the PKCS#11 URI are
# percent-encoded. Percent-encoding is in particular important when
# used from config file. Otherwise the '#' would be mistaken for a
# comment and token name matching would fail!
ssh -p 2222 -vvv -o \
    IdentityFile=\
    "pkcs11:object=user;token=Sun%20Software%20PKCS%2311%20softtoken" \
    <PUT-YOUR-USERNAME-HERE>@localhost
```

Before you use the softtoken you should set its PIN by means of the `setpin` subcommand. See `pktool(1)` for more information.

Exit Status The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

Files `/etc/default/login`

Contains defaults for several `sshd_config` parameters, environment variables, and other environmental factors.

The following parameters affect environment variables (see `login(1)` and descriptions of these variables, above):

- `TIMEZONE`
- `HZ`
- `ALTSHELL`
- `PATH`

- SUPATH

The following /etc/default/login parameters supply default values for corresponding sshd_config(4) parameters:

- CONSOLE (see PermitRootLogin in sshd_config(4))
- PASSREQ (see PermitEmptyPasswords in sshd_config(4))
- TIMEOUT (see LoginGraceTime in sshd_config(4))

The following /etc/default/login parameters:

- UMASK
- ULIMIT

...set the umask(2) and file size limit of, respectively, the shells and commands spawned by sshd.

Finally, two /etc/default/login parameters affect the maximum allowed login attempts per-connection using interactive user authentication methods (for example, keyboard-interactive but not publickey), as per login(1):

- RETRIES
- SYSLOG_FAILED_LOGINS

/etc/ssh/sshd_config
: Contains configuration data for sshd. This file should be writable by root only, but it is recommended (though not necessary) that it be world-readable.

/etc/ssh/ssh_host_key
/etc/ssh/ssh_host_dsa_key
/etc/ssh/ssh_host_rsa_key
: Contains the private part of the host key. This file should only be owned by root, readable only by root, and not accessible to others. sshd does not start if this file is group/world-accessible.

/etc/ssh/ssh_host_key.pub
/etc/ssh/ssh_host_dsa_key.pub
/etc/ssh/ssh_host_rsa_key.pub
: Contains the public part of the host key. This file should be world-readable but writable only by root. Its contents should match the private part. This file is not used for encryption; it is provided only for the convenience of the user so its contents can be copied to known hosts files. These files are created using ssh-keygen(1).

/var/run/sshd.pid
: Contains the process ID of the sshd listening for connections. If there are several daemons running concurrently for different ports, this contains the pid of the one started last. The content of this file is not sensitive; it can be world-readable. You can use the PidFile keyword in sshd_config to specify a file other than /var/run/sshd.pid. See sshd_config(4).

`/etc/ssh/ssh_known_hosts` and `$HOME/.ssh/known_hosts`
: These files are consulted when using `rhosts` with public key host authentication to check the public key of the host. The key must be listed in one of these files to be accepted. The client uses the same files to verify that the remote host is the one it intended to connect. These files should be writable only by root or the owner. `/etc/ssh/ssh_known_hosts` should be world-readable, and `$HOME/.ssh/known_hosts` can but need not be world-readable.

`/etc/nologin`
: If this file exists, `sshd` refuses to let anyone except root log in. The contents of the file are displayed to anyone trying to log in, and non-root connections are refused. The file should be world-readable.

`$HOME/.ssh/authorized_keys`
: Lists the public keys (RSA or DSA) that can be used to log into the user's account. This file must be readable by root. This might, on some machines, imply that it is world-readable if the user's home directory resides on an NFS volume. It is recommended that it not be accessible by others. The format of this file is described above. Users will place the contents of their `identity.pub`, `id_dsa.pub` and/or `id_rsa.pub` files into this file, as described in ssh-keygen(1).

`$HOME/.rhosts`
: This file contains host-username pairs, separated by a space, one per line. The given user on the corresponding host is permitted to log in without password. The same file is used by `rlogind` and `rshd`. The file must be writable only by the user; it is recommended that it not be accessible by others. It is also possible to use `netgroups` in the file. Either host or user name may be of the form +@*groupname* to specify all hosts or all users in the group.

`$HOME/.shosts`
: For `ssh`, this file is exactly the same as for `.rhosts`. However, this file is not used by `rlogin` and `rshd`, so using this permits access using SSH only.

`/etc/hosts.equiv`
: This file is used during `.rhosts` authentication. In its simplest form, this file contains host names, one per line. Users on these hosts are permitted to log in without a password, provided they have the same user name on both machines. The host name can also be followed by a user name; such users are permitted to log in as any user on this machine (except root). Additionally, the syntax +@*group* can be used to specify netgroups. Negated entries start with a hyphen (`-`).

  If the client host/user is successfully matched in this file, login is automatically permitted, provided the client and server user names are the same. Additionally, successful RSA host authentication is normally required. This file must be writable only by root; it is recommended that it be world-readable.

  Warning: It is almost never a good idea to use user names in `hosts.equiv`. Beware that it really means that the named user(s) can log in as anybody, which includes `bin`, `daemon`,

adm, and other accounts that own critical binaries and directories. For practical purposes, using a user name grants the user root access. Probably the only valid use for user names is in negative entries. This warning also applies to rsh/rlogin.

/etc/ssh/moduli
A private file.

/etc/ssh/shosts.equiv
This file is processed exactly as /etc/hosts.equiv. However, this file might be useful in environments that want to run both rsh/rlogin and ssh.

$HOME/.ssh/environment
This file is read into the environment at login (if it exists). It can contain only empty lines, comment lines (that start with #), and assignment lines of the form *name*=*value*. The file should be writable only by the user; it need not be readable by anyone else. Environment processing is disabled by default and is controlled by means of the PermitUserEnvironment option.

$HOME/.ssh/rc
If this file exists, it is run with /bin/sh after reading the environment files but before starting the user's shell or command. If X11 spoofing is in use, this will receive the proto cookie pair in standard input (and DISPLAY in environment). This must call xauth in that case.

The primary purpose of $HOME/.ssh/rc is to run any initialization routines that might be needed before the user's home directory becomes accessible; AFS is a particular example of such an environment. If this file exists, it is run with /bin/sh after reading the environment files, but before starting the user's shell or command. It must not produce any output on stdout; stderr must be used instead. If X11 forwarding is in use, it will receive the proto cookie pair in its standard input and DISPLAY in its environment. The script must call xauth because sshd will not run xauth automatically to add X11 cookies.

This file will probably contain some initialization code followed by something similar to:

```
if read proto cookie && [ -n "$DISPLAY" ]
then
  if [ `echo $DISPLAY | cut -c1-10`  =  'localhost:' ]
  then
    # X11UseLocalhost=yes
    echo add unix:`echo $DISPLAY |
    cut -c11-` $proto $cookie
  else
    # X11UseLocalhost=no
    echo add $DISPLAY $proto $cookie
  fi | xauth -q -
fi
```

If this file does not exist, `/etc/ssh/sshrc` is run, and if that does not exist, `xauth` is used to store the cookie. `$HOME/.ssh/rc` should be writable only by the user, and need not be readable by anyone else.

`/etc/ssh/sshrc`
Similar to `$HOME/.ssh/rc`. This can be used to specify machine-specific login-time initializations globally. This file should be writable only by root, and should be world-readable.

Security `sshd` supports the use of several user authentication mechanisms: a public key system where keys are associated with users (through users' `authorized_keys` files), a public key system where keys are associated with hosts (see the `HostbasedAuthentication` configuration parameter), a GSS-API based method (see the `GssAuthentication` and `GssKeyEx` configuration parameters) and three initial authentication methods: `none`, `password`, and a generic prompt/reply protocol, `keyboard-interactive`.

`sshd` negotiates the use of the GSS-API with clients only if it has a GSS-API acceptor credential for the "host" service. This means that, for GSS-API based authentication, the server must have a Kerberos V `keytab` entry (see below) or the equivalent for any other GSS-API mechanism that might be installed.

In order for Kerberos authentication to work, a `host/`<*FQDN*> Kerberos principal must exist for each Fully Qualified Domain Name associated with the `in.sshd` server. Each of these `host/`<*FQDN*> principals must have a `keytab` entry in the `/etc/krb5/krb5.keytab` file on the `in.sshd` server. An example principal might be:

```
host/bigmachine.eng.example.com
```

See `kadmin`(1M) or `gkadmin`(1M) for instructions on adding a principal to a `krb5.keytab` file. See *Oracle Solaris 11.1 Administration: Security Services* for a discussion of Kerberos authentication.

GSS-API authorization is covered in `gss_auth_rules`(5).

`sshd` uses `pam`(3PAM) for the three initial authentication methods as well as for account management, session management, and password management for all authentication methods.

Specifically, `sshd` calls `pam_authenticate()` for the "none," "password" and "keyboard-interactive" SSHv2 `userauth` types. Other SSHv2 authentication methods do not call `pam_authenticate()`.`pam_acct_mgmt()` is called for each authentication method that succeeds.

`pam_setcred()` and `pam_open_session()` are called when authentication succeeds and `pam_close_session()` is called when connections are closed.

`pam_open_session()` and `pam_close_session()` are also called when SSHv2 channels with `ptys` are opened and closed.

Each SSHv2 `userauth` type has its own PAM service name:

SSHv2 Userauth	PAM Service Name
none	sshd-none
password	sshd-password
keyboard-interactive	sshd-kbdint
pubkey	sshd-pubkey
hostbased	sshd-hostbased
gssapi-with-mic	sshd-gssapi
gssapi-keyex	sshd-gssapi

If `pam_acct_mgmt()` returns `PAM_NEW_AUTHTOK_REQD` (indicating that the user's authentication tokens have expired), then `sshd` forces the use of "keyboard-interactive" `userauth`, if version 2 of the protocol is in use. The "keyboard-interactive" `userauth` will call `pam_chauthtok()` if `pam_acct_mgmt()` once again returns `PAM_NEW_AUTHTOK_REQD`. By this means, administrators are able to control what authentication methods are allowed for SSHv2 on a per-user basis.

Setting up Host-based Authentication

To establish host-based authentication, you must perform the following steps:

- Configure the client.
- Configure the server.
- Publish known hosts.
- Make appropriate entries in `/etc/ssh/shosts.equiv` and `~/.shosts`.

These steps are expanded in the following paragraphs.

- On a client machine, in the system-wide client configuration file, `/etc/ssh/ssh_config`, you must have the entry:

  ```
  HostbasedAuthentication yes
  ```

  See `ssh_config(4)` and `ssh-keysign(1M)`.

- On the server, in the system-wide server configuration file, `/etc/ssh/sshd_config`, you must have the entry:

  ```
  HostbasedAuthentication yes
  ```

  If per-user `.shost` files are to be allowed (see last step), in the same file, you must have:

  ```
  IgnoreRhosts no
  ```

  See `sshd_config(4)` for a description of these keywords.

- To publish known hosts, you must have entries for the clients from which users will be allowed host-based authentication. Make these entries in either or both of the system-wide file (`/etc/ssh/ssh_known_hosts`) or the per-user file (`~/.ssh/known_hosts`).
- Note that `sshd` uses `.shosts`, not `.rhosts`. If you want the functionality provided by `.rhosts`, but do not want to use `rlogin` or `rsh` because of their security shortcomings, you can use `.shosts` in conjunction with `sshd`. To use this feature, make appropriate entries in `/etc/ssh/shosts.equiv` and `~/.shosts`, in the format specified in `rhosts(4)`.

  For the vast majority of network environments, `.shosts` is preferred over `.rhosts`.

Using X.509 Certificates as Host Keys and User Identities

SSH commands can work with X.509v3 certificates when used as host keys and/or user identities. When used this way, PKCS#11 URIs can be used instead of plain filenames to locate keys and certificates. Host keys, user identities, and their corresponding certificates used for X.509 public key authentication can be stored only in PKCS#11 tokens. The only PKCS#11 URI attributes used by SSH are `object`, `token`, and `pinfile`. If no hardware keystore is available, the PKCS#11 `softtoken` keystore can be used.

For certificate validation (a server that authenticates a user, or a client that authenticates a server), the KMF policy needs a certificate mapper. Mappers are not set in the default policy file. See `TrustedAnchorKeystore`, `KMFPolicyDatabase`, and `KMFPolicyName` options in `sshd_config(4)`, and see `libkmf(3LIB)` for more information.

Self-signed certificates can only be used as host keys, never as user identities. If a self-signed certificate is used or if a host certificate cannot be validated on the client side due to a missing relevant trusted anchor certificate, the SSH client offers to store the key in the `known_host` database, as is currently done with plain public keys. Only the public key is extracted from the certificate and stored in the database, thus maintaining the current format. The reason for accepting such a certificate on the client side is that the initial host Key Exchange is not restartable. Given this, if a server uses certificates as its only host key(s), a client that is unable to validate the certificate would otherwise be denied from logging in.

See the `EXAMPLES` section for guidance on how to use the X.509 authentication.

Configuring `sshd` to Run OpenSSL in FIPS-140 Mode

To configure `sshd` to run OpenSSL in FIPS-140 mode, the variable `UseFIPS140` is set to `yes`.

SunSSH can still delegate cryptographic operations for user/host authentication to other parts of Solaris that might or might not be FIPS-140 certified. The default value of the `UseOpenSSLEngine` option is `no`; the setting of `UseOpenSSLEngine` to `yes` does not have an effect in FIPS mode.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	service/network/ssh
Interface Stability	Committed

The interface stability of `/etc/ssh/moduli` is Private.

See Also `kmfcfg(1)`, `login(1)`, `pktool(1)`, `scp(1)`, `ssh(1)`, `ssh-add(1)`, `ssh-agent(1)`, `ssh-keygen(1)`, `svcs(1)`, `gkadmin(1M)`, `kadmin(1M)`, `sftp-server(1M)`, `ssh-keysign(1M)`, `svcadm(1M)`, `libkmf(3LIB)`, `pam(3PAM)`, `rhosts(4)`, `ssh_config(4)`, `sshd_config(4)`, `attributes(5)`, `gss_auth_rules(5)`, `kerberos(5)`, `pam_roles(5)`, `pkcs11_softtoken(5)`, `smf(5)`

See the discussion of the `.k5login` file in `krb5_auth_rules(5)`.

Oracle Solaris 11.1 Administration: Security Services

Notes The `sshd` service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/network/ssh:default
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. The service's status can be queried using the `svcs(1)` command.

`sshd` always sets `PAM_RHOST` and sets `PAM_AUSER` in the case of host-based `userauth`. This behavior allows for remote logins to roles using host-based authentication. See `pam_roles(5)`.

Name ssh-keysign – ssh helper program for host-based authentication

Synopsis

```
ssh-keysign
```

Description `ssh-keysign` is used by `ssh(1)` to access the local host keys and generate the digital signature required during host-based authentication with SSH protocol version 2. This signature is of data that includes, among other items, the name of the client host and the name of the client user.

`ssh-keysign` is disabled by default and can be enabled only in the global client configuration file `/etc/ssh/ssh_config` by setting `HostbasedAuthentication` to `yes`.

`ssh-keysign` is not intended to be invoked by the user, but from `ssh`. See `ssh(1)` and `sshd(1M)` for more information about host-based authentication.

Files

`/etc/ssh/ssh_config` — Controls whether `ssh-keysign` is enabled.

`/etc/ssh/ssh_host_dsa_key`
`/etc/ssh/ssh_host_rsa_key` — These files contain the private parts of the host keys used to generate the digital signature. They should be owned by root, readable only by root, and not accessible to others. Because they are readable only by root, `ssh-keysign` must be `set-uid` root if host-based authentication is used.

Security ssh-keysign will not sign host-based authentication data under the following conditions:

- If the `HostbasedAuthentication` client configuration parameter is not set to `yes` in `/etc/ssh/ssh_config`. This setting cannot be overriden in users' `~/.ssh/ssh_config` files.
- If the client hostname and username in `/etc/ssh/ssh_config` do not match the canonical hostname of the client where `ssh-keysign` is invoked and the name of the user invoking `ssh-keysign`.

In spite of `ssh-keysign`'s restrictions on the contents of the host-based authentication data, there remains the ability of users to use it as an avenue for obtaining the client's private host keys. For this reason host-based authentication is turned off by default.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	network/ssh
Interface Stability	Committed

See Also `ssh(1)`, `sshd(1M)`, `ssh_config(4)`, `attributes(5)`

Authors Markus Friedl, `markus@openbsd.org`

History `ssh-keysign` first appeared in Ox 3.2.

Name statd – network status monitor

Synopsis

```
/usr/lib/nfs/statd
```

Description `statd` is an intermediate version of the status monitor. It interacts with `lockd(1M)` to provide the crash and recovery functions for the locking services on NFS. `statd` keeps track of the clients with processes which hold locks on a server. When the server reboots after a crash, `statd` sends a message to the `statd` on each client indicating that the server has rebooted. The client `statd` processes then inform the `lockd` on the client that the server has rebooted. The client `lockd` then attempts to reclaim the lock(s) from the server.

`statd` on the client host also informs the `statd` on the server(s) holding locks for the client when the client has rebooted. In this case, the `statd` on the server informs its `lockd` that all locks held by the rebooting client should be released, allowing other processes to lock those files.

`lockd` is started by `automountd(1M)`, `mount_nfs(1M)`, and `share(1M)` if NFS automounts are needed.

Files

`/var/statmon/sm`	lists hosts and network addresses to be contacted after a reboot
`/var/statmon/sm.bak`	lists hosts and network addresses that could not be contacted after last reboot
`/var/statmon/state`	includes a number which changes during a reboot
`/usr/include/rpcsvc/sm_inter.x`	contains the rpcgen source code for the interface services provided by the statd daemon.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/file-system/nfs

See Also `svcs(1)`, `automountd(1M)`, `lockd(1M)`, `mount_nfs(1M)`, `share(1M)`, `svcadm(1M)`, `attributes(5)`, `smf(5)`

System Administration Guide: IP Services

Notes The crash of a server is only detected upon its recovery.

The `statd` service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/network/nfs/status
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using svcadm(1M). The service's status can be queried using the svcs(1) command.

If it is disabled, it will be enabled by mount_nfs(1M), share_nfs(1M), and automountd(1M) unless its application/auto_enable property is set to false.

Name stclient – Service Tag Administration Program

Synopsis

```
stclient -x | -E [-r root_dir]

stclient -a [-i instance_URN] -p product_name -e product_version
     -t product_URN [-F parent_URN] -P product_parent
     [-I product_defined_instance_id] -m product_vendor -A platform_arch
     -z container -S source [-r root_dir]

stclient -u -i instance_URN -F parent_URN -I product_defined_instance_id
     [-r root_dir]

stclient -d -i instance_URN [-r root_dir]

stclient -g -i instance_URN [-r root_dir]

stclient -f -t product_URN [-r root_dir]

stclient -h

stclient -v
```

Description The `stclient` command displays, finds, adds, updates and deletes records in the Service Tag registry. The registry is in the XML file `/var/sadm/servicetag/registry/servicetag.xml`, and contains the inventory of the product instances installed in the machine. Each record has a unique product instance identifier which is generated when the service tag is added in the registry. This product instance identifier is used as a key when finding, updating or deleting the service tag records. The extract option prints out the registry contents in XML format in stdout.

The `stclient` command also runs in interactive mode. This mode is invoked by running `stclient` without any parameters. A menu of all the available options are displayed, and the user is prompted to enter different parameters depending on the option chosen.

Any user can extract or get the contents of the registry, but only users with the appropriate privileges, the "svctag" user, or the creator of the service tag is authorized to update or delete a service tag record.

Options The following options are supported:

`-p` *product_name*	Sets the product name of the service tag to be added. For example, "stclient -p "Solaris 10 Operating System"" sets the product name to the "Solaris 10 Operating System".
`-e` *product_version*	Sets the product version of the service tag to be added. For example, "stclient -e 5.10" sets the product version to "5.10".
`-t` *product_URN*	Sets the Sun product unique identifier of the service tag to be added. For example, "stclient -t urn:uuid:5005588c-36f3-11d6-9cec-fc96f718e113" sets

	the Sun product unique identifier to "urn:uuid:5005588c-36f3-11d6-9cec-fc96f718e113".
`-i` *instance_URN*	Sets the product instance unique identifier of the service tag. For example, "stclient -i 3a444ab2-1dd2-11b2-a69d-830020782a6b" sets the product instance unique identifier to "3a444ab2-1dd2-11b2-a69d-830020782a6b" This field may also be set when the service tag is added but is not typically used. The generation of the instance ID is normally left to the `stclient`.
`-F` *product_parent_URN*	Sets the Sun product unique identifier of parent product. For example, "stclient -F urn:uuid:5005588c-36f3-11d6-9cec-fc96f718e113" sets the Sun product unique identifier of parent product to "urn:uuid:5005588c-36f3-11d6-9cec-fc96f718e113".
`-P` *product_parent*	Sets the parent product of the service tag. For example, "stclient -P JES" sets the product parent to "JES".
`-I` *product_defined_instance_id*	Sets the product defined instance identifier. For example, "stclient -I ser:abc-klm-000-7190" sets the product defined instance identifier to "ser:abc-klm-000-7190".
`-m` *product_vendor*	Sets the vendor of the product. For example, "stclient –m Sun" sets the product vendor to "Sun".
`-A` *platform_arch*	Sets the platform architecture that the product is running on. For example, "stclient –A sparc" sets the platform architecture to "sparc"
`-z` *container*	Sets the container that the product is running on. For example, "stclient -z "global zone"" sets the container to "global zone".
`-S` *source*	Sets the source of this service tag, naming the entity that created it.
`-r` *root_dir*	Sets the root directory where the command searches for the registry. The command searches `/var/sadm/servicetag/registry` by default. When this option is used, the command looks for the Registry in the specified root directory, (for example, "`-r /home/user1`" searches `/home/user1/var/sadm/servicetag`). This is typically used for testing.

Operands

Function Letters The function portion of the key is specified by one of the following letters:

x Extract. Extracts and prints the contents of the Service Tag registry in XML format. An alternate root directory may be specified for testing purposes.

a Add. Adds a service tag in the registry. A unique product instance identifier may be supplied and is automatically generated if not provided. This field is returned by the command. The required fields for add are product name, product version, product URN, product parent, vendor, platform architecture, container and source.

u Update. Updates a service tag in the registry using the product instance URN as the key. The parent URN and product defined instance id fields can be updated.

d Delete. Deletes a service tag in the registry using the product instance URN as the key.

E Extract Environment. Enumerates to standard output the environmental or "agent"' information associated with the registered Service Tags on this system.

g Get. Gets and prints a service tag from the registry using the product instance URN as the key.

f Find. Finds and prints all the instance URNs of a given product URN.

h Help. Displays the command options.

v Version. Displays the version number of the command.

Exit Status

The following exit values are returned:

`0` Successful completion.

`> 0` An error occurred.

Examples

EXAMPLE 1 Adding a Service Tag in the Registry

To add a service tag, enter the following:

```
# stclient -a -p "Java Enterprise Web Server 6.0" -e 6.0 \
   -t urn:uuid:f2b8b59c-6eba-11d7-986f-9f5d47ec72fe  \
   -P Java Enterprise Server -m Sun -A sparc -z global -S patch
```

The screen displays the following:

```
Java Enterprise Web Server 6.0 6.0 added
Product instance URN=urn:st:7fc61914-1dd2-11b2-b992-830020782a6b
```

EXAMPLE 2 Updating a Service Tag in the Registry

To update a service tag, enter the following:

EXAMPLE 2 Updating a Service Tag in the Registry *(Continued)*

```
# stclient -u -i 7fc61914-1dd2-11b2-b992-830020782a6b \
   -I urn:st:product.defined.id
```

The screen displays the following:

```
Service tag updated
```

EXAMPLE 3 Extracting a Service Tag Registry

To extract a service tag, enter the following:

```
# stclient -x
```

The screen displays output similar to the following:

```
<?xml version="1.0" encoding="UTF-8"?>
<registry urn="urn:uuid:1234ab-00e1-11b3-98737646873" version="1.0">
   <service_tag>
  .
  .
  .
   </service_tag>
 </registry>
```

EXAMPLE 4 Finding all product instances

To find all product instances, enter the following:

```
# stclient -f -t urn:uuid:f2b8b59c-6eba-11d7-986f-9f5d47ec72fe \
   fc61914-1dd2-11b2-b992-830020782a6b
```

EXAMPLE 5 Listing the Environmental Information

To list the environmental information associated with the registered Service Tags on this system, enter the following:

```
# stclient -E
```

The screen displays output similar to the following:

```
<?xml version="1.0" encoding="UTF-8"?>
<agent>
   <agent_urn>urn:st:af15ee62-0bb3-ef2d-fa96-85a11996cc71</agent_urn>
  .
  .
  .
   </system_info>
 </agent>
```

Environment Variables See environ(5)for descriptions of the following environment variables that affect the execution of `stclient`: `LANG`, `LC_ALL`, `LC_CTYPE`, `LC_MESSAGES`, and `NLSPATH`.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/management/service-tag
Interface Stability	Committed

See Also in.stdiscover(1M), in.stlisten(1M)

Name stmfadm – SCSI target mode framework command line interface

Synopsis

```
stmfadm add-hg-member [-F] -g, --group-name group-name group-member...

stmfadm add-tg-member -g, --group-name group-name group-member...

stmfadm add-view [-n, --lun logical-unit-number
     -t, --target-group group-name -h, --host-group group-name] lu-name

stmfadm create-hg group-name

stmfadm create-lu [-p, --lu-prop logical-unit-property=val
     -s, --size size] lu-file

stmfadm create-tg group-name

stmfadm delete-hg group-name

stmfadm delete-lu lu-name

stmfadm delete-tg group-name

stmfadm import-lu lu-file

stmfadm list-hg [-v] [host-group-name...]

stmfadm list-tg [-v] [target-group-name...]

stmfadm list-lu [-v] [lu-name...]

stmfadm list-target [-v] [target-name...]

stmfadm list-view [-v] -l, --lu-name lu-name [entry-name...]

stmfadm list-state

stmfadm modify-lu [-p, --lu-prop logical-unit-property=val
     -s, --size size, -f, --file] lu-name|lu-file

stmfadm online-lu lu-name

stmfadm offline-lu lu-name

stmfadm online-target target-name

stmfadm offline-target target-name

stmfadm remove-hg-member -g, --group-name group-name group-member...

stmfadm remove-tg-member -g, --group-name group-name group-member...

stmfadm remove-view -l, --lu-name lu-name entry-name
```

Description The `stmfadm` command configures logical units within the SCSI Target Mode Framework (STMF) framework. The framework and this man page use the following terminology:

initiator
A device responsible for issuing SCSI I/O commands to a SCSI target and logical unit.

target
A device responsible for receiving SCSI I/O commands for a logical unit.

logical unit
A device within a target responsible for executing SCSI I/O commands.

logical unit number
The identifier of a logical unit within a target.

initiator group
An initiator group is a set of one or more initiators that are combined for the purposes of being applied to a *view* (see below). Unless the `-F` is used with the `add-hg-member` subcommand, an initiator cannot be a member of more than one initiator group.

target group
A target group is a set of one or more SCSI target ports that are treated the same when creating a *view* (see below). The set of logical units that a particular SCSI initiator can see is determined by the combined set of views.

Each logical unit has a set of view entries, and each view entry specifies a target group, host group, and a LUN. An initiator from that host group, when connecting through that target group, is able to identify and connect to that logical unit using the specified LUN. You can use views to restrict the set of logical units that a specific initiator can see, and assign the set of LUNs that will be used.

view
A view defines the association of an initiator group, a target group, and a logical unit number with a specified logical unit. Any view entry added to a logical unit must not be in conflict with existing view entries for that logical unit. A view entry is considered to be in conflict when an attempt is made to duplicate the association of any given initiator, target and logical unit number. As an example, logical unit `LU_0` has the following view entry associated with it:

```
Logical Unit: LU_0
            View Entry: 0
                initiator group: HostA
                target group: All targets
                logical unit number: 32
```

If you attempted the following:

```
# stmf add-view -n 31 -h HostA LU_0
```

...the operation would return an error with a message indicating that the view entry is in conflict with one or more existing view entries. This conflict arises because the existing view entry, `0`, already has mapped `LU_0` to logical unit number 32 for the initiator group `HostA`.

Sub-commands The `stmfadm` command supports the subcommands listed below.

`add-view` [`-n`, `--lun` *logical-unit-number* `-t`, `--target-group` *group-name* `-h`, `--host-group` *group-name*] *lu-name*

Adds a logical unit view entry to a logical unit *lu-name*, where *lu-name* is the STMF name for the logical unit as displayed by the `list-lu` subcommand. The `add-view` subcommand provides the user with a mechanism to implement access control for a logical unit and also provides a means of assigning a logical unit number to a logical unit for a given set of initiators and targets. A logical unit will not be available to any initiators until at least one view is applied. Each view entry gets assigned an entry name, which can be used to reference the entry in the `remove-view` and `list-view` subcommands.

`add-view` supports the following options:

`-n`, `--lun` *logical-unit-number*

logical-unit-number is an integer in the range 0-16383 to be assigned to the logical unit for this view entry. If this option is not specified, a logical unit number will be assigned by the STMF framework.

`-t`, `--target-group` *group-name*

group-name is the name of a target group previously created using the STMF `create-tg` subcommand. If this option is not specified, the logical unit will be available through all targets.

`-h`, `-host-group` *group-name*

group-name is the name of an host group previously created using the STMF `create-hg` subcommand. If this option is not specified, the logical unit will be available to all initiators that log in to the STMF framework.

`add-hg-member` [`-F`, `-—force`] `-g` *group-name group member...*

Add a host group member to a host group. *group-name* must be an existing group created using the `create-hg` subcommand. *group member* can be specified as *name_type.name_value*, where *name_type* can be one of the following:

```
wwn
iqn
eui
```

...and *name_value* is the value of the initiator name. As an example, to add a fibre channel initiator port world-wide name `200000e08b909221` to the host group `HostA`, the command would be:

```
# stmfadm add-hg-member -g HostA wwn.200000e08b909221
```

To add an ISCSI initiator node member with the name `iqn.1986-03.com.sun:01.46f7e262` to `HostA`, the command would be:

```
# stmfadm add-hg-member -g HostA iqn.1986-03.com.sun:01.46f7e262
```

Alternatively, members can be specified using their SCSI name string identifiers. To add the two initiators above using their SCSI name string identifiers, the commands would be:

```
# stmfadm add-hg-member -g HostA eui.200000e08b909221
# stmfadm add-hg-member -g HostA iqn.1986-03.com.sun:01.46f7e262
```

With the `-F` option, host group member can be a member of more than one host group. Without this option, a host group member can be a member of only one group.

`add-tg-member -g` *group-name group member...*
Add a target group member to a target group. *group-name* must be an existing group created using the `create-tg` subcommand. *group member* can be specified as *name_type.name_value*, where *name_type* can be one of the following:

```
wwn
iqn
eui
```

...and *name_value* is the value of the target name. As an example, to add a fibre channel target port world-wide name `501000e092376af7` to the target group `TG0`, the command would be:

```
# stmfadm add-tg-member -g TG0 wwn.501000e092376af7
```

To add an ISCSI target member with the name `iqn.1986-03.com.sun:target.01.01110` to `TG0`, the command would be:

```
# stmfadm add-tg-member -g TG0 iqn.1986-03.com.sun:target.01.01110
```

Alternatively, members can be specified using their SCSI name string identifiers. To add the two targets above using their SCSI name string identifiers, the commands would be:

```
# stmfadm add-tg-member -g TG0 eui.501000e092376af7
# stmfadm add-tg-member -g TG0 iqn.1986-03.com.sun:target.01.01110
```

A target group member cannot be a member of more than one target group.

`create-hg` *group-name*
Create an initiator group with the name *group-name*. *group-name* is a string of Unicode characters with a maximum length of 255. The group name must be unique within the STMF system.

`create-lu` [`-p`, `--lu-prop` *logical-unit-property=val* `--s`, `--size` *size*] *lu-file*
Create a logical unit that can be registered with STMF. For the `-p` option, *logical-unit-property* can be one of the following properties:

`alias`
Up to 255 characters, representing a user-defined name for the device. The default is the name of the backing store.

`blk`
Specifies the block size for the device. The default is 512, but do not set this value higher than 4096 KB.

`guid`
Thirty–two hexadecimal ASCII characters representing a valid NAA Registered Extended Identifier. The default is set by the STMF to a generated value.

`host-id`
Eight hexadecimal ASCII characters representing the host ID assignment. This will be used to generate the globally unique identifier (GUID) for the logical unit. The default is the value returned by `hostid(1)`.

`meta`
Metadata file name. When specified, will be used to hold the SCSI metadata for the logical unit. There is no default.

`mgmt-url`
Up to 1024 characters representing a Management Network Address URL. More than one URL can be passed as a single parameter by using space-delimited URLs enclosed inside a single pair of quotation marks (").

`oui`
Organizational Unique Identifier. Six hexadecimal ASCII characters representing the IEEE OUI company ID assignment. This will be used to generate the device identifier (GUID). The default is `00144F`.

`pid`
Sixteen characters of product identification SCSI SPC-3. This value will be reflected in the Standard `INQUIRY` data returned for the device. The default is `COMSTAR`.

`serial`
Serial Number. Specifies the SCSI Vital Product Data Serial Number (page `80h`). It is a character value up to 252 bytes in length. There is no default value.

`vid`
Eight characters of vendor identification per SCSI SPC-3. This value will be reflected in the Standard `INQUIRY` data returned for the device. The default is `SUN`.

`wcd`
Write-back cache disable. Specify `true` or `false`to determine write-back cache disable behavior. The default is the write-back cache setting of the backing store device specified by the *lu-file* argument.

`wcms`
Write cache setting changeable. Specified as `true` or `false`. When `true`, a SCSI `MODE SELECT` from the initiator can change the `WRITE CACHE ENABLE` bit on the caching mode page. When false, the `WRITE CACHE ENABLE` bit is not changeable. This setting does not impact the ability for `stmfadm` or a `libstmf(3LIB)` function to modify the write cache disable setting. The default setting is `true`.

`wp`
Write-protect bit. Specify `true` or `false` to determine whether the device reports as write-protected. The default is `false`.

For the `-s` option, *size* is an integer followed by one of the following letters, to indicate a unit of size:

`k` kilobyte

`m` megabyte

`g` gigabyte

`t` terabyte

`p` petabyte

`e` exabyte

lu-file is the file to be used as the backing store for the logical unit. If the `-s` option is not specified, the size of the specified *lu-file* will be used as the size of the logical unit. Logical units registered with the STMF require space for the metadata to be stored. When a `zvol` is specified as the backing store device, the default will be to use a special property of the `zvol` to contain the metadata. For all other devices, the default behavior will be to use the first 64k of the device. An alternative approach would be to use the `meta` property in a `create-lu` command to specify an alternate file to contain the metadata. It is advisable to use a file that can provide sufficient storage of the logical unit metadata, preferably 64k.

`create-tg` *group-name*
Create a target group with the name *group-name*. *group-name* is a string of Unicode characters with a maximum length of 255. The group name must be unique within the STMF system.

`delete-hg` *group-name*
Delete the host group that identified by *group-name*.

`delete-lu` *lu-name*
Deletes an existing logical unit that was created using `stmfadm create-lu`. This effectively unloads the logical unit from the STMF framework. Any existing data on the logical unit remains intact.

`delete-tg` *group-name*
Delete the target group that identified by *group-name*.

`import-lu` *lu-file*
Imports and loads a logical unit into the STMF that was previously created using `stmfadm create-lu` and was then deleted from the STMF using `stmfadm delete-lu`. On success, the logical unit is again made available to the STMF. *lu-file* is the filename used in the `stmfadm create-lu` command. If this logical unit is using a separate metadata file, the filename in the `meta` property value that was used in the `stmfadm create-lu` command must be used here.

`list-hg` [`-v`,`--verbose`] [*host-group-name*...]
Lists information for the host group in the system referenced by *host-group-name*. If *host-group-name* is not specified, all host groups in the system will be listed. If the `--v` or `--verbose` option is specified, all members within a host group are displayed.

`list-lu` [`-v`,`--verbose`] [*lu-name*...]
Lists information for the logical unit in the system referenced by *lu-name*. If *lu-name* is not specified, all logical units in the system will be listed. If the `-v` or `--verbose` option is specified, additional information about the logical unit will be displayed.

`list-target` [`-v`,`--verbose`] [*target-name*...]
Lists information for the target port in the system referenced by *target-name*. If target name is not specified, all target ports in the system will be listed. If the `-v` or `--verbose` option is specified, additional information about the target along with SCSI session information for logged-in initiators is displayed.

`list-tg` [`-v`,`--verbose`] [*target-group-name*...]
Lists information for the target group in the system referenced by *target-group-name*. If *target-group-name* is not specified, all target groups in the system will be listed. If the `--v` or `--verbose` option is specified, all members within a target group are displayed.

`list-view` [`-v`, `-—verbose`] `--l`, `--lu-name` *lu-name* [*entry-name*...]
Lists the view entry for the logical unit referenced by *lu-name*. If *entry-name* is not specified, all view entries for the specified logical unit will be listed.

If the `-v` option is specified, additional information about logical-unit-number assigned per host associated with the view are displayed.

`modify-lu` [`-p`, `--lu-prop` *logical-unit-property=val* `--s`, `--size` *size*, `-f`, `--file`] *lu-name*|*lu-file*
Modifies attributes of a logical unit created using the `stmfadm create-lu` command. For the `-p` option, *logical-unit-property* can be one of the following properties:

`alias`
Up to 255 characters, representing a user-defined name for the device. The default is the name of the backing store.

`mgmt-url`
Up to 1024 characters representing a Management Network Address URL. More than one URL can be passed as a single parameter by using space-delimited URLs enclosed inside a single pair of quotation marks (").

`wcd`
Write-back cache disable. Specify `true` or `false`to determine write-back cache disable behavior. The default is the write-back cache setting of the backing store device specified by the *lu-file* argument.

`wp`
Write-protect bit. Specify `true` or `false` to determine whether the device reports as write-protected. The default is `false`.

`wcms`
Write cache setting changeable. Specified as `true` or `false`. When `true`, a SCSI `MODE SELECT` from the initiator can change the `WRITE CACHE ENABLE` bit on the caching mode page. When false, the `WRITE CACHE ENABLE` bit is not changeable. This setting does not impact the ability for `stmfadm` or a `libstmf(3LIB)` function to modify the write cache disable setting. The default setting is `true`.

For the `-s` option, *size* is an integer followed by one of the following letters, to indicate a unit of size:

`k` kilobyte

`m` megabyte

`g` gigabyte

`t` terabyte

`p` petabyte

`e` exabyte

lu-name is the `guid` of the logical unit to be modified. If the `-f` option is specified, the operand is interpreted as a file name. This provides the ability to modify a logical unit that is not currently imported into the STMF.

`online-lu` *lu-name*
Online a logical unit currently registered with the STMF.

`online-target` *target-name*
Online the specified target.

`offline-lu` *lu-name*
Offline a logical unit currently registered with the STMF.

`offline-target` *target-name*
Offline the specified target.

`list-state`
Lists the operational and configuration state of the STMF.

`remove-hg-member -g` *group-name group member*
Removes a host group member from a host group. *group-name* must be an existing group created using the `create-hg` subcommand. *group member* can be specified as *name_type.name_value*, where *name_type* can be one of the following:

```
wwn
iqn
eui
```

...and *name_value* is the value of the initiator name. As an example, to remove the fibre channel initiator port world-wide name `200000e08b909221` from the host group `HostA`, the command would be:

```
# stmfadm remove-hg-member -g HostA wwn.200000e08b909221
```

To remove the ISCSI initiator node member with the name `iqn.1986-03.com.sun:01.46f7e262` from `HostA`, the command would be:

```
# stmfadm remove-hg-member -g HostA iqn.1986-03.com.sun:01.46f7e262
```

Alternatively, members can be specified using their SCSI name string identifiers. To remove the two initiators above using their SCSI name string identifiers, the commands would be:

```
# stmfadm remove-hg-member -g HostA eui.200000e08b909221
# stmfadm remove-hg-member -g HostA iqn.1986-03.com.sun:01.46f7e262
```

`remove-tg-member -g` *group-name group member*

Removes a target group member from a target group. *group-name* must be an existing group created using the `create-tg` subcommand. *group member* can be specified as *name_type.name_value*, where *name_type* can be one of the following:

```
wwn
iqn
eui
```

...and *name_value* is the value of the target name. As an example, to remove the fibre channel target port world-wide name `501000e092376af7` from the target group `TG0`, the command would be:

```
# stmfadm remove-tg-member -g TG0 wwn.501000e092376af7
```

To remove the ISCSI target member with the name `iqn.1986-03.com.sun:target.01.01110` from `TG0`, the command would be:

```
# stmfadm remove-tg-member -g TG0 iqn.1986-03.com.sun:target.01.01110
```

Alternatively, members can be specified using their SCSI name string identifiers. To remove the two targets above using their SCSI name string identifiers, the commands would be:

```
# stmfadm remove-tg-member -g TG0 eui.501000e092376af7
# stmfadm remove-tg-member -g TG0 iqn.1986-03.com.sun:target.01.01110
```

`remove-view --l, --lu-name` *lu-name entry-name*

Removes one or more logical unit view entries from a logical unit.

Examples

EXAMPLE 1 Creating a Host group with Two Initiator Ports

The following commands use the `create-hg` and `add-hg-member` subcommands to create a host group and add two initiator ports to that host group.

```
# stmfadm create-hg host-group-a
# stmfadm add-hg-member -g host-group-a wwn.210105b0000d92d0
```

EXAMPLE 2 Adding a View Entry to a Logical Unit

The following command uses the `add-view` subcommand to allow access from `host-group-a` to a logical unit.

```
# stmfadm add-view -h host-group-a 6000AE40C5000000000046FC4FEA001C
```

EXAMPLE 3 Listing a View Entry

The following command uses the `list-view` subcommand to list all view entries for the specified logical unit.

```
# stmfadm list-view -l 6000AE40C5000000000046FC4FEA001C
View Entry: 0
                Host group      : host-group-a
                Target group    : All
                LUN               : 0
```

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/storage/scsi-target-mode-framework
Interface Stability	Committed

See Also

`hostid(1)`, `sbdadm(1M)`, `libstmf(3LIB)`, `attributes(5)`

Name stmsboot – administration program for the Solaris I/O multipathing feature

Synopsis

```
/usr/sbin/stmsboot [[-D (fp | mpt | mpt_sas | iscsi) ] -d | -e | -u]
      | -L | -l controller_number]
```

Description The Solaris I/O multipathing feature is a multipathing solution for storage devices that is part of the Solaris operating environment. This feature was formerly known as Sun StorEdge Traffic Manager (STMS) or MPxIO.

The `stmsboot` program is an administrative command to manage enumeration of multipath–capable devices with Solaris I/O multipathing. Solaris I/O multipathing-enabled devices are enumerated under `scsi_vhci(7D)`, providing multipathing capabilities. Solaris I/O multipathing-disabled devices are enumerated under the physical controller.

In the `/dev` and `/devices` trees, Solaris I/O multipathing-enabled devices receive new names that indicate that they are under Solaris I/O multipathing control. This means a device will have a different name from its original name (after enabling) when it is under Solaris I/O multipathing control. The `stmsboot` command automatically updates `/etc/vfstab` and dump configuration to reflect the device names changes when enabling or disabling Solaris I/O multipathing. One reboot is required for changes to take effect.

Options The following options are supported:

`-e [ -D fp | mpt | mpt_sas | iscsi ]`
Enables Solaris I/O multipathing on all supported multipath-capable controller ports, including `fp(7d)`, `mpt(7D)`, `mpt_sas(7D)`, and `iscsi(7D)` port drivers. Multipath-capable ports include fibre channel (`fp(7d)`) controller ports and SAS (`mpt(7D)` or `mpt_sas(7D)`) controller ports. Following this enabling, you are prompted to reboot. During the reboot, `vfstab` and the dump configuration will be updated to reflect the device name changes. Specifying `-D mpt`, `-D mpt_sas`, or `-D fp` limits the enabling operation to ports attached using the specified driver.

`-d [ -D fp | mpt | mpt_sas | iscsi ]`
Disables Solaris I/O multipathing on all supported multipath-capable controller ports, including `fp(7d)`, `mpt(7D)`, `mpt_sas(7D)`, and `iscsi(7D)` port drivers. Multipath-capable ports include fibre channel (`fp(7d)`) controller ports and SAS (`mpt(7D)` or `mpt_sas(7D)`) controller ports. Following this disabling, you are prompted to reboot. During the reboot, `vfstab` and the dump configuration will be updated to reflect the device name changes. Specifying `-D mpt`, `-D mpt_sas`, or `-D fp` limits the disabling operation to ports attached using the specified driver.

`-u [ -D fp | mpt | mpt_sas | iscsi ]`
Updates `vfstab` and the dump configuration after you have manually modified the configuration to have Solaris I/O multipathing enabled or disabled on specific `fp(7d)`, `mpt(7D)`, `mpt_sas(7D)`, and `iscsi(7D)` controller ports. This option prompts you to reboot. During the reboot, `vfstab` and the dump configuration will be updated to reflect the device name changes.

`-L`
Display the device name changes from non-Solaris I/O multipathing device names to Solaris I/O multipathing device names for multipath-enabled controller ports. If Solaris I/O multipathing is not enabled, then no mappings are displayed.

`-l` *controller_number*
Display the device name changes from non-Solaris I/O multipathing device names to Solaris I/O multipathing device names for the specified controller. If Solaris I/O multipathing is not enabled, then no mappings are displayed.

Note that `mpt_sas(7D)` has MPxIO turned on by default. This means that when using the `-L` or `-l` option with `-D mpt_sas`, `stmsboot` does not display any non-multipathed and multipathed device names.

Usage

The primary function of `stmsboot` is to control the enabling and disabling of Solaris I/O multipathing on the host. The utility automatically updates `vfstab(4)` and `dumpadm(1M)` configuration to reflect device name changes. The system administrator is responsible for modifying application configuration (for example, backup software, DBMS, and so forth) to reflect updated device names.

The `-L` and `-l` options display the mapping between multipathed and non-multipathed device names. These options function only after changes to the Solaris I/O multipathing configuration have taken effect, that is, following the reboot after invoking `stmsboot -e`.

ZFS datasets, including ZFS root datasets, are correctly handled by `stmsboot`.

Examples

EXAMPLE 1 Enabling Solaris I/O Multipathing

To enable Solaris I/O multipathing for all multipath-capable controllers, run:

```
# stmsboot -e
```

To enable Solaris I/O multipathing on multipath-capable `mpt(7D)` controller ports, enter:

```
# stmsboot -D mpt -e
```

To enable Solaris I/O multipathing on multipath-capable `mpt_sas(7D)` controller ports, enter:

```
# stmsboot -D mpt_sas -e
```

To enable Solaris I/O Multipathing on multipath-capable fibre channel controller ports, enter:

```
# stmsboot -D fp -e
```

To enable Solaris I/O Multipathing on multipath-capable iSCSI controller ports, enter:

```
# stmsboot -D iscsi -e
```

EXAMPLE 2 Disabling Solaris I/O Multipathing

To disable Solaris I/O multipathing on all multipath-capable controllers, enter:

```
# stmsboot -d
```

To disable Solaris I/O multipathing on multipath-capable mpt(7D) controller ports, enter:

```
# stmsboot -D mpt -d
```

To disable Solaris I/O multipathing on multipath-capable mpt_sas(7D) controller ports, enter:

```
# stmsboot -D mpt_sas -d
```

To disable Solaris I/O multipathing on multipath-capable iSCSI controller ports, enter:

```
# stmsboot -D iscsi -d
```

To disable Solaris I/O multipathing on multipath-capable fibre channel controller ports, enter:

```
# stmsboot -D fp -d
```

EXAMPLE 3 Enabling Solaris I/O Multipathing on Selected Ports

To enable Solaris I/O multipathing on specific fibre channel controller ports and disable the feature on others, manually edit the `/etc/driver/drv/fp.conf` file. (See fp(7d).) The following command will update vfstab(4) and dumpadm(1M) configurations to reflect the changed device names:

```
# stmsboot -u
```

A similar procedure involving the `/etc/driver/drv/mpt.conf` file should be followed for devices attached by means of the mpt(7D) driver. For devices attached by means of the iscsi(7D) driver, follow a similar procedure that uses the `/etc/driver/drv/iscsi.conf` file.

Attributes

See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os, system/library
Interface Stability	Obsolete

See Also

dumpadm(1M), fsck(1M), mpathadm(1M), ufsdump(1M), zfs(1M), zpool(1M), ufsdump(4), vfstab(4), emlxs(7D), fcp(7D), fp(7d), iscsi(7D), mpt(7D), mpt_sas(7D), qlc(7D), scsi_vhci(7D)

Solaris SAN Configuration and Multipathing Guide (see `http://docs.oracle.com`)

Consult a particular storage product's system administrator's guide and release notes for further information specific to that product.

Notes Solaris I/O multipathing is not supported on all devices. After enabling Solaris I/O multipathing, only supported devices are placed under Solaris I/O multipathing control. Non-supported devices remain unchanged.

For Solaris releases prior to the current release, the `-e` and `-d` options replace `mpxio-disable` property entries with a global `mpxio-disable` entry in `fp.conf`.

Enabling Solaris I/O Multipathing on a Sun StorEdge Disk Array

The following applies to Sun StoreEdge T3, 3910, 3960, 6120, and 6320 storage subsystems.

To place your Sun StorEdge disk subsystem under Solaris I/O multipathing control, in addition to enabling Solaris I/O multipathing, the `mp_support` of the subsystem must be set to `mpxio` mode. The preferred sequence is to change the subsystem's `mp_support` to `mpxio` mode, then run `stmsboot -e`. If Solaris I/O multipathing is already enabled but the subsystem's `mp_support` is not in `mpxio` mode, then change the `mp_support` to `mpxio` mode and run `stmsboot -u`.

Refer to the *Sun StorEdge Administrator's Guide* for your subsystem for more details.

Using `ufsdump`

The `ufsdump(1M)` command records details of filesystem dumps in `/etc/dumpdates` (see `ufsdump(4)`). Among other items, the entries contain device names. An effect of the "active" `stmsboot` options (`-e`, `-d`, and `-u`) is to change the device name of a storage device.

Because `stmsboot` does not modify `dumpdates`, entries will refer to obsolete device names, that is, device names that were in effect before Solaris I/O multipathing configuration changes were performed. In this situation `ufsdump` will behave as if no previous dump of the filesystem had been performed. A level 0 dump will be performed.

Procedure to Use `stmsboot` in Conjunction with Sun Cluster

If possible, invoke `stmsboot -e` before installing Sun Cluster software. After executing `stmsboot`, install Sun Cluster software normally.

If Sun Cluster software is installed before executing `stmsboot`, follow this procedure:

On each machine in the cluster where Solaris I/O multipathing is required, execute:

```
# stmsboot -e
```

...and allow the system to reboot.

When the system comes up, enter the following two commands:

1. # **/usr/cluster/bin/scdidadm -C**
2. # **/usr/cluster/bin/scdidadm -r**

   The preceding commands update `did` mappings with new device names while preserving `did` instance numbers for disks that are connected to multiple cluster nodes. `did` instance numbers of the local disks might not be preserved. For this reason, the `did` disk names for local disks might change.

   The remaining steps are required only if local `did` number changes require editing of `/etc/vfstab`. If such editing is not required, run `/usr/cluster/bin/scgdevs` from each node in the cluster to complete the procedure.

3. Update `/etc/vfstab` to reflect any new `did` disk names for your local disks.
4. Reboot the system.

To disable the Solaris multipathing feature, use `stmsboot -d` (instead of `stmsboot -e`), then follow the procedure above.

To view mappings between the old and new device names, run `stmsboot -L`. To view `did` device name mappings, run `/usr/cluster/bin/scdidadm -L`.

With active-passive storage arrays, it is possible that while your host is rebooting the array controller could failover the path that a particular target is using. In this scenario, `fsck(1M)` will fail to open the physical path listed in `/etc/vfstab`. The `svc:/system/filesystem/local:default` SMF service will transition to a maintenance state as a result. To rectify this, consult the documentation for your storage array to failback the path. The `mpathadm(1M)` can assist with determining the active and passive path(s).

Limitations On x86 platforms, the current Solaris release does not support disabling Solaris I/O multipathing of boot devices attached by means of fibre channel. Solaris I/O multipathing is always enabled for supported fibre channel-attached boot devices. Disabling Solaris I/O multipathing in this situation must be performed on a per-port basis. See `fp(7d)`.

Executing `devfsadm -C` removes obsolete device entries that `stmsboot` relies on. This will prevent correct operation of the `-d` option for boot devices (regardless of platform type) and the `-L` option.

Name strace – print STREAMS trace messages

Synopsis `strace [`*mid sid level*`]...`

Description `strace` without arguments writes all STREAMS event trace messages from all drivers and modules to its standard output. These messages are obtained from the STREAMS log driver (see `log(7D)`). If arguments are provided, they must be in triplets of the form *mid, sid, level*, where *mid* is a STREAMS module ID number, *sid* is a sub-ID number, and *level* is a tracing priority level. Each triplet indicates that tracing messages are to be received from the given module/driver, sub-ID (usually indicating minor device), and priority level equal to, or less than the given level. The token `all` may be used for any member to indicate no restriction for that attribute.

The format of each trace message output is:

<*seq*> <`time`> <*ticks*> <*level*> <*flags*> <*mid*> <*sid*> <*text*>

<*seq*>	trace sequence number
<`time`>	time of message in *hh:mm:ss*
<*ticks*>	time of message in machine ticks since boot
<*level*>	tracing priority level
<*flags*>	E : message is also in the error log F : indicates a fatal error N : mail was sent to the system administrator (hardcoded as root)
<*mid*>	module ID number of source
<*sid*>	sub-ID number of source
<*text*>	formatted text of the trace message

Once initiated, `strace` will continue to execute until terminated by the user.

Examples **EXAMPLE 1** A sample output of the `strace` command:

The following example outputs all trace messages from the module or driver whose module ID is `41`:

```
strace  41 all all
```

The following example outputs those trace messages from driver or module ID `41` with sub-IDs `0`, `1`, or `2`:

```
strace  41 0 1  41 1 1  41 2 0
```

Messages from sub-IDs `0` and `1` must have a tracing level less than or equal to `1`. Those from sub-ID `2` must have a tracing level of `0`.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also attributes(5), log(7D)

STREAMS Programming Guide

Notes

- There is no restriction to the number of strace processes opening the STREAMS log driver at a time.
- The log-driver records the list of the triplets specified in the command invocation, and compares each potential trace message against this list to decide if it should be formatted and sent up to the strace process. Hence, long lists of triplets will have a greater impact on overall STREAMS performance. Running strace will have the most impact on the timing of the modules and drivers generating the trace messages that are sent to the strace process. If trace messages are generated faster than the strace process can handle them, some of the messages will be lost. This last case can be determined by examining the sequence numbers on the trace messages output.

Name strclean – STREAMS error logger cleanup program

Synopsis

```
strclean [-a age] [-d logdir]
```

Description `strclean` is used to clean up the STREAMS error logger directory on a regular basis (for example, by using `cron`. By default, all files with names matching `error.*` in `/var/adm/streams` that have not been modified in the last three days are removed.

Options The following options are supported:

`-a` *age* The maximum age in days for a log file can be changed using the `-a` option.

`-d` *logdir* A directory other than `/var/adm/streams` can be specified using the `-d` option.

Examples **EXAMPLE 1** A sample of using the `strclean` command.

This example has the same result as running `strclean` with no arguments:

```
example%  strclean -d /var/adm/streams -a 3
```

Files `/var/adm/streams/error.*`

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also `cron(1M)`, `strerr(1M)`, `attributes(5)`

STREAMS Programming Guide

Notes `strclean` is typically run from `cron` on a daily or weekly basis.

Name strerr – STREAMS error logger daemon

Synopsis

```
strerr
```

Description `strerr` receives error log messages from the STREAMS-based log driver (see `log(7D)`) and appends them to a log file. The resultant error log files reside in the directory `/var/adm/streams`, and are named `error.`*mm*`-dd`, where *mm* is the month and `dd` is the day of the messages contained in each log file.

The format of an error log message is:

<*seq*> <`time`> <*ticks*> <*flags*> <*mid*> <*sid*> <*text*>

<*seq*> error sequence number

<`time`> time of message in hh:mm:ss

<*ticks*> time of message in machine ticks since boot priority level

<*flags*> `T` : the message was also sent to a tracing process `F` : indicates a fatal error `N` : send mail to the system administrator (hardcoded as root)

<*mid*> module ID number of source

<*sid*> sub-ID number of source

<*text*> formatted text of the error message

Messages that appear in the error log are intended to report exceptional conditions that require the attention of the system administrator. Those messages which indicate the total failure of a STREAMS-based driver or module should have the `F` flag set. Those messages requiring the immediate attention of the administrator will have the `N` flag set, which causes the error logger to send the message to the system administrator using `mail`. The priority level usually has no meaning in the error log but will have meaning if the message is also sent to a tracer process.

Once initiated, `strerr` continues to execute until terminated by the user. It is commonly executed asynchronously.

Files `/var/adm/streams/error.`*mm*`-dd` error log file.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also `attributes(5)`, `log(7D)`

STREAMS Programming Guide

Notes There is no restriction to the number of `strerr` processes opening the STREAMS–based log driver at a time.

If a module or driver is generating a large number of error messages, running the error logger will cause a degradation in STREAMS performance. If a large burst of messages are generated in a short time, the log driver may not be able to deliver some of the messages. This situation is indicated by gaps in the sequence numbering of the messages in the log files.

Name sttydefs – maintain line settings and hunt sequences for TTY ports

Synopsis

```
/usr/sbin/sttydefs -a ttylabel [-b] [-f final-flags]
     [-i initial-flags] [-n nextlabel]

/usr/sbin/sttydefs -l [ttylabel]

/usr/sbin/sttydefs -r ttylabel
```

Description `sttydefs` is an administrative command that maintains the line settings and hunt sequences for the system's TTY ports by making entries in, and deleting entries from the `/etc/ttydefs` file.

`sttydefs` with a `-a` or `-r` option may be invoked only by the super-user. `sttydefs` with `-l` may be invoked by any user on the system.

Options The following options are supported:

`-a` *ttylabel* — Add a record to the `ttydefs` file, using *ttylabel* as its label. The following describes the effect of the `-b`, `-n`, `-i`, or `-f` options when used in conjunction with the `-a` option:

`-b` — Enable autobaud. Autobaud allows the system to set the line speed of a given TTY port to the line speed of the device connected to the port without the user's intervention.

`-f` *final-flags* — Specify the value to be used in the *final-flags* field in `/etc/ttydefs`. *final-flags* must be in a format recognized by the `stty` command. *final-flags* are the `termio(7I)` settings used by `ttymon` after receiving a successful connection request and immediately before invoking the service on the port. If this option is not specified, `sttydefs` will set *final-flags* equal to the `termio(7I)` flags `9600` and `sane`.

`-i` *initial-flags* — Specify the value to be used in the *initial-flags* field in `/etc/ttydefs`. *initial-flags* must be in a format recognized by the `stty` command. These flags are used by `ttymon` when searching for the correct baud rate. They are set prior to writing the prompt. If this option is not specified, `sttydefs` will set *initial-flags* equal to the `termio(7I)` flag `9600`.

`-n` *nextlabel* — Specify the value to be used in the *nextlabel* field in `/etc/ttydefs`. If this option is not specified, `sttydefs` will set *nextlabel* equal to *ttylabel*.

`-l`[*ttylabel*] — If a *ttylabel* is specified, `sttydefs` displays the record from `/etc/ttydefs` whose TTY label matches the specified *ttylabel*. If no *ttylabel* is specified, `sttydefs` displays the entire contents of `/etc/ttydefs`. `sttydefs` verifies that each entry it displays is correct and that the entry's *nextlabel* field references an existing

`-r` *ttylabel* — Remove any record in the `ttydefs` file that has *ttylabel* as its label.

Output If successful, `sttydefs` will exit with a status of `0`. `sttydefs -l` will generate the requested information and send it to standard output.

Examples **EXAMPLE 1** A sample of `sttydefs` command.

The following command lists all the entries in the `ttydefs` file and prints an error message for each invalid entry that is detected.

```
example# sttydefs -l
```

The following shows a command that requests information for a single label and its output:

```
example# sttydefs -l 9600
----------------------------------------------------------------
9600:9600 hupcl erase ^h:9600 sane ixany tab3 hupcl erase ^h::4800
----------------------------------------------------------------
ttylabel:    9600
initial flags:    9600 hupcl erase ^h
final flags:    9600 sane ixany tab3 hupcl erase ^h
autobaud:    no
nextlabel:      4800
```

The following sequence of commands will add the labels `1200`, `2400`, `4800`, and `9600` and put them in a circular list:

```
sttydefs -a 1200 -n 2400 -i 1200 -f "1200 sane"
sttydefs -a 2400 -n 4800 -i 2400 -f "2400 sane"
sttydefs -a 4800 -n 9600 -i 4800 -f "4800 sane"
sttydefs -a 9600 -n 1200 -i 9600 -f "9600 sane"
```

Files `/etc/ttydefs`

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also `attributes(5)`, `termio(7I)`

Name su – become superuser or another user

Synopsis

```
su [-] [username [arg...]]
```

Description The su command allows one to become another user without logging off or to assume a role. The default user *name* is root (superuser).

To use su, the appropriate password must be supplied (unless the invoker is already root). If the password is correct, su creates a new shell process that has the real and effective user ID, group IDs, and supplementary group list set to those of the specified *username*. Additionally, the new shell's project ID is set to the default project ID of the specified user. See getprojent(3PROJECT), setproject(3PROJECT). The new shell will be the shell specified in the shell field of *username*'s password file entry (see passwd(4)). If no shell is specified, /usr/bin/sh is used (see sh(1)). If superuser privilege is requested and the shell for the superuser cannot be invoked using exec(2), /sbin/sh is used as a fallback. To return to normal user ID privileges, type an EOF character (CTRL-D) to exit the new shell.

Any additional arguments given on the command line are passed to the new shell. When using programs such as sh, an *arg* of the form -c *string* executes *string* using the shell and an *arg* of -r gives the user a restricted shell.

To create a login environment, the command "su –" does the following:

- In addition to what is already propagated, the LC* and LANG environment variables from the specified user's environment are also propagated.
- Propagate TZ from the user's environment. If TZ is not found in the user's environment, su uses the TZ value from the TIMEZONE parameter found in /etc/default/login.
- Set MAIL to /var/mail/*new_user*.

If the first argument to su is a dash (-), the environment will be changed to what would be expected if the user actually logged in as the specified user. Otherwise, the environment is passed along, with the exception of $PATH, which is controlled by PATH and SUPATH in /etc/default/su.

All attempts to become another user using su are logged in the log file /var/adm/sulog (see sulog(4)).

Security su uses pam(3PAM) with the service name su for authentication, account management, and credential establishment.

Examples **EXAMPLE 1** Becoming User bin While Retaining Your Previously Exported Environment

To become user bin while retaining your previously exported environment, execute:

```
example% su bin
```

EXAMPLE 2 Becoming User bin and Changing to bin's Login Environment

To become user `bin` but change the environment to what would be expected if `bin` had originally logged in, execute:

```
example% su - bin
```

EXAMPLE 3 Executing command with user bin's Environment and Permissions

To execute command with the temporary environment and permissions of user `bin`, type:

```
example% su - bin -c "command args"
```

Environment Variables

Variables with `LD_` prefix are removed for security reasons. Thus, `su bin` will not retain previously exported variables with `LD_` prefix while becoming user `bin`.

If any of the `LC_*` variables (`LC_CTYPE`, `LC_MESSAGES`, `LC_TIME`, `LC_COLLATE`, `LC_NUMERIC`, and `LC_MONETARY`) (see `environ(5)`) are not set in the environment, the operational behavior of `su` for each corresponding locale category is determined by the value of the `LANG` environment variable. If `LC_ALL` is set, its contents are used to override both the `LANG` and the other `LC_*` variables. If none of the above variables are set in the environment, the "C" (U.S. style) locale determines how `su` behaves.

`LC_CTYPE` Determines how `su` handles characters. When `LC_CTYPE` is set to a valid value, `su` can display and handle text and filenames containing valid characters for that locale. `su` can display and handle Extended Unix Code (EUC) characters where any individual character can be `1`, `2`, or `3` bytes wide. `su` can also handle EUC characters of `1`, `2`, or more column widths. In the "C" locale, only characters from ISO 8859-1 are valid.

`LC_MESSAGES` Determines how diagnostic and informative messages are presented. This includes the language and style of the messages, and the correct form of affirmative and negative responses. In the "C" locale, the messages are presented in the default form found in the program itself (in most cases, U.S. English).

Files

`$HOME/.profile`
user's login commands for `sh` and `ksh`

`/etc/passwd`
system's password file

`/etc/profile`
system-wide `sh` and `ksh` login commands

`/var/adm/sulog`
log file

`/etc/default/su`
the default parameters in this file are:

`SULOG` If defined, all attempts to `su` to another user are logged in the indicated file.

`CONSOLE` If defined, all attempts to `su` to `root` are logged on the console.

`PATH` Default path. (`/usr/bin:`)

`SUPATH` Default path for a user invoking `su` to `root`. (`/usr/sbin:/usr/bin`)

`SYSLOG` Determines whether the `syslog(3C)` `LOG_AUTH` facility should be used to log all `su` attempts. `LOG_NOTICE` messages are generated for `su`'s to root, `LOG_INFO` messages are generated for `su`'s to other users, and `LOG_CRIT` messages are generated for failed `su` attempts.

`/etc/default/login`
the default parameters in this file are:

`SLEEPTIME` If present, sets the number of seconds to wait before login failure is printed to the screen and another login attempt is allowed. Default is `4` seconds. Minimum is `0` seconds. Maximum is `5` seconds.

Both `su` and `login(1)` are affected by the value of `SLEEPTIME`.

`TIMEZONE` Sets the TZ environment variable of the shell.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also

`csh(1)`, `env(1)`, `ksh(1)`, `login(1)`, `roles(1)`, `sh(1)`, `syslogd(1M)`, `exec(2)`, `getprojent(3PROJECT)`, `setproject(3PROJECT)`, `pam(3PAM)`, `pam_authenticate(3PAM)`, `pam_acct_mgmt(3PAM)`, `pam_setcred(3PAM)`, `pam.conf(4)`, `passwd(4)`, `profile(4)`, `sulog(4)`, `syslog(3C)`, `attributes(5)`, `environ(5)`

Name sulogin – access single-user mode

Synopsis

```
sulogin
```

Description The `sulogin` utility is automatically invoked by `init` when the system is first started. It prompts the user to type a user name and password to enter system maintenance mode (single-user mode) or to type EOF (typically CTRL-D) for normal startup (multi-user mode). The user should never directly invoke `sulogin`. The user must have the `solaris.system.maintenance` authorization.

The `sulogin` utility can prompt the user to enter the root password on a variable number of serial console devices, in addition to the traditional console device. See `consadm(1m)` and `msglog(7D)` for a description of how to configure a serial device to display the single-user login prompt.

Files `/etc/default/sulogin`

Default value can be set for the following flag:

PASSREQ

Determines if login requires a password. Default is `PASSREQ=YES`.

`/etc/default/login`

Default value can be set for the following flag:

`SLEEPTIME`

If present, sets the number of seconds to wait before login failure is printed to the screen and another login attempt is allowed. Default is `4` seconds. Minimum is `0` seconds. Maximum is `5` seconds.

Both `su(1M)` and `login(1)` are affected by the value of `SLEEPTIME`.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also `auths(1)`, `login(1)`, `consadm(1m)`, `init(1M)`, `su(1M)`, `attributes(5)`, `msglog(7D)`

Notes By default, the root user has all authorizations.

Granting the `solaris.system.maintenance` authorization to the Console User Rights Profile may have an undesirable side effect of granting the currently logged in user maintenance mode access. The `solaris.system.maintenance` authorization should be directly granted to appropriate users rather than through the Console User Rights Profile.

Name suriadm – administer shared objects based on storage URIs

Synopsis

```
/usr/sbin/suriadm command [options] [operands]

/usr/sbin/suriadm parse [-H] [-o p,p,...] URI
/usr/sbin/suriadm map [-H] [-o p,p,...] URI
/usr/sbin/suriadm lookup-mapping [-H] [-o p,p,...] URI
/usr/sbin/suriadm unmap URI
/usr/sbin/suriadm lookup-mapping [-t uri-type] [-H]
     [-o p,p,...] device-name
```

Description The `suriadm` command line administration tool allows system users to manage storage objects via storage URIs. The command allows to parse, map, unmap, query the state of mappings and look up storage URIs.

Supported Storage URIs Supported storage URIs are defined in `suri(5)` manual page.

Sub-commands The following subcommands are supported:

`parse` [`-H`] [`-o` *p*,*p*,...] *URI*
: Parses a given URI and displays a default list of properties. With `-H`, a header is omitted from the output. With `-o`, only properties from the list provided are displayed. The `-o` option implies output on one line, with property values separated by tabs, meant to be further processed by another command.

  Allowed property names for the `-o` option are: `uri-type`, `uri`, `path`, `mapped-dev`, `initiator`, `target`, `luname`, `hostname`, and `port`.

`map` [`-H`] [`-o` *p*,*p*,...] *URI*
: Parses a storage URI, configures the storage subsystem if necessary to instantiate a device corresponding to the URI provided, and displays the device path. If the device is already instantiated, the map operation only looks up the device path.

  For an iSCSI URI, this subcommand will add a `send-targets` discovery address(es) a hostname resolves to if a URI authority section is present.

  For logical unit and `dev` URI types, this subcommand has no effect on system configuration.

`lookup-mapping` [`-H`] [`-o` *p*,*p*,...] *URI*
: Parses a URI and looks up an existing mapping between a storage URI and the shared storage object represented by a local system device path. Default list of properties is displayed. Options `-H` and `-o` have the same meaning as for the `parse` subcommand.

`unmap` *URI*
: Parses and unmaps an object presumably mapped before. Does not display any properties or accept any options.

  For an iSCSI URI, this subcommand removes discovery addresses to which a hostname from a URI authority section resolves, if present.

For logical unit and `dev` URI types, this subcommand has no effect on system configuration.

`lookup-uri` [`-t` *uri-type*] *dev-name*
Looks up and displays URIs based on a local system device path. Allowed URI types for `-t` are `dev`, `lu`, and `iscsi`. If the `-t` option is not specified, the output consists of all URIs that match the device path for any URI type.

Examples

EXAMPLE 1 Parsing a URI and Displaying Properties

The following command parses and iSCSI URI and displays a default list of properties.

```
$ suriadm parse iscsi://10.0.0.1:3260/luname.naa.0123456789abcdef
PROPERTY          VALUE
uri-type          iscsi
hostname          10.0.0.1
port              3260
luname            naa.0123456789abcdef
```

EXAMPLE 2 Mapping an iSCSI URI and Displaying a Device Name

The following command maps an iSCSI URI and displays a mapped local system device name. The effect of this command is to automatically add a `send-targets` discovery address if one is not already present.

```
$ suriadm suriadm map iscsi://127.0.0.1/luname.naa.\
600144F0F4977D4000004F7EC8F00001
PROPERTY        VALUE
mapped-dev      /dev/dsk/c0t600144F0F4977D4000004F7EC8F00001d0s2
```

EXAMPLE 3 Looking Up Mapping

The following command looks up an existing iSCSI mapping.

```
$ suriadm lookup-mapping iscsi://127.0.0.1/\
luname.naa.600144F0F4977D4000004F7EC8F00001
PROPERTY        VALUE
mapped-dev      /dev/dsk/c0t600144F0F4977D4000004F7EC8F00001d0s2
```

EXAMPLE 4 Parsing a Logical Unit URI

The following command parses an initiator/target/`luname` logical unit URI.

```
$ suriadm parse lu:initiator.naa.2101001b32ae7ab5,\
target.naa.2100001d38089fb0,luname.naa.500000e012942880
PROPERTY        VALUE
uri-type        lu
luname          naa.500000e012942880
initiator       naa.2101001b32ae7ab5
target          naa.2100001d38089fb0
```

EXAMPLE 5 Mapping a Logical Unit URI, Looking Up URIs

The following command sequence maps a logical unit URI, then looks up the matched logical unit URIs based on a found device name.

```
$ suriadm map lu:luname.naa.5000c5000288fa25
PROPERTY        VALUE
mapped-path     /dev/dsk/c7t26d0s2

$ suriadm lookup-uri -t lu /dev/dsk/c7t26d0s2
lu:luname.naa.5000c5000288fa25
lu:initiator.naa.500605b000ae7010,target.naa.\
5001636000019c11,naa.5000c5000288fa25
```

EXAMPLE 6 Looking Up Matching URIs

The following command looks up all URIs that match a specific device name without specifying a URI type.

```
$ suriadm lookup-uri /dev/dsk/c7t26d0s2
lu:luname.naa.5000c5000288fa25
lu:initiator.naa.500605b000ae7010,target.naa.5001636000019c11,\
luname.naa.5000c5000288fa25
dev:dsk/c7t26d0s2
```

EXAMPLE 7 Parsing a URI, Displaying Selected Properties

The following command parses a URI and displays only selected properties, all on the same line, separated by tabs, and with no header.

```
$ suriadm map -Ho uri-type,luname,mapped-path \
lu:luname.naa.5000c5000288fa25
lu      naa.5000c5000288fa25    /dev/dsk/c7t26d0s2
```

EXAMPLE 8 Looking Up Logical Unit URIs

The following command looks up logical unit URIs for a device accessible by means of multiple paths.

```
$ suriadm lookup-uri -t lu /dev/dsk/c11t2000001D38089FB0d0
lu:luname.naa.2000001d38089fb0
lu:initiator.naa.2101001b32ae7ab5,target.naa.2100001d38089fb0,luname.\
naa.2000001d38089fb0
lu:initiator.naa.2100001b328e7ab5,target.naa.2200001d38089fb0,luname.\
naa.2000001d38089fb0
```

EXAMPLE 9 Trying to Parse Incorrect URI

The following command attempts to parse a syntactically incorrect URI.

```
$ suriadm parse lu:luname.naa.0123456789
Failed to parse URI "lu:luname.naa.0123456789": GUID part
in "luname.naa.GUID" not 16 or 32 character hexadecimal
```

EXAMPLE 9 Trying to Parse Incorrect URI *(Continued)*

```
number: "0123456789"
```

EXAMPLE 10 Trying to Map LU URI with Inaccessible LU

The following command attempts to map an LU URI with a logical unit name not accessible from the system.

```
$ suriadm map lu:luname.naa.0123456789abcdef
Failed to map URI "lu:luname.naa.0123456789abcdef": No such
logical unit "naa.0123456789abcdef" found
```

EXAMPLE 11 Looking Up URI for Non-Existent Device Path

The following command attempts to lookup a URI for a non-existent device path.

```
$ suriadm lookup-uri /dev/dsk/nonexistent
Failed to map "/dev/dsk/nonexistent" to URI: No such device:
"/dev/dsk/nonexistent"
```

Exit Status

`0`
Command succeeded.

`>0`
Command failed.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/library/storage/suri
Interface Stability	Committed

See Also

`stmsboot(1M)`, `svcadm(1M)`, `attributes(5)`, `suri(5)`, `scsi_vhci(7D)`

Small Computer System Interface-3 (SCSI-3)

Notes

When an iSCSI URI is used, the `svc:/network/iscsi/initiator` service must be enabled, unless a parse operation is being performed. If this service is disabled and an iSCSI URI is being processed, the iSCSI initiator service will be automatically enabled temporarily. The service is never disabled through the `suriadm` command.

Name svadm – command line interface to control Availability Suite Storage Volume operations

Synopsis

```
svadm -h

svadm -v

svadm [-C tag]

svadm [-C tag] -i

svadm [-C tag] -e {-f config_file | volume}

svadm [-C tag] -d {-f config_file | volume}

svadm [-C tag] -r {-f config_file | volume}
```

Description The `svadm` command controls the Storage Volume (SV) driver by providing facilities to enable and disable the SV driver for specified volumes, and to dynamically reconfigure the system.

Options If you specify no arguments to an `svadm` command, the utility displays the list of volumes currently under SV control. `svadm` supports the following options:

`-C` *tag*
: On a clustered node, limits operations to only those volumes belonging to the cluster resource group, or disk group name, specified by tag. This option is illegal on a system that is not clustered. The special *tag*, `local`, can be used to limit operations to only those volumes that cannot switch over to other nodes in the cluster.

`-d`
: Disables the SV devices specified on the command line or in the configuration file. If `-C` tag is specified with this option, then the volume should be in this cluster disk group.

`-e`
: Enables the SV devices specified on the command line or in the configuration file. Details of the volume are saved in the current configuration. See `dscfg(1M)`. If `-C` tag is specified with this option, then the volume should be in this cluster disk group.

`-f` *config_file*
: Specifies a configuration file that contains a list of volumes. A command reads this volume list and then perform the operation. The format of the *config_file* is a simple list of volume pathnames, one per line. Blank lines and lines starting with the comment character (#) are ignored.

`-h`
: Displays the `svadm` usage summary.

`-i`
: Displays extended status for the volumes currently under SV control.

`-r`
: When a *config_file* is specified, reconfigure the running system to match the configuration specified in the *config_file*. When the `-C` option is specified, compare the cluster tag for each

volume and change it to *cluster_tag*. If a volume is specified with this option, it is valid only to reconfigure the cluster tag associated with the volume. The `-e` or `-d` options should be used to enable or disable single volumes.

`-v`
Displays the SV version number.

Usage When an SV volume is enabled, normal system call access to the device (see `Intro(2)`) is redirected into the StoreEdge architecture software. This allows standard applications to use StorageTek features such as Sun StorageTek Point-in-Time Copy and Remote Mirror Software.

The `svadm` command generates an entry in the Availability Suite log file, `/var/adm/ds.log` (see `ds.log(4)`), when performing enable (`-e`) and disable (`-d`) operations.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	driver/storage/sv, driver/storage/sv
Interface Stability	Committed

See Also `dscfg(1M)`, `ds.log(4)`, `attributes(5)`, `sv(7D)`

Name svcadm – manipulate service instances

Synopsis

```
/usr/sbin/svcadm [-v] enable [-rst] {FMRI | pattern}...

/usr/sbin/svcadm [-v] disable [-st] {FMRI | pattern}...

/usr/sbin/svcadm [-v] restart {FMRI | pattern}...

/usr/sbin/svcadm [-v] refresh {FMRI | pattern}...

/usr/sbin/svcadm [-v] clear {FMRI | pattern}...

/usr/sbin/svcadm [-v] mark [-It] instance_state
     {FMRI | pattern}...

/usr/sbin/svcadm [-v] delegate [-s] restarter_FMRI svc|inst_FMRI
     [ svc|inst_FMRI ... ]

/usr/sbin/svcadm [-v] milestone [-d] milestone_FMRI
```

Description `svcadm` issues requests for actions on services executing within the service management facility (see `smf(5)`). Actions for a service are carried out by its assigned service restarter agent. The default service restarter is `svc.startd` (see `svc.startd(1M)`).

Options The following options are supported:

`-v` Print actions verbosely to standard output.

Subcommands

Common Operations The subcommands listed below are used during the typical administration of a service instance.

For subcommands taking one or more operands, if the operand specifies a service (instead of a service instance), and that service has only a single instance, `svcadm` operates on that instance. If an abbreviated *FMRI* (a fault management resource identifier) or pattern matches more than one service, a warning message is displayed and that operand is ignored. See `smf(5)`.

In the case that the service has more than one instance, `svcadm` return a non-zero exit status.

`enable [-rst]` {*FMRI* | *pattern*}... Enables the service instances specified by the operands. For each service instance, the assigned restarter will try to bring it to the online state. This action requires permission to modify the "general" property group of the service instance (see `smf_security(5)`).

If the `-r` option is specified, `svcadm` enables each service instance and recursively enables its dependencies.

If the `-s` option is specified, `svcadm` enables each service instance and then waits for each service instance to enter the `online` or `degraded` state. `svcadm` will return early if it determines that the service cannot reach these states without administrator intervention.

If the `-t` option is specified, `svcadm` temporarily enables each service instance. Temporary enable only lasts until reboot. This action requires permission to modify the "restarter_actions" property group of the service instance (see `smf_security(5)`). By default, `enable` is persistent across reboot.

`disable` [`-st`] {*FMRI* | *pattern*}...

Disables the service instance specified by the operands. For each service instance, the assigned restarter will try to bring it to the disabled state. This action requires permission to modify the "general" property group of the service instance (see `smf_security(5)`).

If the `-s` option is specified, `svcadm` disables each service instance and then waits for each service instance to enter the disabled state. `svcadm` will return early if it determines that the service cannot reach this state without administrator intervention.

If the `-t` option is specified, `svcadm` temporarily disables each service instance. Temporary disable only lasts until reboot. This action requires permission to modify the "restarter_actions" property group of the service instance (see `smf_security(5)`). By default, `disable` is persistent across reboot.

`restart` {*FMRI* | *pattern*}...

Requests that the service instances specified by the operands be restarted. This action requires permission to modify the "restarter_actions" property group of the service instance (see `smf_security(5)`). Restarting a service is implemented by most restarters as a full service "stop" followed by a "start".

This subcommand can restart only those services that are in the online or degraded states, as those states are defined in `smf(5)`.

`refresh` {*FMRI* | *pattern*}...

For each service instance specified by the operands, requests that the assigned restarter update the service's running configuration snapshot with the values from

the current configuration. Some of these values take effect immediately (for example, dependency changes). Other values do not take effect until the next service `restart`. See the restarter and service documentation for more information.

If the service is managed by `svc.startd(1M)`, the `refresh` method will be invoked if it exists to request the service reread its own configuration. For other restarters, see the restarter documentation.

This action requires permission to modify the "restarter_actions" property group of the service instance (see `smf_security(5)`).

`clear` {*FMRI* | *pattern*}. . .

For each service instance specified by the operands, if the instance is in the `maintenance` state, signal to the assigned restarter that the service has been repaired. If the instance is in the `degraded` state, request that the assigned restarter take the service to the `online` state. This action requires permission to modify the "restarter_actions" property group of the service instance (see `smf_security(5)`).

Exceptional Operations

The following subcommands are used for service development, management of services by higher level frameworks, and temporary administrative manipulation.

`delegate` [`-s`] *restarter_FMRI svc*|*inst_FMRI* [*svc*|*inst_FMRI* ...]

Change the restarter assignment for the given *inst_FMRI* to the restarter specified by *restarter_FMRI*. The special token `master` will set the delegated restarter to the master restarter, `svc.startd(1M)`. The special token `reset` will set the delegated restarter back to the original, file-backed restarter by removing the restarter customization. Redelegation requires a restart operation to take effect. Not all restarters support the same underlying application model, so not all potential delegations will result in a functioning service instance; see the manual page for the specific restarters involved in the operation to determine compatibility.

If the restarter does not exist or is disabled, the service instances will not be delegated and an error will be returned. If the restarter exists but is in an offline or maintenance state, the instances will be delegated but may not transition back to an online state. A warning message will be printed.

If the `-s` option is specified, `svcadm` delegates each service instance and then waits for each service instance to enter the online state, if previously online, or waits for the general/restarter property group or property to be updated. `svcadm` will return early if it determines that the service cannot reach these states without administrator intervention.

`mark` [`-It`] *instance_state* {*FMRI* | *pattern*}...

If *instance_state* is "maintenance", then for each service specified by the operands, `svcadm` requests that the assigned restarter place the service in the `maintenance` state. See `svc.startd(1M)` and `inetd(1M)` for a detailed description of the actions taken for each restarter.

If *instance_state* is "degraded", then for services specified by the operands in the online state, `svcadm` requests that the restarters assigned to the services move them into the `degraded` state.

If the `-I` option is specified, the request is flagged as immediate.

The `-t` option is only valid for maintenance requests. When this option is specified, the request is flagged as temporary, and its effect will only last until the next reboot.

`milestone` [`-d`] *milestone_FMRI*

If *milestone_FMRI* is the keyword "none", all services other than the master restarter, `svc:/system/svc/restarter:default`, will be temporarily disabled.

If *milestone_FMRI* is the keyword "all", temporary enable and disable requests for all services will be nullified.

If *milestone_FMRI* is one of the following:

```
svc:/milestone/single-user:default
svc:/milestone/multi-user:default
svc:/milestone/multi-user-server:default
```

then temporary enable and disable requests for the indicated service and all services it depends on (directly or indirectly) will be nullified. All other services will be temporarily disabled.

Changing the system's current milestone with the "milestone" subcommand will not change the current run level of the system. To change the system's run level, invoke `/usr/sbin/init` directly.

This action requires permission to modify the "options_ovr" property group of the `svc:/system/svc/restarter:default` service instance (see `smf_security(5)`).

The `-d` option immediately changes the milestone to the requested milestone, as above. Additionally, it makes the specified milestone the default boot milestone, which persists across reboot. The default milestone is defined by the `options/milestone` property on the master restarter, `svc:/system/svc/restarter:default`. If this property is absent, "all" is the default. This action requires permission to modify the "options" property group of the `svc:/system/svc/restarter:default` service instance (see `smf_security(5)`).

Operands The following operands are supported:

FMRI An *FMRI* that specifies one or more instances. *FMRI*s can be abbreviated by specifying the instance name, or the trailing portion of the service name. For example, given the *FMRI*:

```
svc:/network/smtp:sendmail
```

All the following are valid abbreviations:

```
sendmail
:sendmail
smtp
smtp:sendmail
network/smtp
```

While the following are invalid:

```
mail
network
network/smt
```

If the *FMRI* specifies a service, then the command applies to all instances of that service. Abbreviated forms of *FMRI*s are unstable, and should not be used in scripts or other permanent tools.

pattern A pattern that is matched against the *FMRI*s of service instances according to the "globbing" rules described by `fnmatch(5)`. If the pattern does not begin with "svc:", then "svc:/" is prepended.

If an abbreviated *FMRI* or pattern matches more than one service, a warning message is displayed and that operand is ignored.

Examples

EXAMPLE 1 Restarting a Service Instance

The following command restarts the NFS server. The full *FMRI* for the default service instance is: `svc:/network/nfs/server:default`

However, you can abbreviate the full *FMRI* as follows:

```
# svcadm restart nfs/server
```

EXAMPLE 2 Disabling the Standard HTTP Server

The following command disables the standard HTTP server, using an abbreviated *FMRI*:

```
$ svcadm disable http
```

EXAMPLE 3 Enabling an Instance and Its Dependent Instances

The following command enables the `foo:bar` instance, and all instances on which it depends:

```
$ svcadm enable -r foo:bar
```

EXAMPLE 4 Synchronously enabling an instance

The following command enables the `foo:bar` instance. The command will not return until the instance comes online or `svcadm` determines it is not possible for the service to come online.

```
$ svcadm enable -s foo:bar
```

EXAMPLE 5 Restricting and Restoring the Running Services

The following command restricts the running services to single user mode:

```
# svcadm milestone milestone/single-user
```

The following command restores the running services:

```
# svcadm milestone all
```

Exit Status

The following exit values are returned:

`0` Successful completion.

`1` A fatal error occurred. One or more error messages are displayed on standard error.

`2` Invalid command line options were specified.

`3` `svcadm` determined that a service instance that it was waiting for could not reach the desired state without administrator intervention due to a problem with the service instance itself.

`4` `svcadm` determined that a service instance that it was waiting for could not reach the desired state without administrator intervention due to a problem with the service's dependencies.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os
Interface Stability	See below.

The interactive output is Uncommitted. The invocation and non-interactive output are Committed.

See Also svcprop(1), svcs(1), inetd(1M), init(1M), svccfg(1M), svc.startd(1M), libscf(3LIB), contract(4), attributes(5), smf(5), smf_security(5)

Notes The amount of time svcadm will spend waiting for services and their dependencies to change state is implicitly limited by their method timeouts. For example, a service using the default restarter whose start method hangs will be transitioned to the maintenance state when its timeout expires. svcadm will then consider it impossible for this service to come online without administrator intervention.

Attempts to synchronously enable a service which depends (directly or indirectly) on a file may fail with an exit status indicating that dependencies are unsatisfied if the caller does not have the privileges necessary to search the directory containing the file. This limitation may be removed in a future Solaris release.

Name svcbundle – create an SMF service bundle

Synopsis

```
svcbundle [-i | -o output_file] -s name=value...

svcbundle help [name]
```

Description The `svcbundle` command is used to generate SMF manifests. The manifest is specified by multiple `-s` options. To generate a manifest, you must specify `service-name` and `start-method`. The other NV pairs are optional and allow the user to specify more details of the service. The `svccfg(1M)` validate command will be run on the generated manifest to detect any templating conflicts.

You can also use `svcbundle` to generate profiles. See *bundle-type* below.

The second synopsis prints a help message on standard out listing all legal names. Alternatively, you can specify help and a name to see a discussion of legal values for that name.

In order to reduce the burden on the user, `svcbundle` makes several simplifying assumptions when generating a manifest. You can edit the generated manifest if these assumptions are not correct for your application:

- The generated manifest is intended to be used with the master restarter, `svc.startd(1M)`.
- A dependency on `svc:/milestone/multi-user` will be generated to keep the service from starting too early. See `rc-script` below for an exception to this.
- The bundle name will be the same as the value provided for `service-name`.
- Timeouts for all `exec_methods` will be 60 seconds.

By using the `-i` option, you can get `svcbundle` to do much of the work of installing the service. It will automatically save the generated manifest in `/lib/svc/manifest/site` or the profile in `/etc/svc/profile/site`. The name of the generated file will be the basename of the service name that is specified with `-s` *service-name*, and the file will have an `.xml` extension. Warning: `svcbundle` will overwrite any existing file with that name.

`svcbundle` will then restart the `manifest-import` service to process the newly created file and incorporate it into SMF. In the manifest case, `svcbundle` will then wait for the service to enter a final state -- one of online, disabled, or maintenance. At the start of this wait period, `svcbundle` will print:

```
svcbundle: waiting for service to reach final_state state
```

...where *final_state* is enabled or disabled. It is safe to interrupt `svcbundle` after this message appears.

Clearly, to use the `-i` option, you need sufficient authorizations to create the file in these restricted directories and to restart the `manifest-import` service. See `smf_security(5)`.

Extended Description

This section discusses name/value (NV) pairs.

The generated bundle is entirely defined by the use of multiple -s options on the command line. Each NV pair is of the form *name=value* where *name* and *name* come from this list:

`bundle-type`
Type of service bundle to generate. Legal values are `manifest` and `profile`. `manifest` is the default.

`duration`
Synonym for model.

`enabled`
Indicates whether or not the instance should be enabled. Legal values are `true` and `false`. The default value is `true`.

`model`
Sets the service model. This is the value of the `startd/duration` property. Refer to `svc.startd(1M)`. Model can be set to one of the following values:

- `contract`
- `daemon` — synonym for `contract`
- `child`
- `wait` — synonym for `child`
- `transient`

The default is `transient`.

`instance-name`
Name of the instance. The default value is `default`.

`instance-property=`*pg_name*`:`*prop_name*`:`*prop_type*`:`*value*
`service-property=`*pg_name*`:`*prop_name*`:`*prop_type*`:`*value*
These options are used to create a property group named *pg_name* in the instance or service, and it will have a type of application. The PG will have a single property named *prop_name* with a single value that is of type *prop_type*. Property groups with multiple properties can be created by invoking `*-property` multiple times. Zero or more `*-property=` declarations can be used.

The property type can be defaulted by using two consecutive colons. See "Examples," below. For manifests, a default property type of `astring` will be used. Profiles do not require that the property be specified, since it can usually be determined from other sources of information.

`rc-script=`*script_path*`:`*run_level*
This NV pair causes `svcbundle` to emit a manifest that facilitates conversion of a legacy `rc` script to an SMF service. *script_path* is the path to the `rc` script and *run_level* is the run level (see `init(1M)`) where the `rc` script runs. *script_path* is used to generate the start and stop `exec_method` elements in the manifest. The exec attribute will be set to:

script_path `%m`

run_level is used to generate dependencies, so that the script runs at the appropriate time during booting.

`refresh-method`
The command to execute when a service is refreshed. Whitespace is allowed in the value. The value can include method tokens introduced by a percent sign (`%`), as documented in `smf_method(5)`. The default value is `:true`.

`service-name`
Name of the service. This NV pair is required.

`start-method`
The command to execute when the service is started. Whitespace is allowed in the value. The method tokens that are introduced by `%` as documented in `smf_method(5)` are allowed and will be placed in the manifest for expansion by the restarter. `:true` is allowed. This NV pair is required for manifests unless the *rc-script* NV pair is specified. It is not required for profiles.

`stop-method`
The command to execute when the service is stopped. It accepts values like start-method and also accepts `:kill`. `:true` is the default value for transient services and `:kill` for contract and child services.

Options The following command-line options are supported:

`-i`
Install the generated file. See "Description" for details.

`-o` *output_file*
Specifies the name of the file to be created.

`-s` *name=value*
Specifies a name/value pair. See "Extended Description" for details.

If neither `-i` nor `-o` are specified, the generated file will be written to stdout.

Examples **EXAMPLE 1** Creating Manifest for Transient Service

The following command creates a manifest for a simple transient service. Since transient services are the default for `svcbundle`, you can specify the manifest with just two options.

```
# svcbundle -s service-name=site/sneezy \
-s start-method=/lib/svc/method/sneezy
```

EXAMPLE 2 Creating Manifest for Daemon Service

The following command creates a manifest for a daemon service.

```
# svcbundle -s service-name=site/sneezy \
-s start-method=/lib/svc/method/sneezy \
```

EXAMPLE 2 Creating Manifest for Daemon Service *(Continued)*

```
-s model=daemon
```

EXAMPLE 3 Creating Manifest for Daemon Service with Stop and Refresh Commands

The following command creates a manifest for a daemon service with stop and refresh commands.

```
# svcbundle -s service-name=site/sleepy \
-s start-method="/lib/svc/method/sleepy %m" \
-s stop-method="/lib/svc/method/sleepy %m" \
-s refresh-method="/lib/svc/method/sleepy %m" \
-s model=daemon
```

EXAMPLE 4 Creating Manifest with Instance Properties

The following command creates a manifest with instance properties.

```
# svcbundle -s service-name=system/happy \
-s start-method=/lib/svc/method/happy \
-s instance-property=config:velocity:count:50 \
-s instance-property=config:color:astring:red
```

The generated manifest will create a service instance with a config property group containing two properties, `velocity` and `color`.

EXAMPLE 5 Creating Manifest for `rc` Script Conversion

The following command creates a manifest to assist in converting an `rc` script. This example assumes that the `rc` script runs at the multiuser level and does not start a daemon.

```
# svcbundle -s service-name=doc \
-s rc-script=/etc/init.d/doc:2
```

The conscientious user will modify the `rc` script to include `/lib/svc/share/smf_include.sh`.

EXAMPLE 6 Generating Profile to Modify Service Property

The following command generates a profile to set the `nfsmapid_domain` property of the `grumpy` service.

```
# svcbundle -s bundle-type=profile \
-s service-name=network/nfs/grumpy \
-s service-property=nfs-props:nfsmapid_domain:astring:grumpy
```

EXAMPLE 7 Using the Default Property Type

The following command sets the property type for config/color defaults to `astring`.

EXAMPLE 7 Using the Default Property Type *(Continued)*

```
# svcbundle -s service-name=system/happy \
-s start-method=/lib/svc/method/happy \
-s instance-property=config:color::red
```

EXAMPLE 8 Installing the Manifest

The following command uses the -i option to install the manifest. In this example the generated manifest will be written to `/lib/svc/manifest/site/bashful.xml`, since `bashful` is the basename of the service name.

```
# svcbundle -i -s service-name=application/bashful \
-s start-method=/opt/bashful/start
svcbundle: waiting for application/bashful to reach enabled state
```

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/
Interface Stability	Committed

See Also

`svcs(1)`, `init(1M)`, `svc.startd(1M)`, `svccfg(1M)`, `service_bundle(4)`, `attributes(5)`, `smf_method(5)`, `smf_security(5)`, `smf_template(5)`

Name svccfg – import, export, and modify service configurations

Synopsis

```
/usr/sbin/svccfg [-v] [-s FMRI]

/usr/sbin/svccfg [-v] [-s FMRI] subcommand [args]...

/usr/sbin/svccfg [-v] [-s FMRI] -f command-file
```

Description The `svccfg` command manipulates data in the service configuration repository. `svccfg` can be invoked interactively, with an individual subcommand, or by specifying a command file that contains a series of subcommands.

Changes made to an existing service in the repository typically do not take effect for that service until the next time the service instance is refreshed. See the `refresh` subcommand, below, or the `refresh` subcommand in the `svcadm(1M)` man page for more details.

Options The following options are supported:

`-f` *command-file*
: Reads and executes `svccfg` subcommands from *command-file*.

`-s` *FMRI*
: Selects the entity indicated by *FMRI* (a fault management resource identifier) before executing any subcommands. If `-f` *command-file* is not provided and no subcommands are specified on the command line, then masked entities will be treated as nonexistent. See `smf(5)`.

`-v`
: Verbose.

Subcommands Subcommands are divided into the categories specified in the subsections that follow.

All subcommands that accept *FMRI*s also accept abbreviated or globbed patterns. Instances and services can be abbreviated by specifying the instance name, or the trailing portion of the service name. For example, given the *FMRI*:

```
svc:/network/smtp:sendmail
```

All the following are valid abbreviations:

```
sendmail
:sendmail
smtp
smtp:sendmail
network/smtp
```

While the following are invalid:

```
mail
network
network/smt
```

Abbreviated forms of *FMRI*s are unstable, and should not be used in scripts or other permanent tools. If a pattern matches more than one instance or service, an error message is printed and no action is taken.

General Subcommands

`end`
`exit`
`quit`
Exits immediately.

`repository` [`-p` *prefix*] *repfile*
Uses *repfile* as a repository. By default, `svccfg`(1M) uses the system repository.

Use repository only with files from the identical version of Solaris, including patches, that you are currently running. Do not use this subcommand with the system repository, `/etc/svc/repository.db`.

If you use `svccfg repository` to pre-populate the SMF repository before deployment time, use `-p` option to specify the root prefix for the system standard location for manifests imported with `import`. This prefix will be replaced by `/lib/svc/manifest` and `/var/svc/manifest` once the repository is on a live system. If manifests from your `-p` directory do not appear in a system standard location at runtime, the services associated with them will be removed.

`set` [`-v`|`-V`]
Sets optional behavior. If no options are specified, set displays the options currently in effect.

`-v`
Turns on verbose mode.

`-V`
Turns off verbose mode.

Service Manifest and Profile Subcommands

`apply` [`-n`] [`-v`] *file* | *directory*
If the argument is a service profile or manifest, apply the configuration to the admin layer of the SMF repository. Services, instances, property groups, and properties will be created as necessary.

If the type attribute of a property or property group is unspecified, an attempt will be made to determine the type from existing type settings or from the service template. If a type cannot be determined, a warning will be presented and the service will be skipped so inconsistent data will not be introduced into a service and instance. Nonexistent services and instances are ignored.

To use the relaxed element definitions in a profile, the following definitions need to be added to the `DOCTYPE` entry:

```
<!ENTITY % profile "INCLUDE">
<!ENTITY % manifest "IGNORE">
```

Services and instances modified by the profile will be refreshed. If `-n` is specified, the profile is processed and no changes are applied to the SMF repository. Any syntax error found will be reported on stderr and an exit code of `1` will be returned. See `smf(5)` for a description of service profiles. This command requires privileges to modify properties in the service and instance. See `smf_security(5)` for the privileges required to modify properties.

Services and instances in the manifest or profile will be validated against template data in the manifest and the repository, and warnings will be issued for all template violations. See `smf_template(5)` for a description of templates. If the `-V` option is specified, manifests that violate the defined templates will fail to import. In interactive invocations of `svccfg`, `-V` is the default behavior.

If the argument to `apply` is a directory, all profiles found under that directory tree will get applied as described above. The subcommand fails if a specified file or any file found under a specified directory is not a service profile.

`extract` [`-a`] [`-l` *layer*] [*fmri*] [> *file*]
Displays a service profile for the specified FMRI or the whole system if an FMRI is not specified.

If `-l` is supplied, a list of layers can be selected from which to extract values. The `-l` option requires a layer name and takes the arguments: `manifest`, `system-profile`, `site-profile`, `admin`, `current`, `all`. `current` and `all` are synonyms, and select the highest-layer values. Multiple layers can be comma-separated or specified with multiple `-l` options.

If `-l` is not supplied, the default is `-l admin,site-profile`.

If a property is defined in multiple selected layers, only the highest layer is exported in the profile.

Without the `-a` option, property groups containing protected information (identified by the presence of the `read_authorization` property—see `smf_security(5)`) will be extracted without their property values. When the `-a` option is specified, all values will be extracted. An error results if there are insufficient privileges to read these values.

If an FMRI is given and that FMRI is a service, the profile will contain customizations only for that service and the instances of the service. If the provided FMRI is an instance, the profile will contain customizations for the service and the instance provided.

`export` [`-a`] *service_FMRI* [>*file*]
Running `svccfg export` is equivalent to:

```
svccfg extract -l current [-a] service_FMRI [>file]
```

`import` [`-V`] [*file* | *directory*]
`svccfg import` on a file in a system-managed filesystem location (subdirectories of `/lib/svc/manifest` and `/var/svc/manifest`) invokes: `svcadm restart manifest-import`.

Placing your manifests in a system-managed location and invoking `svcadm restart manifest-import` to import them is the recommended practice.

`svccfg import` on files in other locations imports their properties as administrative customization into the admin layer. It is equivalent to:

`svccfg apply [`*file* `|` *directory*`]`

`inventory` *file*
If *file* is determined to be a service manifest or profile, then the FMRIs of the services and instances the *file* describes are printed. For each service, the FMRIs of its instances are displayed before the FMRI of the service.

`validate` [*file* | *fmri*]
The `validate` subcommand can operate on a manifest file, an instance FMRI, or the current instance or snapshot entity selection. When an argument is specified, `svccfg` will check to see whether the specified file exists. If the file exists, it will be validated. If a file of the specified name does not exist, the argument is treated as an FMRI pattern. If a conflict arises between a filename and an FMRI, use the `svc:` and `file:` prefixes to tell `svccfg` how to interpret the argument.

When you specify a file, the file is processed in a manner similar to `import -V`, but no changes are made to the repository. If any errors are detected, `svccfg` displays the errors and exits with a nonzero exit status.

For an instance *fmri*, instance entity selection, or snapshot entity selection, the specified instance in its composed form (see "Properties and Property Groups" in `smf(5)`) will be validated against template data in the repository. Instance FMRIs and instance entity selections use the "running" snapshot for validation. Warnings will be issued for all template violations. See `smf_template(5)` for a description of templates.

Entity Selection, Modification, and Navigation Subcommands

An "entity" refers to a scope, service, or service instance.

`add` *name*
A new entity with the given name is created as a child of the current selection. See `smf_security(5)` for the privileges required to create entities.

`delete` [`-f`] {*name* | *fmri*}
The named child of the current selection or the entity specified by *fmri* is deleted. Attempts to delete service instances in the "online" or "degraded" state will fail unless the `-f` flag is specified. If a service or service instance has a "dependents" property group of type "framework", then for each of its properties with type "astring" or "fmri", if the property has a single value which names a service or service instance then the dependency property group in the indicated service or service instance with the same name as the property will be deleted. See `smf_security(5)` for the privileges required to delete service configurations.

Invoking the `delete` subcommand with an FMRI that identifies a service with a manifest in a standard location only masks, and does not delete, that service's definition. To delete a service, you must delete its manifest, then restart the manifest-import service with the following command:

```
# svcadm restart manifest-import
```

Note that reimporting a manifest does not remove a mask.

Use the `liscust` subcommand with the `-M` option to list masked services. See `EXAMPLES` for an example of unmasking a service.

See `smf(5)` for a description of the Oracle Solaris service management facility.

`list` [*pattern*]
The child entities of the current selection whose names match the glob pattern *pattern* are displayed (see `fnmatch(5)`). ':properties' is also listed for property-bearing entities, namely services and service instances.

`refresh`
Commit the values from the current configuration to the running snapshot, making them available for use by the currently selected instance. If the repository subcommand has not been used to select a repository, direct the instance's restarter to reread the updated configuration. If the selection is a service, all instances of the service will be refreshed.

`select` {*name* | *fmri*}
If the argument names a child of the current selection, it becomes the current selection. Otherwise, the argument is interpreted as an FMRI and the entity that the argument specifies becomes the current selection.

`unselect`
The parent of the current selection becomes the current selection.

Property Inspection and Modification Subcommands

`addpg` *name type* [*flags*]
Adds a property group with the given *name* and type to the current selection. *flags* is a string of characters which designates the flags with which to create the property group. 'P' represents SCF_PG_FLAG_NONPERSISTENT (see `scf_service_add_pg(3SCF)`). See `smf_security(5)` for the privileges required to create property groups.

`addpropvalue` *pg/name* [*type:*] *value*
Adds the given value to a property. If *type* is given and the property exists, then if *type* does not agree with the property's *type*, the subcommand fails. If the *pg* does not exist, addpropvalue will create one if it can find the *pg* type and flags in the template definitions. If the selection is an instance, `addpropvalue` will look for the *pg* type and flags in the service before looking up the template definitions. If no *pg* type and flags are found, the subcommand will fail. The values may be enclosed in double-quotes. String values containing double-quotes or backslashes must be enclosed by double-quotes and the contained double-quotes and backslashes must be quoted by backslashes. Nonexistent properties are created, in which case the *type* specifier must be present. See

scf_value_create(3SCF) for a list of available property types. See smf_security(5) for the privileges required to modify properties. The new value will be appended to the end of the list of property values associated with the property.

`delcust` [`-M`] [*pattern*]
Delete any administrative customizations for the current selection. If an argument is supplied, it is taken as a glob pattern and only property groups and properties with names that match the argument are deleted.

If there is no current selection, no changes are made and the subcommand fails.

If `-M` is supplied, delete only masked entities.

To see what customizations `delcust` would remove, use `listcust` with the same options. As `delcust` can potentially remove all administrative customizations on the current selection, always run `listcust` first to determine you are removing what you intend to.

`delpg` *name*
Deletes the property group *name* of the current selection. See smf_security(5) for the privileges required to delete property groups.

If the property group is backed by a manifest or profile, it is masked. See smf(5).

`delprop` *pg*[*/name*]
Deletes the named property group or property of the current selection. See smf_security(5) for the privileges required to delete properties.

`delpropvalue` *pg/name globpattern*
Deletes all values matching the given *glob* pattern in the named property. Succeeds even if no values match. See smf_security(5) for the privileges required to modify properties.

`describe` [`-v`] [`-t`] [*propertygroup*/*property*]
Describes either the current or the possible settings.

When invoked without arguments, `describe` gives basic descriptions (if available) of the currently selected entity and all of its currently set property groups and properties. A property group or specific property can be queried by specifying either the property group name, or the property group name and property name, separated by a slash (`/`), as an argument.

The `-v` option gives all information available, including descriptions for current settings, constraints, and other possible setting choices.

The `-t` option shows only the template data for the selection (see smf_template(5)), and does not display the current settings for property groups and properties.

`editprop` [`-a`]
Comments of commands to reproduce the property groups and properties of the current selection are placed in a temporary file and the program named by the `VISUAL` environment variable is invoked to edit it. If `VISUAL` is not defined, `EDITOR` is used instead. If both

environment variables are not defined, then the default editor vi(1) is used. Upon completion, the commands in the temporary file are executed. See smf_security(5) for the privileges required to create, modify, or delete properties.

By default editprop will not display SMF infrastructure property groups such as framework, dependency, templates, firewall, and notification parameters or properties templated with visibility hidden. If an instance is selected, the composed view of the properties are placed in the temporary file. The -a option will place all properties in the temporary file, including properties in SMF infrastructure property groups and those templated with visibility hidden.

listpg [*pattern*]
Displays the names, types, and flags of property groups of the current selection. If an argument is given, it is taken as a glob pattern and only property groups with names which match the argument are listed.

In interactive mode, a basic description of the property groups is also given.

listprop [-l *layer*...] [-f | -o *format*] [*pattern*]
Lists property groups and properties of the current selection. For property groups, names, types, and flags are listed. For properties, names (prepended by the property group name and a slash (/)), types, and values are listed. See scf_value_create(3SCF) for a list of available property types. If an argument is supplied it is taken as a glob pattern and only property groups and properties with names which match the argument are listed.

With the -l option, print the layer the value came from. The -l option requires a layer, and takes the arguments: manifest, system-profile, site-profile, admin, current, all. current prints the same property values as listprop without -l, along with the layer that value was defined in.

The -f and -o options are mutually exclusive. -f prints the file, if any, a property came from. -o allows field selection. Selectable fields include:

propname	the property name
pgname	the property group name
instname	the instance name
servicename	the service name
layer	the layer
proptype	the property type
value	the property value
file	the source file
masked	whether the property group or property is currently masked
time	the time this property last changed

`listcust` [`-L`] [`-M`] [*pattern*]
Print all admin layer customizations and masked entities for the current selection. If an argument is supplied, it is taken as a glob pattern and only property groups and properties with names that match the argument are listed.

If there is no current selection, list all customizations for all services.

If `-M` is supplied, print only masked entities.

If `-L` is supplied, show all local customizations, including those in the site profile layer in addition to those in the admin layer.

`setenv` [`-i` | `-s`] [`-m` *method_name*] *envvar value*
Sets a method environment variable for a service or instance by changing the "environment" property in the *method_name* property group, if that property group has type "method". If *method_name* is not specified and the `-i` option is used, the "method_context" property group is used, if an instance is currently selected. If the `-s` option is used and a service is currently selected, its "method_context" property group is used. If the `-s` option is used and an instance is currently selected, the "method_context" property group of its parent is used. If neither the `-i` option nor the `-s` option is used, the "start" property group is searched for in the currently selected entity and, if an instance is currently selected, its parent is also searched. If the "inetd_start" property group is not located, it is searched for in a similar manner.

Once the property is located, all values which begin with *envvar* followed by a "=" are removed, and the value "*envvar=value*" is added. See `smf_security(5)` for the privileges required to modify properties.

`setprop` *pg/name* = [[*type:*] *value*]
setprop *pg/name* = [*type:*] ([*values ...*])
Sets the *name* property of the *pg* property group of the current selection to the given values of type *type*. See `scf_value_create(3SCF)` for a list of available property types. If the pg does not exist setprop will create one if it can find the pg type and flags in the template definitions. If the selection is an instance, setprop will look for the pg type and flags in the service before looking up the template definitions. If no pg type and flags are found, the subcommand will fail. If the named property does not exist, it is created, as long as the type is specified. If the property already exists and the type disagrees with the existing type on the property, the subcommand fails. If no type and no value are provided, setprop will delete all values for the pg/name. Values may be enclosed in double-quotes. String values which contain double-quotes or backslashes must be enclosed by double-quotes and the contained double-quotes and backslashes must be quoted by backslashes. Multiple values will be stored in the order in which they are specified. See smf_security(5) for the privileges required to create or modify properties.

`unsetenv` [`-i` | `-s`] [`-m` *method_name*] *envvar*
Removes a method environment variable for a service or instance by changing the "environment" property in the *method_name* property group, if that property group has

type “method”. If *method_name* is not specified and the `-i` option is used, the “method_context” property group is used, if an instance is currently selected. If the `-s` option is used and a service is currently selected, its “method_context” property group is used. If the `-s` option is used and an instance is currently selected, the “method_context” property group of its parent is used. If neither the `-i` option nor the `-s` option is used, the “start” property group is searched for in the currently selected entity and, if an instance is currently selected, its parent is also searched. If the “inetd_start” property group is not located, it is searched for in a similar manner.

Once the property is located, all values which begin with *envvar* followed by “=” are removed. See `smf_security(5)` for the privileges required to modify properties.

Notification Parameters Subcommands

`setnotify` {[`-g`] *tset* | *class*} *notification_parameters*

Sets notifications parameters for software events and Fault Management problem lifecycle events in the SMF repository.

`-g`

Used to set system-wide notification parameters for SMF state transition. See `smf(5)`. These notification parameters are set in `svc:/system/svc/global:default` regardless of any `svccfg` current selection. This subcommand refreshes all instances it modifies.

class

Comma-separated list of FMA Event classes or aliases. See `smf(5)` Notification Parameters.

tset

Comma-separated list of SMF state transitions. See `smf(5)` Notification Parameters.

notification_parameters

URI format for each notification mechanism implemented: For SMTP use:

`mailto:`*addr*`[?`*header1*`=`*value1*`[&`*header2*`=`*value2*`]]`

...or:

`mailto:{[active]|inactive}`

...and for SNMP traps use:

`snmp:{[active]|inactive}`

The parameter `msg_template` defined in `smtp-notify(1M)` can be set as a header value in the `mailto` URI. For example:

`mailto:root@localhost?msg_template=<`*path to template file*`>`

SNMP traps are directed to the host as defined by the `trapsink` directive in `/etc/net-snmp/snmp/snmpd.conf` or as specified by the SNMP trap notification daemon. See `smtp-notify(1M)`.

The notification parameters are specific to the class or tset specified and overwrite preexisting notification parameters. The active/inactive form does not overwrite

previous notification parameters. It just switches on or off the notification mechanism for the specified class or tset. Setting notification parameters implicitly sets them as active.

`listnotify` [`-g`] [*tset*] | *class*
Displays the existing notification parameters for the specified class or tset. With the `-g` option, the notification parameters in `svc:/system/svc/global:default` are displayed. If tset is omitted, `all` is implied.

`delnotify` [`-g`] *tset* | *class*
Delete the existing notification parameters for the specified class or tset. With the `-g` option, the notification parameters in `svc:/system/svc/global:default` are deleted.

Snapshot Navigation and Selection Subcommands

`listsnap`
Displays snapshots available for the currently selected instance.

`revert` [*snapshot*]
Reverts the administrative customizations of the currently selected instance and its service to those recorded in the named snapshot. If no argument is given, use the currently selected snapshot and deselect it on success. The changed property values can be made active via the `refresh` subcommand of `svcadm(1M)`. See `smf_security(5)` for the privileges required to change properties.

`selectsnap` [*name*]
Changes the current snapshot to the one named by *name*. If no *name* is specified, deselect the currently selected snapshot. Snapshots are read-only.

Instance Subcommands

`refresh`
Commit the values from the current configuration to the running snapshot, making them available for use by the currently selected instance. If the repository subcommand has not been used to select a repository, direct the instance's restarter to reread the updated configuration.

Examples

EXAMPLE 1 Importing a Service Description

The following example imports a service description for the `seismic` service in the XML manifest specified on the command line.

```
# svccfg import /var/svc/manifest/site/seismic.xml
```

Note that the manifest must follow the format specified in `service_bundle(4)`.

EXAMPLE 2 Exporting a Service Description

To export a service description on the local system:

```
# svccfg export dumpadm >/tmp/dump.xml
```

EXAMPLE 3 Deleting a Service Instance

To delete a service instance:

```
# svccfg delete network/inetd-upgrade:default
```

EXAMPLE 4 Checking Properties in an Alternate Repository

To examine the state of a service's properties after loading an alternate repository, use the sequence of commands shown below. One might use such commands, for example, to determine whether a service was enabled in a particular repository backup.

```
# svccfg
svc:> repository /etc/svc/repository-boot
svc:> select telnet:default
svc:/network/telnet:default> listprop general/enabled
general/enabled  boolean  false
svc:/network/telnet:default> exit
```

EXAMPLE 5 Enabling Debugging

To modify `LD_PRELOAD` for a start method and enable the use of `libumem(3LIB)` with debugging features active:

```
$ svccfg -s system/service setenv LD_PRELOAD libumem.so
$ svccfg -s system/service setenv UMEM_DEBUG default
```

EXAMPLE 6 Using `describe` Subcommand

The following command illustrates the use of the `describe` subcommand.

```
# svccfg -s console-login describe ttymon
ttymon                      application
ttymon/device               astring  /dev/console
   terminal device to be used for the console login prompt
ttymon/label                astring  console
   appropriate entry from /etc/ttydefs
   ...
```

EXAMPLE 7 Configuring Notification Preferences

The following command configures notification preferences for SMF service state transition events.

```
# svccfg setnotify -g from-online,to-maintenance \
mailto:admin@somehost.com
```

EXAMPLE 8 Enabling SNMP Notifications

The following command enables SNMP notifications for Fault Management events.

```
# svccfg setnotify problem-diagnosed,problem-updated \
mailto:admin@somehost.com snmp:
```

EXAMPLE 9 Listing Notification Settings

The following command lists notification settings for Fault Management events.

```
# svccfg listnotify problem-diagnosed,problem-updated
Event: problem-diagnosed
  Notification Type: smtp
    active: true
   to: admin@somehost.com
  Notification Type: snmp
    active: true

Event: problem-updated
  Notification Type: smtp
    active: true
   to: admin@somehost.com
  Notification Type: snmp
    active: true
```

EXAMPLE 10 Unmasking a Service

The following sequence of commands shows the existence of the service `mysvc`, that `mysvc` is masked, and finally unmasks the service.

```
$ svcs -l mysvc
fmri         svc:/system/mysvc:default
name         Manifest to test snapshots
enabled      true
state        online
next_state   none
state_time   January 13, 2012 09:42:55 AM MST
logfile      /var/svc/log/system-mysvc:default.log
restarter    svc:/system/svc/restarter:default
manifest     /lib/svc/manifest/test/mysvc.xml
dependency   require_all/none svc:/system/filesystem/local (online)

[ Note manifest file in standard location. ]

# svccfg delete -f mysvc
$ svcs mysvc
svcs: Pattern 'mysvc' doesn't match any instances
STATE          STIME    FMRI

[ Not listed because service is masked. ]

$ svccfg listcust -M | grep mysvc
svc:/system/mysvc manifest MASKED
  manifestfiles/lib_svc_manifest_test_mysvc_xml astring     admin \
         MASKED /lib/svc/manifest/test/mysvc.xml
svc:/system/mysvc:default manifest MASKED
```

EXAMPLE 10 Unmasking a Service *(Continued)*

```
[ First line, above, shows that service is masked.  Masking is
  propagated down, so the instance is also masked as shown in
  the last line. ]

# svccfg -s svc:/system/mysvc delcust
 Deleting customizations for service: system/mysvc
$ svcs mysvc
STATE          STIME    FMRI
online          9:48:25 svc:/system/mysvc:default

[ Masking has been removed. ]
```

EXAMPLE 11 Setting a Multi-Value Property

The following command sets a multi-value property. Note that the property is enclosed in single quotes so that the parentheses and double quotes do not need to be escaped.

```
# svccfg -s svc:/stooges setprop foo/bar = astring: \
'("moe" "Curly" "Larry")'
```

EXAMPLE 12 Clearing All Values from a Property Using `setprop`

The following command uses `setprop` to clear all values from a property.

```
# svccfg -s svc:/stooges setprop foo/bar =
```

EXAMPLE 13 Clearing All Values from a Property Using `delpropvalue`

The following command uses `delpropvalue` to clear all values from a property.

```
# svccfg -s svc:/stooges delpropvalue foo/bar \*
```

EXAMPLE 14 Setting Property with Embedded Whitespace

The following command shows the correct use of quotation marks when setting a property whose value is a string with embedded whitespace.

```
# svccfg -s svc:inst setprop pg/prop = \
astring '"without single quotes, shells eat double quotes"'
```

Environmental Variables

EDITOR

The command to run when the `editprop` subcommand is used. The default editor is `vi(1)`.

Exit Status

The following exit values are returned:

0

Successful execution.

1

One or more subcommands resulted in failure. Error messages are written to the standard error stream.

2

Invalid command line options were specified.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os
Interface Stability	See below.

The interactive output is Uncommitted. The invocation and non-interactive output are Committed.

See Also `svcprop(1)`, `svcs(1)`, `smtp-notify(1M)`, `svcadm(1M)`, `svc.configd(1M)`, `libscf(3LIB)`, `libumem(3LIB)`, `scf_service_add_pg(3SCF)`, `scf_value_create(3SCF)`, `contract(4)`, `service_bundle(4)`, `attributes(5)`, `fnmatch(5)`, `smf(5)`, `smf_method(5)`, `smf_security(5)`, `smf_template(5)`

Name svc.configd – Service Management Facility repository daemon

Synopsis

```
/lib/svc/bin/svc.configd
```

Description `svc.configd` is the repository daemon for the Service Management Facility. `svc.configd` is invoked automatically during system startup, and restarted if any failures occur. `svc.configd` should never be invoked directly.

Interaction with `svc.configd` is by way of `libscf(3LIB)` and the command line tools:`svcs(1)`, `svcprop(1)`, `svcadm(1M)`, and `svccfg(1M)`.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also `svcs(1)`, `svcprop(1)`, `svcadm(1M)`, `svccfg(1M)`, `libscf(3LIB)`, `attributes(5)`

Name svc.ipfd – IP Filter firewall monitoring daemon

Synopsis

```
/lib/svc/bin/svc.ipfd

svc:/network/ipfilter:default
```

Description The `svc.ipfd` daemon monitors actions on services that use firewall configuration and initiates update services' IP Filter configuration. The daemon allows the system to react to changes in system's firewall configuration in an incremental fashion, at a per-service level.

A service's firewall policy is activated when it is enabled, deactivated when it is disabled, and updated when its configuration property group is modified. `svc.ipfd` monitors the services management facility (SMF) repository for these actions and invokes the IP Filter rule-generation process to carry out the service's firewall policy.

This daemon is started by the `network/ipfilter` service either through the `start` or `refresh` method. Thus, the daemon inherits the environment variables and credentials from the method and runs as root with all zone privileges.

Firewall Static Configuration

A static definition describes a service's network resource configuration that is used to generate service-specific IPF rules. The per-service `firewall_context` property group contains a service's static definition, similar to the `inetd` property group in `inetd` managed services. This property group supports:

`firewall_context/name`
: For non-`inetd` services. The IANA name or RPC name, equivalent to the `inetd/name` property.

`firewall_context/isrpc`
: For non-`inetd` services. A boolean property where a `true` value indicates an RPC service, equivalent to the `inetd/isrpc` property. For RPC services, the value of `firewall_context/name` is not an IANA name but is either an RPC program number or name. See rpc(4).

Additionally, some services may require a mechanism to generate and supply their own IPF rules. An optional property `ipf_method`, provides a mechanism to allow such custom rule generation:

`firewall_context/ipf_method`
: A command. Normally a script that generates IPF rules for a service. The framework does not generate rules for services with this property definition. Rather, the framework expects these services to provide their own rules.

A service's `ipf_method` specifies a command that takes an additional argument, its own fault management resource identifier (FMRI), and generates the service's firewall rules and outputs those rules to stdout. To generate rules for a service with the `ipf_method` property, the framework execs the command specified in `ipf_method`, passing the service FMRI as the additional argument, and stores the rules for that service by redirecting the command output,

the rules, to the service's rule file. Because an `ipf_method` is `exec`'ed from the context of either the `network/ipfilter` `start` or `refresh` method process, it inherits the execution context and runs as root.

The service static configuration is delivered by the service developer and not intended to be modified by users. These properties are only modified upon installation of an updated service definition.

Firewall Policy Configuration

A per-service property group, `firewall_config`, stores the services' firewall policy configuration. Because `network/ipfilter:default` is responsible for two firewall policies, the Global Default and Global Override system-wide policies (as explained in `ipfilter(5)`), it has two property groups, `firewall_config_default` and `firewall_config_override`, to store the respective system-wide policies.

Below are the properties, their possible values, and corresponding semantics:

`policy`
The `policy` has the following modes:

`none` policy mode
No access restriction. For a global policy, this mode allows all incoming traffic. For a service policy, this mode allows all incoming traffic to its service.

`deny` policy mode
More restrictive than `none`. This mode allows incoming traffic from all sources except those specified in the `apply_to` property.

`allow` policy mode
Most restrictive mode. This mode blocks incoming traffic from all sources except those specified in the `apply_to` property.

`apply_to`
A multi-value property listing network entities to enforce the chosen policy mode. Entities listed in `apply_to` property will be denied if policy is `deny` and allowed if policy is `allow`. The syntax for possible values are:

`host:`	`host:`*IP*	`"host:192.168.84.14"`
`subnet:`	`network:`*IP/netmask*	`"network:129.168.1.5/24"`
`ippool:`	`pool:`*pool number*	`"pool:77"`
`interface:`	`if:`*interface_name*	`"if:e1000g0"`

`exceptions`
A multi-value property listing network entities to be excluded from the `apply_to` list. For example, when `deny` policy is applied to a subnet, exceptions can be made to some hosts in that subnet by specifying them in the `exceptions` property. This property has the same value syntax as `apply_to` property.

For individual network services only:

`firewall_config/policy`
A service's policy can also be set to `use_global`. Services with `use_global` policy mode inherits the Global Default firewall policy.

For the Global Default only:

`firewall_config_default/policy`
Global Default policy, `firewall_config` property group in `svc:/network/ipfilter:default`, can also be set to `custom`. Users can set `policy` to `custom` to use prepopulated IP Filter configuration, for example, an existing IP Filter configuration or custom configurations that cannot be provided by the framework. This Global Default-only policy mode allows users to supply a text file containing the complete set of IPF rules. When `custom` mode is selected, the specified set of IPF rules is *complete* and the framework will not generate IPF rules from configured firewall policies.

`firewall_config_default/custom_policy_file`
A file path to be used when Global Default policy is set to `custom`. The file contains a set of IPF rules that provide the desired IP Filter configuration. For example, users with existing IPF rules in `/etc/ipf/ipf.conf` can execute the following commands to use the existing rules:

1. Set custom policy:

```
# svccfg -s ipfilter:default setprop \
firewall_config_default/policy = astring: "custom"
```

2. Specify custom file:

```
# svccfg -s ipfilter:default setprop \
firewall_config_default/custom_policy_file = astring: \
"/etc/ipf/ipf.conf"
```

3. Refresh configuration:

```
# svcadm refresh ipfilter:default
```

`firewall_config_default/open_ports`
Non-service program requiring allowance of its incoming traffic can request that the firewall allow traffic to its communication ports. This multi-value property contains protocol and port(s) tuple in the form:

"`{tcp | udp}:{`*PORT* | *PORT*-*PORT*`}`"

Initially, the system-wide policies are set to `none` and network services' policies are set to `use_global`. Enabling `network/ipfilter` activates the firewall with an empty set of IP Filter rules, since system-wide policy is `none` and all services inherit that policy. To configure a more restrictive policy, use `svccfg(1M)` to modify network services and system-wide policies.

A user configures firewall policy by modifying the service's `firewall_config` property group. A new authorization, `solaris.smf.value.firewall.config`, is created to allow delegation

of the firewall administration privilege to users. Users with Service Operator privileges will need this new authorization to be able to configure firewall policy.

Firewall Availability

During boot, a firewall is configured for enabled services prior to the starting of those services. Thus, services are protected on boot. While the system is running, administrative actions such as service restarting, enabling, and refreshing may cause a brief service vulnerability during which the service runs while its firewall is being configured.

`svc.ipfd` monitors a service's start and stop events and configures or unconfigures a service's firewall at the same time that SMF is starting or stopping the service. Because the two operations are simultaneous, there is a possible window of exposure (less than a second) if the service is started before its firewall configuration completed. RPC services typically listen on ephemeral addresses, which are not known until the services are actually running. Thus RPC services are subjected to similar exposure since their firewalls are not configured until the services are running.

Developer Documentation

Services providing remote capabilities are encouraged to participate in the firewall framework to control network access to the service. While framework integration is not mandatory, remote access to services that are not integrated in the framework may not function correctly when a system-wide policy is configured.

Integrating a service into the framework is as straightforward as defining two additional property groups and their corresponding properties in the service manifest. IP Filter rules are generated when a user enables the service. In the non-trivial case of custom rule generation, where a shell script is required, there are existing scripts that can be used as examples.

The additional property groups, `firewall_config` and `firewall_context`, stores firewall policy configuration and provides static firewall definition, respectively. Below is a summary of new property groups and properties and their appropriate default values.

Firewall policy configuration:

`firewall_config`
: Access to the system is protected by a new authorization definition and a user-defined property type. The new authorization should be assigned to the property group `value_authorization` property in a way such as:

```
<propval name='value_authorization' type='astring'
value='solaris.smf.value.firewall.config' />
```

A third party should follow the service symbol namespace convention to generate a user-defined type. Sun-delivered services can use `com.sun,fw_configuration` as the property type.

See "Firewall Policy Configuration," above, for more information.

`firewall_config/policy`
: This property's initial value should be `use_global` since services, by default, inherit the Global Default firewall policy.

`firewall_config/apply_to`
An empty property, this property has no initial value.

`firewall_config/exceptions`
An empty property, this property has no initial value.

Firewall static definition:

`firewall_context`
A third party should follow service symbol namespace convention to generate a user-defined type, Sun delivered services can use `com.sun,fw_definition` as the property type.

See "Firewall Static Configuration," above, for more information.

`firewall_context/name`
Service with well-known, IANA defined port, which can be obtained by `getservbyname(3SOCKET)`. The service's IANA name is stored in this property. For RPC services, the RPC program number is stored in this property.

`firewall_context/isrpc`
For RPC services, this property should be created with its value set to `true`.

`firewall_context/ipf_method`
In general, the specified firewall policy is used to generate IP Filter rules to the service's communication port, derived from the `firewall_context/name` property. Services that do not have IANA-defined ports and are not RPC services will need to generate their own IP Filter rules. Services that generate their own rules may choose not to have `firewall_context/name` and `firewall_context/isrpc` properties. See the following services:

```
svc:/network/ftp:default
svc:/network/nfs/server:default
svc:/network/ntp:default
```

...and others with the `ipf_method` for guidance.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os, network/ipfilter
Interface Stability	Committed

See Also

`svcprop(1)`, `svcs(1)`, `ipf(1M)`, `svcadm(1M)`, `svccfg(1M)`, `getservbyname(3SOCKET)`, `rpc(4)`, `attributes(5)`, `ipfilter(5)`, `smf(5)`

Name svc.startd – Service Management Facility master restarter

Synopsis

```
/lib/svc/bin/svc.startd

svc:/system/svc/restarter:default
```

Description `svc.startd` is the master restarter daemon for Service Management Facility (SMF) and the default restarter for all services. `svc.startd` starts, stops, and restarts services based on administrative requests, system failures, or application failures.

`svc.startd` maintains service state, as well as being responsible for managing faults in accordance with the dependencies of each service.

`svc.startd` is invoked automatically during system startup. It is restarted if any failures occur. `svc.startd` should never be invoked directly.

See `smf_restarter(5)` for information on configuration and behavior common to all restarters.

`svcs(1)` reports status for all services managed by the Service Configuration Facility. `svcadm(1M)` allows manipulation of service instances with respect to the service's restarter.

Environment Variables Environment variables with the "SMF_" prefix are reserved and may be overwritten.

`svc.startd` supplies the "SMF_" environment variables specified in `smf_method(5)` to the method. PATH is set to "`/usr/sbin:/usr/bin`" by default. By default, all other environment variables supplied to `svc.startd` are those inherited from `init(1M)`.

Duplicate entries are reduced to a single entry. The value used is undefined. Environment entries that are not prefixed with "<*name*>=" are ignored.

Restarter Options `svc.startd` is not configured by command line options. Instead, configuration is read from the service configuration repository. You can use `svccfg(1M)` to set all options and properties.

The following configuration variables in the `options` property group are available to developers and administrators:

`boot_messages`
: An *astring* (as defined in `scf_value_is_type`; see `scf_value_create(3SCF)`) that describes the default level of messages to print to the console during boot. The supported message options include `quiet` and `verbose`. The `quiet` option prints minimal messages to console during boot. The `verbose` option prints a single message per service started to indicate success or failure. You can use the `boot -m` option to override the `boot_messages` setting at boot time. See `kernel(1M)`.

`logging`
: Control the level of global service logging for `svc.startd`. An *astring* (as defined in `scf_value_is_type`; see `scf_value_create(3SCF)`) that describes the default level of messages to log to `syslog` (see `syslog(3C)` and `svc.startd`'s global logfile,

/var/svc/log/svc.startd.log. The supported message options include quiet, verbose, and debug. The quiet option sends error messages requiring administrative intervention to the console, syslog and svc.startd's global logfile. The verbose option sends error messages requiring administrative intervention to the console, syslog and svc.startd's global logfile, and information about errors which do not require administrative intervention to svc.startd's global logfile. A single message per service started is also sent to the console. The debug option sends svc.startd debug messages to svc.startd's global logfile, error messages requiring administrative intervention to the console, syslog and svc.startd's global logfile, and a single message per service started to the console.

`milestone`
An FMRI which determines the milestone used as the default boot level. Acceptable options include only the major milestones:

```
svc:/milestone/single-user:default
svc:/milestone/multi-user:default
svc:/milestone/multi-user-server:default
```

or the special values all or none. all represents an idealized milestone that depends on every service. none is a special milestone where no services are running apart from the master svc:/system/svc/restarter:default. By default, svc.startd uses all, a synthetic milestone that depends on every service. If this property is specified, it overrides any initdefault setting in inittab(4).

`system/reconfigure`
Indicates that a reconfiguration reboot has been requested. Services with actions that must key off of a reconfiguration reboot may check that this property exists and is set to 1 to confirm a reconfiguration boot has been requested.

This property is managed by svc.startd and should not be modified by the administrator.

Configuration errors, such as disabling svc.startd are logged by syslog, but ignored.

SERVICE STATES

Services managed by svc.startd can appear in any of the states described in smf(5). The state definitions are unmodified by this restarter.

SERVICE REPORTING

In addition to any logging done by the managed service, svc.startd provides a common set of service reporting and logging mechanisms.

Reporting properties svc.startd updates a common set of properties on all services it manages. These properties are a common interface that can be used to take action based on service instance health. The svcs(1) command can be used to easily display these properties.

`restarter/state`
`restarter/next_state`
The current and next (if currently in transition) state for an instance.

`restarter/auxiliary_state`
A caption detailing additional information about the current instance state. The auxiliary state available for services managed by `svc.startd` is:

maintenance

```
fault_threshold_reached
stop_method_failed
administrative_request
custom
```

disabled

```
custom
```

`restarter/auxiliary_custom_state`
When `restarter/auxiliary_custom_state` is set to `custom`, a method-provided caption detailing additional information about the current instance state.

`restarter/auxiliary_reason`
When `restarter/auxiliary_custom_state` is set to `custom`, a method-provided string detailing additional information about the current instance state.

`restarter/auxiliary_textdomain`
When `restarter/auxiliary_custom_state` is set to `custom`, a method-provided textdomain in which `restarter/auxiliary_reason` may be localized.

`restarter/state_timestamp`
The time when the current state was reached.

`restarter/contract`
The primary process contract ID, if any, that under which the service instance is executing.

Logs

By default, `svc.startd` provides logging of significant restarter actions for the service as well as method standard output and standard error file descriptors to `/var/svc/log/`*service*`:`*instance*`.log`. The level of logging to system global locations like `/var/svc/log/svc.startd.log` and `syslog` is controlled by the `options/logging` property.

SERVICE DEFINITION

When developing or configuring a service managed by `svc.startd`, a common set of properties are used to affect the interaction between the service instance and the restarter.

Methods

The general form of methods for the fork/exec model provided by `svc.startd` are presented in `smf_method(5)`. The following methods are supported as required or optional by services managed by `svc.startd`.

`refresh` Reload any appropriate configuration parameters from the repository or `config` file, without interrupting service. This is often implemented using `SIGHUP` for

system daemons. If the service is unable to recognize configuration changes without a restart, no refresh method is provided.

This method is optional.

`start` Start the service. Return success only after the application is available to consumers. Fail if a conflicting instance is already running, or if the service is unable to start.

This method is required.

`stop` Stop the service. In some cases, the stop method can be invoked when some or all of the service has already been stopped. Only return an error if the service is not entirely stopped on method return.

This method is required.

If the service does not need to take any action in a required method, it must specify the `:true` token for that method.

`svc.startd` honors any method context specified for the service or any specific method. The method expansion tokens described in `smf_method(5)` are available for use in all methods invoked by `svc.startd`.

Properties

An overview of the general properties is available in `smf(5)`. The specific way in which these general properties interacts with `svc.startd` follows:

`general/enabled`
If enabled is set to true, the restarter attempts to start the service once all its dependencies are satisfied. If set to false, the service remains in the disabled state, not running.

`general/restarter`
If this FMRI property is empty or set to `svc:/system/svc/restarter:default`, the service is managed by `svc.startd`. Otherwise, the restarter specified is responsible (once it is available) for managing the service.

`general/single_instance`
If `single_instance` is set to true, `svc.startd` only allows one instance of this service to transition to online or degraded at any time.

Additionally, `svc.startd` managed services can define the optional properties listed below in the `startd` property group.

`startd/duration`
The `duration` property defines the service's model. It can be set to `transient`, `child` also known as "`wait`" model services, or `contract` (the default).

`startd/ignore_error`
The `ignore_error` property, if set, specifies a comma-separated list of ignored events. Legitimate string values in that list are `core` and `signal`. The default is to restart on all errors.

`startd/need_session`
The `need_session` property, if set to true, indicates that the instance should be launched in its own session. The default is not to do so.

`startd/utmpx_prefix`
The `utmpx_prefix` string property defines that the instance requires a valid `utmpx` entry prior to start method execution. The default is not to create a `utmpx` entry.

SERVICE METHOD SPECIAL REQUESTS

A service may use `smf_method_exit()` to request special consideration for the state transition in progress. If such requests are made by means of `exit()`, without using `smf_method_exit()`, they will be treated as described in `SERVICE FAILURES` below.

When a start or refresh method requests `$SMF_EXIT_TEMP_DISABLE`, `svc.startd` will temporarily disable the service and place it in the disabled state without running its stop method. If appropriate, the start or refresh method may explicitly invoke the stop method to gracefully shut down any processes it may have started prior to returning `$SMF_EXIT_TEMP_DISABLE`.

When a contract service requests `$SMF_EXIT_TEMP_TRANSIENT` from a start or refresh method, `svc.startd` will complete the state transition currently in progress, and will treat the service as if it were transient.

When a stop method requests `$SMF_EXIT_TEMP_DISABLE` or `$SMF_EXIT_TEMP_TRANSIENT`, `svc.startd` will treat it as if it had returned `$SMF_EXIT_ERR_OK`.

When a non-contract service requests `$SMF_EXIT_TEMP_TRANSIENT` from a start or refresh method, `svc.startd` will treat it as if it had returned `$SMF_EXIT_ERR_OK`.

SERVICE FAILURE

Except under the conditions described in `SERVICE METHOD SPECIAL REQUESTS`, `svc.startd` assumes that a method has failed if it returns a non-zero exit code or if fails to complete before the timeout specified expires. If `$SMF_EXIT_ERR_CONFIG` or `$SMF_EXIT_ERR_FATAL` is returned, `svc.startd` immediately places the service in the maintenance state. For all other failures, `svc.startd` places the service in the offline state. If a service is offline and its dependencies are satisfied, `svc.startd` tries again to start the service (see `smf(5)`).

If a contract or transient service does not return from its start method before its defined timeout elapses, `svc.startd` sends a `SIGKILL` to the method, and returns the service to the offline state.

If five failures happen in a row, or if the service is restarting due to an error more than once every ten minutes, `svc.startd` places the service in the maintenance state.

The conditions of service failure are defined by a combination of the service model (defined by the `startd/duration` property) and the value of the `startd/ignore_error` property.

A contract model service fails if any of the following conditions occur:

- all processes in the service exit
- any processes in the service produce a core dump
- a process outside the service sends a service process a fatal signal (for example, an administrator terminates a service process with the `pkill` command)

The last two conditions may be ignored by the service by specifying core and/or signal in `startd/ignore_error`.

Defining a service as transient means that `svc.startd` does not track processes for that service. Thus, the potential faults described for contract model services are not considered failures for transient services. A transient service only enters the maintenance state if one of the method failure conditions occurs.

"`Wait`" model services are restarted whenever the child process associated with the service exits. A child process that exits is not considered an error for "`wait`" model services, and repeated failures do not lead to a transition to maintenance state.

LEGACY SERVICES

`svc.startd` continues to provide support for services invoked during the startup run level transitions. Each `/etc/rc?.d` directory is processed after all managed services which constitute the equivalent run level milestone have transitioned to the online state. Standard `init` scripts placed in the `/etc/rc?.d` directories are run in the order of their sequence numbers.

The milestone to run-level mapping is:

`milestone/single-user`	Single-user (`S`)
`milestone/multi-user`	Multi-user (`2`)
`milestone/multi-user-server`	Multi-user with network services (`3`)

Additionally, `svc.startd` gives these legacy services visibility in SMF by inserting an instance per script into the repository. These legacy instances are visible using standard SMF interfaces such as `svcs(1)`, always appear in the `LEGACY-RUN` state, cannot be modified, and can not be specified as dependencies of other services. The initial start time of the legacy service is captured as a convenience for the administrator.

Files

`/var/svc/log`	Directory where `svc.startd` stores log files.
`/etc/svc/volatile`	Directory where `svc.startd` stores log files in early stages of boot, before `/var` is mounted read-write.

Example

EXAMPLE 1 Turning on Verbose Logging

To turn on verbose logging, type the following:

EXAMPLE 1 Turning on Verbose Logging *(Continued)*

```
# /usr/sbin/svccfg -s system/svc/restarter:default
svc:/system/svc/restarter:default> addpg options application
svc:/system/svc/restarter:default> setprop options/logging = \
astring: verbose
svc:/system/svc/restarter:default> exit
```

This request will take effect on the next restart of `svc.startd`.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also

`svcs(1)`, `svcprop(1)`, `kernel(1M)`, `init(1M)`, `svcadm(1M)`, `svccfg(1M)`, `svc.configd(1M)`, `setsid(2)`, `syslog(3C)`, `libscf(3LIB)`, `scf_value_create(3SCF)`, `smf_method_exit(3SCF)`, `contract(4)`, `init.d(4)`, `process(4)`, `inittab(4)`, `attributes(5)`, `smf(5)`, `smf_method(5)`

Name swap – swap administrative interface

Synopsis

```
/usr/sbin/swap -a swapname [swaplow] [swaplen]

/usr/sbin/swap -d swapname [swaplow]

/usr/sbin/swap -l [-h | -k]

/usr/sbin/swap -s [-h]
```

Description The `swap` utility provides a method of adding, deleting, and monitoring the system swap areas used by the memory manager.

Options The following options are supported:

`-a` *swapname* [*swaplow*] [*swaplen*]
Add the specified swap area. This option can only be used by an administrator who is assigned the File System Management rights profile or by root. *swapname* is the name of the swap area or regular file. For example, on system running a UFS root file system, specify a slice, such as `/dev/dsk/c0t0d0s1`, or a regular file for a swap area. On a system running a ZFS file system, specify a ZFS volume, such as `/dev/zvol/dsk/rpool/swap`, for a swap area. Using a regular file for swap is not supported on a ZFS file system. In addition, you cannot use the same ZFS volume for both the swap area and a dump device when the system is running a ZFS root file system.

swaplow is the offset in 512-byte blocks into the file where the swap area should begin. *swaplen* is the desired length of the swap area in 512-byte blocks. The value of *swaplen* can not be less than `16`. For example, if *n* blocks are specified, then (*n*-1) blocks would be the actual swap length. *swaplen* must be at least one page in length. The size of a page of memory can be determined by using the `pagesize` command. See `pagesize(1)`. Since the first page of a swap file is automatically skipped, and a swap file needs to be at least one page in length, the minimum size should be a multiple of 2 pagesize bytes. The size of a page of memory is machine-dependent.

swaplow + *swaplen* must be less than or equal to the size of the swap file. If *swaplen* is not specified, an area will be added starting at *swaplow* and extending to the end of the designated file. If neither *swaplow* nor *swaplen* are specified, the whole file will be used except for the first page. Swap areas are normally added automatically during system startup by the `/usr/sbin/swapadd` script. This script adds all swap areas which have been specified in the `/etc/vfstab` file; for the syntax of these specifications, see `vfstab(4)`.

You can encrypt a ZFS volume used as a swap device by specifying the `encrypted` option in `vfstab(4)` and specifying the `encryption` property for the ZFS volume. See `zfs(1M)`.

To use an NFS or local file system *swapname*, you should first create a file using `mkfile(1M)`. A local file system swap file can now be added to the running system by just running the `swap -a` command. For NFS mounted swap files, the server needs to export the file. Do this by performing the following steps:

1. Add the following line to `/etc/dfs/dfstab`:

```
share -F nfs -o \
rw=clientname,root=clientname path-to-swap-file
```

2. Run shareall(1M).
3. Have the client add the following line to /etc/vfstab:

```
server:path-to-swap-file - local-path-to-swap-file nfs \
    - - - local-path-to-swap-file - - swap - - -
```

4. Have the client run mount:

```
# mount local-path-to-swap-file
```

5. The client can then run swap -a to add the swap space:

```
# swap -a local-path-to-swap-file
```

-d *swapname*
Delete the specified swap area. This option can only be used by the super-user. *swapname* is the name of the swap file: for example, /dev/dsk/c0t0d0s1 or a regular file. *swaplow* is the offset in 512-byte blocks into the swap area to be deleted. If *swaplow* is not specified, the area will be deleted starting at the second page. When the command completes, swap blocks can no longer be allocated from this area and all swap blocks previously in use in this swap area have been moved to other swap areas.

-h
All sizes are scaled to a human readable format. Scaling is done by repetitively dividing by 1024.

-k
Write the files sizes in units of 1024 bytes.

-l
List the status of all the swap areas. The output has five columns:

path
The path name for the swap area.

dev
The major/minor device number in decimal if it is a block special device; zeroes otherwise.

swaplo
The *swaplow* value for the area in 512-byte blocks.

blocks
The *swaplen* value for the area in 512-byte blocks.

free
The number of 512-byte blocks in this area that are not currently allocated.

The list does not include swap space in the form of physical memory because this space is not associated with a particular swap area.

If `swap -l` is run while *swapname* is in the process of being deleted (by `swap -d`), the string INDEL will appear in a sixth column of the swap stats.

`-s`

Print summary information about total swap space usage and availability:

`allocated`
The total amount of swap space in bytes currently allocated for use as backing store.

`reserved`
The total amount of swap space in bytes not currently allocated, but claimed by memory mappings for possible future use.

`used`
The total amount of swap space in bytes that is either allocated or reserved.

`available`
The total swap space in bytes that is currently available for future reservation and allocation.

These numbers include swap space from all configured swap areas as listed by the `-l` option, as well swap space in the form of physical memory.

Usage A block device larger than 2 Gbytes can be fully utilized for swap up to $2^{63} - 1$ bytes.

Environment Variables See `environ(5)` for descriptions of the following environment variables that affect the execution of `swap`: `LC_CTYPE` and `LC_MESSAGE`.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also `pagesize(1)`, `mkfile(1M)`, `shareall(1M)`, `zfs(1M)`, `getpagesize(3C)`, `vfstab(4)`, `attributes(5)`, `largefile(5)`

Notes For information about setting up a swap area with ZFS, see the *ZFS Administration Guide*.

Warnings No check is done to determine if a swap area being added overlaps with an existing file system.

Name sxadm – manage security extensions configuration

Synopsis

```
/usr/sbin/sxadm enable [-c conf=value[,conf=value,...]
     extension[ extension...]

/usr/sbin/sxadm disable extension[ extension...]

/usr/sbin/sxadm delcust extension[ extension...]

/usr/sbin/sxadm exec [-s extension=value]... command

/usr/sbin/sxadm info [-p] [extension]

/usr/sbin/sxadm help [subcommand]
```

Description The `sxadm` command controls and configures Solaris security extensions both at the system level (global zone, non-global zone) and at the process level (`/usr/sbin/sxadm exec`).

The `enable` and `disable` subcommands enable and disable a given security extension system-wide. The `-c` option passes extension-specific configuration information to enable. The `delcust` subcommand resets an extension to the out-of-the-box default configuration.

The `info` subcommand reports the status of security extensions for the current zone. The `-p` option produces easily parseable output for external consumers.

The `exec` subcommand allows you to control the status of a given security extension at the process level. The specified command is executed with the security extension configured as expressed by any `-s`*extension*=*value* entry following the `exec` subcommand. Security extensions that are configured on the command line are inherited by child processes.

Security Extensions

Security extensions for a process are determined during `exec(2)` and become effective for a process upon exit from the `exec(2)` system call. Extensions persist for the lifetime of the process until the process exits or calls `exec(2)` again.

ASLR - Address Space Layout Randomization

ASLR activates the randomization of key areas of the process such as stack, `brk`-based heap, memory mappings, and so forth.

By default, the global zone and all non-global zones boot with ASLR enabled only for tagged binaries. Tagged binaries are built using the link-editor's `-z aslr` option. See the Address Space Layout Randomization (ASLR) section in the *Developer's Guide to Oracle Solaris 11 Security* for more details. Many core Solaris binaries are tagged with ASLR enabled. The `sxadm enable`, `disable`, and `restore` subcommands can be used to configure ASLR system-wide. ASLR configuration values for `sxadm enable` are:

`model=all`	Enable ASLR for all processes.
`model=tagged-files`	Enable ASLR for tagged binaries only.
`model=default`	Follow system default. Currently: `tagged-files`

ASLR configuration values for the `sxadm exec` command are:

`aslr=enable`	Enable ASLR for the process.

`aslr=disable` Disable ASLR for the process.

ASLR is not supported for Solaris 10 Containers.

Sub-commands The `sxadm` command has the following subcommands:

`sxadm enable` [`-c` *conf*=*value*[,*conf*=*value*,...] *extension*[,*extension*]
Enable the specified extension for the current zone. The `-c` option allows `sxadm` to pass configuration information for the specific extension.

Multiple extensions and multiple configuration values can both be specified on the command line, although if the configuration value does not apply to all extensions, the command will fail. Most common uses of this command are thus:

```
% sxadm enable extension1 extension2
```

Also:

```
% sxadm enable -c prop=value,prop2=value2 extension
```

See the Examples section for more examples.

`sxadm disable` *extension*[,*extension*]
Disable the specified extension for the current zone.

`sxadm delcust` *extension*[,*extension*]
Restore the extension to the default (out-of-the-box) configuration.

`sxadm info` [`-p`] [*extension*]
Report information on the status of all security extensions for the current zone. If `-p` is specified, the output is displayed in an easily parseable format. Specifying an *extension* on the command line filters for the specific extension.

Machine parseable output is a list of colon-separated fields:

extension_name:*status*[.*extra*]:*configuration*[.*extra*]

where:

extension_name	The name of the extension.
status	The current status for the extension (enabled or disabled).
extra	Represents (significant) extra information that the extension wishes to report. As an example, in the ASLR case, if ASLR is enabled, *extra* can either be `tag` or `all` depending on the model.
configuration	The stored configuration for the extension (enabled, disabled, or system default)

The characters colon (`:`), null sign (`\0`), and newline (`\n`) are not permitted for any of the components, *extension_name*, *status*, *extra*, and *configuration*.

`sxadm exec -s` [*extension=value*]... *command*
Execute the specified command with a specific configuration for security extensions. For each security extension not explicitly configured on the command line, the system configuration is used. Child processes eventually spawned by *command* inherit the same security extension configuration that was specified on the command line. `set-uids` and privileged binaries do not inherit any configuration. Multiple configurations can be expressed on a single command line using multiple `-s` options. If the same extension is configured more than once, the last occurrence takes precedence. For example:

```
% sxadm exec -s aslr=disable -s aslr=enable /usr/bin/pmap
```

...executes `/usr/bin/pmap` with `aslr` enabled.

The `sxadm exec` subcommand is designed to accommodate the common case in which a debugger is applied to a single process started directly by the debugger. It may not be sufficient for more complex scenarios. In such cases, it may be necessary to use `sxadm` to change the system or zone level security extension defaults, or to apply per-object tagging using the `ld(1)` utility, in order to facilitate debugging.

`sxadm help` [*subcommand*]
Display usage information about `sxadm` or more detailed information for each subcommand.

Examples

EXAMPLE 1 Executing `pmap` Binary

The `sxadm` command below executes the `pmap` binary with ASLR disabled at runtime.

```
bash$ pmap self
# memory addresses are randomized
101731: pmap self
101731: pmap self
0000000000400000         28K r-x--  /usr/bin/pmap
0000000000417000          4K rw---  /usr/bin/pmap
0000000000418000          8K rw---  /usr/bin/pmap
000003B0E8DF8000         36K rw---    [ heap ]
[...]
FFFFF843B8098000        344K r-x--  /lib/amd64/ld.so.1
FFFFF843B80FE000         12K rwx--  /lib/amd64/ld.so.1
FFFFF843B8101000          8K rwx--  /lib/amd64/ld.so.1
FFFFFBF4A14E0000         12K rw---    [ stack ]
      total         2592K

bash$ sxadm exec aslr=disable /usr/bin/pmap self
101733: /usr/bin/pmap self
101733: /usr/bin/pmap self
0000000000400000         28K r-x--  /usr/bin/pmap
0000000000417000          4K rw---  /usr/bin/pmap
0000000000418000         40K rw---    [ heap ]
[...]
```

EXAMPLE 1 Executing pmap Binary *(Continued)*

```
FFFFFD7FFF394000        344K r-x--  /lib/amd64/ld.so.1
FFFFFD7FFF3FA000         12K rwx--  /lib/amd64/ld.so.1
FFFFFD7FFF3FD000          8K rwx--  /lib/amd64/ld.so.1
FFFFFD7FFFDFD000         12K rw---    [ stack ]
      total         2588K
```

EXAMPLE 2 Displaying Information about the Security Extensions Configuration

The following `sxadm info` commands display information about the security extensions configuration.

```
bash$ sxadm info -p
aslr:enabled.tagged-files:system default.default
bash$ sxadm info
EXTENSION        STATUS                  CONFIGURATION
aslr             enable (tagged-files)   system default (default)
bash$ sxadm enable -c model=all aslr
bash$ sxadm info
EXTENSION        STATUS                  CONFIGURATION
aslr             enable (all)            enable (all)
bash$ sxadm info -p
aslr:enabled.all:enabled.all
```

EXAMPLE 3 Reset to Default Configuration

The following command `sxadm delcust` command restores the extension to the default, out-of-the-box configuration.

```
bash$ sxadm info
EXTENSION        STATUS                  CONFIGURATION
aslr             enable (all)            enable (all)
bash$ sxadm delcust aslr
bash$ sxadm info
EXTENSION        STATUS                  CONFIGURATION
aslr             enable (tagged-files)   system default (default)
```

EXAMPLE 4 Running a Debugging Session

The following command sequence illustrates a debugging session being conducted with ASLR disabled.

```
bash$ sxadm exec -s aslr=disable /bin/bash
bash$
# Because all processes (except privileged ones) inherit the (disabled)
# aslr configuration mdb, truss & co will have repeatable results.

bash$ truss -t mmap /bin/true
mmap(0x00000000, 32, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANON, -1, 0)
```

EXAMPLE 4 Running a Debugging Session *(Continued)*

```
= 0xFE5B0000
mmap(0x00000000, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANON, -1, 0)
= 0xFE5A0000
mmap(0x00000000, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANON, -1, 0)
= 0xFE590000
[...]
bash$ truss -t mmap /bin/true
mmap(0x00000000, 32, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANON, -1, 0)
= 0xFE5B0000
mmap(0x00000000, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANON, -1, 0)
= 0xFE5A0000
mmap(0x00000000, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANON, -1, 0)
= 0xFE590000
[...]
bash$ truss -t mmap /bin/true
mmap(0x00000000, 32, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANON, -1, 0)
= 0xFE5B0000
mmap(0x00000000, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANON, -1, 0)
= 0xFE5A0000
mmap(0x00000000, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANON, -1, 0)
= 0xFE590000
```

Exit Status

`0`
The command completed successfully.

`1`
The command exited due to an error.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os
Interface Stability	Committed

See Also

`ld(1)`, `exec(2)`, `attributes(5)`

Oracle Solaris 11.1 Administration: Security Services

Address Space Layout Randomization (PaxTeam). Under `http://pax.grsecurity.net/`

Address Space Layout Randomization in Windows Vista. Under `http://blogs.msdn.com/b/michael_howard/`

Address space randomization in 2.6. Under `http://lwn.net/`

Official mention on the web site of Library Randomization for Mac OS X Snow Leopard (Mac OS X Lion has full randomization). Under `http://www.apple.com/macosx/security`

Name sync – update the super block

Synopsis

```
sync
```

Description `sync` executes the `sync` system primitive. If the system is to be stopped, `sync` must be called to ensure file system integrity. It will flush all previously unwritten system buffers out to disk, thus assuring that all file modifications up to that point will be saved. See `sync(2)` for details.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also `sync(2)`, `attributes(5)`

Name syncinit – set serial line interface operating parameters

Synopsis

```
/usr/sbin/syncinit device
     [ [baud_rate] | [keyword=value,]... | [single-word option]]
```

Description The `syncinit` utility allows the user to modify some of the hardware operating modes common to synchronous serial lines. This can be useful in troubleshooting a link, or necessary to the operation of a communications package.

If run without options, `syncinit` reports the options as presently set on the port. If options are specified, the new settings are reported after they have been made.

Options Options to `syncinit` normally take the form of a keyword, followed by an equal sign and a value. The exception is that a baud rate may be specified as a decimal integer by itself. Keywords must begin with the value shown in the options table, but may contain additional letters up to the equal sign. For example, `loop=` and `loopback=` are equivalent.

The following options are supported:

Keyword	Value	Effect
`loop`	yes	Set the port to operate in `internal loopback` mode. The receiver is electrically disconnected from the DCE receive data input and tied to the outgoing transmit data line. Transmit data is available to the DCE. The Digital Phase-Locked Loop (DPLL) may not be used as a clock source in this mode. If no other clocking options have been specified, perform the equivalent of `txc=baud` and `rxc=baud`.
	no	Disable internal loopback mode. If no other clocking options have been specified, perform the equivalent of `txc=txc` and `rxc=rxc`.
`echo`	yes	Set the port to operate in `auto-echo` mode. The transmit data output is electrically disconnected from the transmitter and tied to the receive data input. Incoming receive data is still visible. Use of this mode in combination with local loopback mode has no value, and should be rejected by the device driver. The `auto-echo` mode is useful to make a system become the endpoint of a remote loopback test.
	no	Disable `auto-echo` mode.
`nrzi`	yes	Set the port to operate with `NRZI` data encoding.
	no	Set the port to operate with `NRZ` data encoding.
`txc`	`txc`	Transmit clock source will be the `TxC` signal (pin 15).
	`rxc`	Transmit clock source will be the `RxC` signal (pin 17).
	`baud`	Transmit clock source will be the internal `baud rate generator`.
	`pll`	Transmit clock source will be the output of the `DPLL` circuit.

`rxc`	`rxc`	Receive clock source will be the `RxC` signal (pin 17).
	`txc`	Receive clock source will be the `TxC` signal (pin 15).
	`baud`	Receive clock source will be the internal `baud rate generator`.
	`pll`	Receive clock source will be the output of the `DPLL` circuit.
`speed`	*integer*	Set the baud rate to *integer* bits per second.

There are also several single-word options that set one or more parameters at a time:

Keyword	Equivalent to Options:
`external`	`txc=txc rxc=rxc loop=no`
`sender`	`txc=baud rxc=rxc loop=no`
`internal`	`txc=pll rxc=pll loop=no`
`stop`	`speed=0`

Examples

EXAMPLE 1 Using `syncinit`

The following command sets the first CPU port to loop internally, using internal clocking and operating at 38400 baud:

```
example# syncinit zsh0 38400 loop=yes
device: /dev/zsh  ppa: 0
speed=38400, loopback=yes, echo=no, nrzi=no, txc=baud, rxc=baud
```

The following command sets the same port's clocking, local loopback and baud rate settings to their default values:

```
example# syncinit zsh0 stop loop=no
device: /dev/zsh  ppa: 0
speed=0, loopback=no, echo=no, nrzi=no, txc=txc, rxc=rxc
```

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also

`syncloop(1M)`, `syncstat(1M)`, `Intro(2)`, `ioctl(2)`, `attributes(5)`, `zsh(7D)`

Diagnostics

device `missing minor device number`	The name *device* does not end in a decimal number that can be used as a minor device number.

`bad speed:` *arg*	The string *arg* that accompanied the `speed=` option could not be interpreted as a decimal integer.
`Bad arg:` *arg*	The string *arg* did not make sense as an option.
`ioctl failure code =` *errno*	An `ioctl(2)` system called failed. The meaning of the value of *errno* may be found in `Intro(2)`.

Warnings Do not use `syncinit` on an active serial link, unless needed to resolve an error condition. Do not use this command casually or without being aware of the consequences.

Name syncloop – synchronous serial loopback test program

Synopsis `/usr/sbin/syncloop [-cdlstv]` *device*

Description The `syncloop` command performs several loopback tests that are useful in exercising the various components of a serial communications link.

Before running a test, `syncloop` opens the designated port and configures it according to command line options and the specified test type. It announces the names of the devices being used to control the hardware channel, the channel number (ppa) corresponding to the *device* argument, and the parameters it has set for that channel. It then runs the loopback test in three phases.

The first phase is to listen on the port for any activity. If no activity is seen for at least four seconds, `syncloop` proceeds to the next phase. Otherwise, the user is informed that the line is active and that the test cannot proceed, and the program exits.

In the second phase, called the "first-packet" phase, `syncloop` attempts to send and receive one packet. The program will wait for up to four seconds for the returned packet. If no packets are seen after five attempts, the test fails with an excoriating message. If a packet is returned, the result is compared with the original. If the length and content do not match exactly, the test fails.

The final phase, known as the "multiple-packet" phase, attempts to send many packets through the loop. Because the program has verified the integrity of the link in the first-packet phase, the test will not fail after a particular number of timeouts. If a packet is not seen after four seconds, a message is displayed. Otherwise, a count of the number of packets received is updated on the display once per second. If it becomes obvious that the test is not receiving packets during this phase, the user may wish to stop the program manually. The number and size of the packets sent during this phase is determined by default values, or by command line options. Each returned packet is compared with its original for length and content. If a mismatch is detected, the test fails. The test completes when the required number of packets have been sent, regardless of errors.

After the multiple-packet phase has completed, the program displays a summary of the hardware event statistics for the channel that was tested. The display takes the following form:

```
CRC errors   Aborts   Overruns   Underruns   In<-Drops-> Out
        0         0          0           0   0             0
```

This is followed by an estimated line speed, which is an approximation of the bit rate of the line, based on the number of bytes sent and the actual time that it took to send them.

Options The options for `syncloop` are described in the following table:

Option	Parameter	Default	Description
`-c`	*packet_count*	100	Specifies the number of packets to be sent in the multiple-packet phase.
`-d`	*hex_data_byte*	*random*	Specifies that each packet will be filled with bytes with the value of *hex_data_byte*.
`-l`	*packet_length*	100	Specifies the length of each packet in bytes.
`-s`	*line_speed*	9600	Bit rate in bits per second.
`-v`			Sets verbose mode. If data errors occur, the expected and received data is displayed.
`-t`	*test_type*	*none*	A number, from 1 to 4, that specifies which test to perform. The values for *test_type* are as follows: 1: Internal loopback test. Port loopback is on. Transmit and receive clock sources are internal (baud rate generator). 2: External loopback test. Port loopback is off. Transmit and receive clock sources are internal. Requires a loopback plug suitable to the port under test. 3: External loopback test. Port loopback is off. Transmit and receive clock sources are external (modem). Requires that one of the local modem, the remote modem, or the remote system be set in a loopback configuration. 4: Test using predefined parameters. User defines hardware configuration and may select port parameters using the `syncinit(1M)` command.

All numeric options except `-d` are entered as decimal numbers (for example, `-s 19200`). If you do not provide the `-t` *test_type* option, `syncloop` prompts for it.

Examples

EXAMPLE 1 A sample display of using the `syncloop` command.

In the following command `syncloop` uses a packet length of 512 bytes over the first CPU port:

```
example# syncloop -l 512 zsh0
```

In response to the above command, `syncloop` prompts you for the test option you want.

The following command performs an internal loopback test on the first CPU port, using 5000 packets and a bit rate of 56Kbps:

```
example# syncloop -t 1 -s 56000 -c 5000 zsh0
```

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also syncinit(1M), syncstat(1M), attributes(5), zsh(7D)

Diagnostics

device `missing minor device number`	The name *device* does not end in a decimal number that can be used as a minor device number.
`invalid packet length:` *nnn*	The packet length was specified to be less than zero or greater than 4096.
`poll: nothing to read`	
`poll: nothing to read or write.`	The poll(2) system call indicates that there is no input pending and/or that output would be blocked if attempted.
`len` *xxx* `should be` *yyy*	The packet that was sent had a length of *yyy*, but was received with a length of *xxx*.
nnn `packets lost in outbound queueing`	
nnn `packets lost in inbound queueing`	A discrepancy has been found between the number of packets sent by `syncloop` and the number of packets the driver counted as transmitted, or between the number counted as received and the number read by the program.

Warnings To allow its tests to run properly, as well as prevent disturbance of normal operations, `syncloop` should only be run on a port that is not being used for any other purpose at that time.

Name syncstat – report driver statistics from a synchronous serial link

Synopsis

```
/usr/sbin/syncstat [-c] device [interval]
```

Description The `syncstat` command reports the event statistics maintained by a synchronous serial device driver. The report may be a single snapshot of the accumulated totals, or a series of samples showing incremental changes. Prior to these it prints the device name being used to query a particular device driver, along with a number indicating the channel number (ppa) under control of that driver.

Event statistics are maintained by a driver for each physical channel that it supports. They are initialized to zero at the time the driver module is loaded into the system, which may be either at boot time or when one of the driver's entry points is first called.

The *device* argument is the name of the serial device as it appears in the `/dev` directory. For example, `zsh0` specifies the first on-board serial device.

The following is a breakdown of `syncstat` output:

`speed`	The line speed the device has been set to operate at. It is the user's responsibility to make this value correspond to the modem clocking speed when clocking is provided by the modem.
`ipkts`	The total number of input packets.
`opkts`	The total number of output packets.
`undrun`	The number of transmitter underrun errors.
`ovrrun`	The number of receiver overrun errors.
`abort`	The number of aborted received frames.
`crc`	The number of received frames with CRC errors.
`isize`	The average size (in bytes) of input packets.
`osize`	The average size (in bytes) of output packets.

Options

`-c` Clear the accumulated statistics for the device specified. This may be useful when it is not desirable to unload a particular driver, or when the driver is not capable of being unloaded.

interval `syncstat` samples the statistics every *interval* seconds and reports incremental changes. The output reports line utilization for input and output in place of average packet sizes. These are the relationships between bytes transferred and the baud rate, expressed as percentages. The loop repeats indefinitely, with a column heading printed every twenty lines for convenience.

Examples

EXAMPLE 1 Sample output from the `syncstat` command:

```
example# syncstat zsh0


speed ipkts opkts undrun ovrrun abort crc isize osize
9600  15716 17121   0      0      1    3   98    89

example# syncstat -c zsh0

speed ipkts opkts undrun ovrrun abort crc isize osize
9600   0     0     0      0      0     0    0     0
```

In the following sample output a new line of output is generated every five seconds:

```
example# syncstat zsh0 5

ipkts opkts undrun ovrrun abort crc iutil outil
12    10      0     0      0     0   5%    4%
22    60      0     0      0     0   3%    90%
36    14      0     0      0     1   51%   2%
```

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also

`syncinit(1M)`, `syncloop(1M)`, `attributes(5)`, `zsh(7D)`

Diagnostics

`bad interval:` *arg*	The argument *arg* is expected to be an interval and could not be understood.
device `missing minor device number`	The name *device* does not end in a decimal number that can be used as a minor device number.
`baud rate not set`	The *interval* option is being used and the baud rate on the device is zero. This would cause a divide-by-zero error when computing the line utilization statistics.

Warnings

Underrun, overrun, frame-abort, and CRC errors have a variety of causes. Communication protocols are typically able to handle such errors and initiate recovery of the transmission in which the error occurred. Small numbers of such errors are not a significant problem for most protocols. However, because the overhead involved in recovering from a link error can be much greater than that of normal operation, high error rates can greatly degrade overall link throughput. High error rates are often caused by problems in the link hardware, such as

cables, connectors, interface electronics or telephone lines. They may also be related to excessive load on the link or the supporting system.

The percentages for input and output line utilization reported when using the *interval* option may occasionally be reported as slightly greater than 100% because of inexact sampling times and differences in the accuracy between the system clock and the modem clock. If the percentage of use greatly exceeds 100%, or never exceeds 50%, then the baud rate set for the device probably does not reflect the speed of the modem.

Name sysconfig – unconfigure or reconfigure a Solaris instance

Synopsis

```
/usr/sbin/sysconfig

/usr/sbin/sysconfig configure [-s] [-c config_profile.xml | dir]
     [--destructive] [-g system]

/usr/sbin/sysconfig unconfigure [-s] [--destructive]
     [-g system]

/usr/sbin/sysconfig create-profile [-o output_file [-l logfile]
     [-v verbosity] [-b] [-g system]
```

Description The `sysconfig` utility is the interface for unconfiguring and reconfiguring a Solaris instance. A Solaris instance is defined as a boot environment in either a global or a non-global zone.

There are three operations that are performed using the `sysconfig` utility: unconfiguration, configuration, and profile creation.

When `sysconfig` is called with the `unconfigure` subcommand, the system is unconfigured and left in an unconfigured state.

System configuration can occur either interactively or non-interactively. If the `configure` sub-command is invoked without a profile, the SCI Tool is activated and walks the user through the system configuration process. If the `configure` subcommand is invoked with a profile, then the configuration reads the profile and the configuration occurs non-interactively. The result in either case is a new system configuration.

The `sysconfig` command can also be used to generate a system configuration profile using the `create-profile` subcommand. The resulting profile is used with the `sysconfig configure` subcommand to configure systems non-interactively. Valid profile names include an `.xml` extension.

Configuration of a system can be performed either interactively, using the System Configuration Interactive (SCI) Tool, or non-interactively, using a system configuration profile.

The SCI tool configures the target system in an interactive way using a text user interface. It can also be used to collect information generated by the user that describes the desired configuration of the target system. The tool then generates a system configuration profile containing the desired system configuration.

The SCI tool supports configuration of freshly installed or unconfigured systems. It is designed to provide system configuration for newly created non-global zones and during text installation. If there is a need to modify the configuration of an already configured system utilizing SCI tool, such a system has to be unconfigured first before SCI tool can run.

The functional groupings that can be configured on a system are `network`, `location`, `users`, `identity`, and `kbd_layout`. Groupings can also be `unconfigured` and left in an `unconfigured` state. The default values for `unconfigured` groupings are shown below.

The following groupings are configurable.

```
Grouping        | Components            | Unconfigured State
-------------------------------------------------------------
identity        | system nodename       | unknown
-------------------------------------------------------------
kbd_layout      | Keyboard              | U.S. English
-------------------------------------------------------------
network         | network               | No network
-------------------------------------------------------------
location        | timezone              | UTC
                | locale                | C locale
-------------------------------------------------------------
users           | root                  | Empty root password
                | initial user account  | Remove user account
-------------------------------------------------------------
naming_services | DNS, NIS and LDAP     | No network naming
                | clients, nsswitch     | services
-------------------------------------------------------------
system          | all groupings         | all groupings
                |                       | unconfigured
-------------------------------------------------------------
```

Sub-commands This section describes supported subcommands and their associated options.

`unconfigure` [`-s`] [`-g` *system*] [`--destructive`]
Unconfigure a system and leave it in the `unconfigured` state.

`-s`
Shut the system down after the unconfiguration completes.

`-g` *system*
If `-g` is not specified, the user will be queried for confirmation before *system* configuration occurs.

`--destructive`
Do not preserve system data that is normally preserved during unconfiguration. By specifying this flag, the user indicates to any groupings unconfigured that data they would ordinarily preserve might be deleted.

`configure` [`-s`] [`-g` *system*] [`-c` *config_profile*.xml | *dir*] [`--destructive`]
Configure or reconfigure a system. The configure subcommand has access to the same options as the `unconfigure` subcommand. It also includes the following additional option.

`-c` *config_profile*.xml | *dir*
Provides a profile or a directory of profiles to apply during configuration. If a profile is applied, the configuration step occurs non-interactively. If no profile is provided, the interactive system configuration tool is used for the configuration of the system.

All profiles must have an `.xml` file extension.

If you supply a directory to `-c`, all profiles in that directory must be valid (correctly formed) configuration profiles.

`create-profile` [`-o` *output_file* [`-l` *logfile*] [`-v` *verbosity*] [`-b`] [`-g` *grouping*,...]
Run the SCI tool and create a system configuration profile. The default location for the profile is `/system/volatile/profile/sc_profile.xml`. The configuration generated is not applied to the system.

`-o` *output_file*
Replace the default profile location with *output_file* for the configuration profile.

`-l` *logfile*, `--log-location=`*logfile*
Location of the log file. The default is `/var/tmp/install/sysconfig.log`

`-v` *verbosity*, `--log-level=`*verbosity*
Verbosity level, one of `error`, `warn`, `info`, `debug`, or `input`. These are in order of increasing verbosity, from least to most. The default is `info`.

`-b`
Black-and-white version of SCI tool.

Examples

EXAMPLE 1 Unconfiguring and Shutting Down

The following command unconfigures the system and leaves it in an `unconfigured` state. By default, if no grouping is specified, the groupings for the whole system are unconfigured.

```
# sysconfig unconfigure -s
```

EXAMPLE 2 Unconfiguring the System

The following command unconfigures the system groupings and leaves the system `unconfigured`.

```
# sysconfig unconfigure -g system
```

EXAMPLE 3 Reconfiguring System Using SCI Tool

The following command brings up the SCI Tool to reconfigure a system.

```
# sysconfig configure
```

EXAMPLE 4 Reconfiguring Using a Profile

The following command reconfigures a system using a profile.

```
# sysconfig configure -c some_profile.xml
```

EXAMPLE 5 Creating and Using a Profile

The following sequence of commands creates a profile, then uses it to reconfigure a system.

```
# sysconfig create-profile -o /tmp/myprofile.xml
# sysconfig configure -g system -c /tmp/myprofile.xml
```

EXAMPLE 6 Configuring the System in a Zone

The following command configures the system in a zone.

```
# zlogin ZONENAME
root@ZONENAME# sysconfig configure -g system
```

Exit Status

`0`
Success.

`>0`
Failure.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os
	system/install
	system/install/configuration
	system/library/install
Interface Stability	Committed

See Also

`svcprop(1)`, `svcadm(1M)`, `svccfg(1M)`, `attributes(5)`, `attributes(5)`

Name sysdef – output system definition

Synopsis

```
/usr/sbin/sysdef [-i] [-n namelist]

/usr/sbin/sysdef [-h] [-d] [-i] [-D]
```

Description The `sysdef` utility outputs the current system definition in tabular form. It lists all hardware devices, as well as pseudo devices, system devices, loadable modules, and the values of selected kernel tunable parameters.

It generates the output by analyzing the named bootable operating system file (*namelist*) and extracting the configuration information from it.

The default system *namelist* is `/dev/kmem`.

Options

`-i` Prints the configuration information from `/dev/kmem`. This is the default and only needs to be specified if the configuration information from both `/dev/kmem` and the system file specified with the "`-n` *namelist*" option is needed.

`-n` *namelist* Specifies a *namelist* other than the default (`/dev/kmem`). The *namelist* specified must be a valid bootable operating system.

`-h` Prints the identifier of the current host in hexadecimal. If `sysdef -h` is executed within a non-global zone and the zone emulates a host identifier, then the zone's host identifier is printed. This numeric value is not guaranteed to be unique.

`-d` The output includes the configuration of system peripherals formatted as a device tree.

`-D` For each system peripheral in the device tree, display the name of the device driver used to manage the peripheral.

Examples **EXAMPLE 1** Sample output format

The following example displays the format of the `sysdef -d` output:

```
example% sysdef -d
Node 'SUNW,Ultra-5_10', unit #-1
    Node 'packages', unit #-1 (no driver)
        Node 'terminal-emulator', unit #-1 (no driver)
        Node 'deblocker', unit #-1 (no driver)
        Node 'obp-tftp', unit #-1 (no driver)
        Node 'disk-label', unit #-1 (no driver)
        Node 'SUNW,builtin-drivers', unit #-1 (no driver)
        Node 'sun-keyboard', unit #-1 (no driver)
        Node 'ufs-file-system', unit #-1 (no driver)
    Node 'chosen', unit #-1 (no driver)
    Node 'openprom', unit #-1 (no driver)
        Node 'client-services', unit #-1 (no driver)
```

EXAMPLE 1 Sample output format *(Continued)*

```
    Node 'options', unit #0
    Node 'aliases', unit #-1 (no driver)
    Node 'memory', unit #-1 (no driver)
    Node 'virtual-memory', unit #-1 (no driver)
    Node 'pci', unit #0
        Node 'pci', unit #0
            Node 'ebus', unit #0
                Node 'auxio', unit #-1 (no driver)
                Node 'power', unit #0
                Node 'SUNW,pll', unit #-1 (no driver)
                Node 'se', unit #0 (no driver)
                Node 'su', unit #0
                Node 'su', unit #1
                Node 'ecpp', unit #-1 (no driver)
                Node 'fdthree', unit #0
                Node 'eeprom', unit #-1 (no driver)
                Node 'flashprom', unit #-1 (no driver)
                Node 'SUNW,CS4231', unit #0 (no driver)
            Node 'network', unit #0
            Node 'SUNW,m64B', unit #0
            Node 'ide', unit #0
                Node 'disk', unit #-1 (no driver)
                Node 'cdrom', unit #-1 (no driver)
                Node 'sd', unit #1
                Node 'dad', unit #1
        Node 'pci', unit #-1 (no driver)
    Node 'SUNW,UltraSPARC-IIi', unit #-1 (no driver)
    Node 'pseudo', unit #0
```

[output truncated]

Files `/dev/kmem` default operating system image

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also `hostid(1)`, `prtconf(1M)`, `nlist(3ELF)`, `attributes(5)`, `zones(5)`

Name syseventadm – sysevent event specification administration

Synopsis

```
syseventadm add [-R rootdir] [-v vendor] [-p publisher]
     [-c class] [-s subclass] [-u username] path [args]

syseventadm remove [-R rootdir] [-v vendor] [-p publisher]
     [-c class] [-s subclass] [-u username] [path [args]]

syseventadm list [-R rootdir] [-v vendor] [-p publisher]
     [-c class] [-s subclass] [-u username] [path [args]]

syseventadm restart
```

Description The `syseventadm` command is an administrative front-end to add, remove and list `sysevent` event handlers. You can also restart the sysevent daemon by use of the restart command. `syseventadm` can only be run by root.

The `syseventadm add` command adds a handler for a `sysevent` event specified by at least one of vendor, publisher or class. If *class* is specified, it may be qualified with a *sub-class*. Only the values specified for *vendor*, *publisher*, *class* and *sub-class* when adding the handler are matched against `sysevent` events to determine if the specification matches the event and the handler should be run. *path* is the full pathname of the command to be run in response to matching events, with optional arguments (*args*). If username is specified, the command is invoked as user *username*, otherwise as root.

The `syseventadm remove` command removes handlers for matching sysevent event specifications. Event specifications may be matched by specifying at least one of *vendor*, *publisher*, *class*, *username* or *path*. If *class* is specified, it may be qualified with a *sub-class*. Any of *vendor*, *publisher*, *class*, *sub-class*, *username*, *path* or *args* not specified match the corresponding fields of all events. Handlers for all matching specifications are removed.

The `syseventadm list` command lists the handlers for matching `sysevent` event specifications using the same match criteria as the `remove` command but without the requirement that at least one of *vendor*, *publisher*, *class*, *username* or *path* be specified. With no match criteria, all specifications are listed. The list command output format is: [vendor=*vendor*] [publisher=*publisher*] [class=*class*] [subclass=*subclass*] [username=*username*] *path* [*args*] where each of *class*, *sub-class*, *vendor*, *publisher* and *username* is listed only if part of the match criteria for the listed specification is present.

The `syseventadm restart` command informs the `syseventd` daemon to reread the `sysevent` registry after a change has been made by adding or removing one or more `sysevent` handler specifications.

Argument Macro Substitution The `sysevent` handling facility provides extensive macro capability for constructing the command line arguments to be executed in response to an event. Macro expansion applies only to the command line *args* specified for an event handler, with macros expanded with data from the event itself. Pre-defined macros are provided for the event *class*, *subclass*, *publisher* and *vendor* information. Macros not matching one of the pre-defined macro names cause the

attribute list attached to the event to be searched for an attribute of that name, with the value of the matching attribute substituted on the command line.

Macros are introduced by the `$` character, with the macro name being the following token separated by a SPACE or TAB character. If the macro name is embedded in text, it may be delineated by `${` and `}`. A `\` before the `$` causes macro expansion not to occur.

$*class*	The class string defining the event
$*publisher*	The publisher string defining the event
$*sequence*	The sequence number of the event.
$*subclass*	The subclass string defining the event
$*timestamp*	The timestamp of the event.
$*vendor*	The vendor string defining the event

Macro names other than those pre-defined are compared against the attribute list provided with the event. An attribute with name matching the macro name causes the value of the attribute to be substituted as ASCII text on the generated command line.

Use of a macro for which no attribute with that name is defined, or for which multiple attributes with that name are provided, cause an error and the command is not invoked.

Attributes with signed data types (`DATA_TYPE_INT16`, `DATA_TYPE_INT32` and `DATA_TYPE_INT64`) are expanded as decimal digits.

Attributes with unsigned data types (`DATA_TYPE_BYTE`, `DATA_TYPE_UINT16`, `DATA_TYPE_UINT32`, `DATA_TYPE_UINT64` and `DATA_TYPE_HTTIME`) are expanded as hexadecimal digits with a `0x` prefix.

Attributes with string data type (`DATA_TYPE_STRING`) are expanded with the string data. The data is not quoted. If if it desired that the quoted strings be generated on the command line, put quotes around the macro call in the arguments.

Array types are expanded with each element expanded as defined for that scalar type, with a space separating each element substitution.

Options The `add`, `list` and `remove` subcommands support the following options:

`-c` *class*	Specify the event class, *class*.
`-p` *publisher*	Specify the event publisher, *publisher*.
`-R` *rootdir*	Specify an alternate root path, *rootdir*.

Note – The root file system of any non-global zones must not be referenced with the `-R` option. Doing so might damage the global zone's file system,

might compromise the security of the global zone, and might damage the non-global zone's file system. See zones(5).

`-s` *subclass* Specify the event subclass, *subclass*.

`-u` *username* Specify the username (*username*) to invoke the command.

`-v` *vendor* Specify the vendor (*vendor*) that defines the event. Events defined by third-party software should specify the company's stock symbol as *vendor*. Oracle-defined events use `SUNW`.

Operands The `add`, `list` and `remove` subcommands support the following options:

args Command arguments

path Full path of command to be run in response to event

Examples **EXAMPLE 1** Adding an Event Handler

The following example adds an event handler for an event defined by vendor `MYCO` ("My Company"), class `EC_ENV` and sub-class `ESC_ENV_TEMP`. The command to be run is `/opt/MYCOenv/bin/ec_env_temp`, with arguments being the class name, sub-class name and pathname derived from the event attributes. The `$` characters are preceded by a backslash to circumvent shell interpretation. There is no need to restart the service after the change since the registry is maintained on `$ALTROOT`.

```
# syseventadm add -R \ALTROOT -v MYCO -c EC_ENV -s ESC_ENV_TEMP \
     /opt/MYCOenv/bin/ec_env_temp \$class \$subclass \$pathname
```

Note the caveat on the use of the `-R` option in the description of that option, above.

EXAMPLE 2 Removing an Event Handler

The following example removes the event handler added in Example 1.

```
# syseventadm remove -R \ALTROOT -v MYCO -c EC_ENV -s ESC_ENV_TEMP \
     /opt/MYCOenv/bin/ec_env_temp \class} \subclass} \pathname}
```

Note the caveat on the use of the `-R` option in the description of that option, above.

EXAMPLE 3 Listing Event Handlers

The following example lists all event handlers for events of class `EC_ENV`, subclass `ESC_ENV_TEMP`, as defined by vendor `MYCO`:

```
# syseventadm list -v MYCO -c EC_ENV -s ESC_ENV_TEMP \
     vendor=MYCO class=EC_ENV subclass=ESC_ENV_TEMP \
     /opt/MYCOenv/bin/ec_env_temp \${class} \${subclass} \${pathname}
```

EXAMPLE 4 Listing Event Handlers

The following example lists all event handlers defined by vendor `VRTS`.

```
# syseventadm list -v VRTS
```

EXAMPLE 5 Removing Event Handlers

The following example removes all event handlers defined by vendor `VRTS`, and restarts service.

```
# syseventadm remove -v VRTS
# syseventadm restart
```

EXAMPLE 6 Listing All Event Handlers Specified to Run a Command

The following example lists all event handlers specified to run the command `/opt/MYCOenv/bin/ec_env_temp`:

```
# syseventadm list /opt/MYCOenv/bin/ec_env_temp
```

EXAMPLE 7 Removing Event Handlers and Restarting Service

The following example removes all event handlers specified to run the command `/opt/MYCOenv/bin/ec_env_temp`, and restarts service:

```
# syseventadm remove /opt/MYCOenv/bin/ec_env_temp
# syseventadm restart
```

Exit Status

The following exit values are returned:

`0` Successful completion.

`1` No matching event specification found (remove or list commands only).

`2` Incorrect command usage.

`3` Permission denied.

`4` Command failed.

`5` Out of memory.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also

`syseventd(1M)`, `sysevent_post_event(3SYSEVENT)`, `attributes(5)`, `ddi_log_sysevent(9F)`

Notes To avoid upgrade problems, packages delivering a sysevent event handler should install the event handler by running `syseventadm` from the package's postinstall script. The event handler can then be removed by running `syseventadm` from the package's preremove script using the same arguments as when added.

Name syseventconfd – kernel system event command invocation daemon

Synopsis

```
/usr/lib/sysevent/syseventconfd [-r rootdir]
```

Description `syseventconfd` is the user-level daemon that invokes user-level commands in response to kernel system events received from `syseventd(1M)`.

Options The following options are supported:

`-r` *rootdir* Cause `syseventconfd` to use an alternate root path when creating its door. The root path must match the root path used to invoke `syseventd`.

Files

`/etc/sysevent/syseventconfd_event_service`
`syseventconfd` event service door file

`/usr/lib/sysevent/modules/sysevent_conf_mod.so`
`syseventd` loadable module (SLM) managing `sysevent.conf` files

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also `svcs(1)`, `svcadm(1M)`, `syseventd(1M)`, `attributes(5)`, `smf(5)`

Notes The `syseventconfd` service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/system/sysevent:default
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. The service's status can be queried using the `svcs(1)` command.

Name syseventd – kernel system event notification daemon

Synopsis

```
/usr/lib/sysevent/syseventd [-d debug_level] [-r rootdir]
```

Description `syseventd` is a user-level daemon that accepts delivery of system event buffers from the kernel. Once an event buffer has been delivered to `syseventd`, it, in turn, attempts to propagate the event to all interested end event subscribers.

Event subscribers take the form of a `syseventd` loadable module (SLM). syseventd passes the event buffer to each of its subscribers and in return expects a notification as to the successful or unsuccessful delivery attempt.

Upon successful delivery of the event buffer to all interested event subscribers, `syseventd` frees the event buffer from the kernel event queue.

Options The following option is supported:

`-d` *debug_level* Enable debug mode. Messages are printed to the invoking user's terminal.

Exit Status The following exit values are returned:

`0` Successful completion.

`non-zero` An error occurred.

Files

`/etc/sysevent/syseventd_daemon.lock`
daemon lock file

`/etc/sysevent/sysevent_door`
kernel to `syseventd` door file

`/usr/lib/sysevent/modules`
SLM directory repository

`/usr/platform/‘uname –i‘/lib/sysevent/modules`
SLM directory repository

`/usr/platform/‘uname –m‘/lib/sysevent/modules`
SLM directory repository

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also `svcs(1)`, `svcadm(1M)`, `syseventconfd(1M)`, `attributes(5)`, `smf(5)`

Notes The `sysevent`d service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/system/sysevent:default
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. The service's status can be queried using the `svcs(1)` command.

Name syslogd – log system messages

Synopsis

```
/usr/sbin/syslogd [-d] [-f configfile] [-m markinterval]
     [-p path] [-t | -T]
```

Description `syslogd` reads and forwards system messages to the appropriate log files or users, depending upon the priority of a message and the system facility from which it originates. The configuration file `/etc/syslog.conf` (see `syslog.conf(4)`) controls where messages are forwarded. `syslogd` logs a mark (timestamp) message every *markinterval* minutes (default `20`) at priority `LOG_INFO` to the facility whose name is given as `mark` in the `syslog.conf` file.

A system message consists of a single line of text, which may be prefixed with a priority code number enclosed in angle-brackets (`< >`); priorities are defined in `<sys/syslog.h>`.

`syslogd` reads from the STREAMS log driver, `/dev/log`, and from any transport provider specified in `/etc/netconfig`, `/etc/net/transport/hosts`, and `/etc/net/transport/services`.

`syslogd` reads the configuration file when it starts up, and again whenever it receives a `HUP` signal (see `signal.h(3HEAD)`, at which time it also closes all files it has open, re-reads its configuration file, and then opens only the log files that are listed in that file. `syslogd` exits when it receives a `TERM` signal.

As it starts up, `syslogd` creates the file `/var/run/syslog.pid`, if possible, containing its process identifier (PID).

If message ID generation is enabled (see `log(7D)`), each message will be preceded by an identifier in the following format: `[ID` *msgid facility*`.`*priority*`]`. *msgid* is the message's numeric identifier described in `msgid(1M)`. *facility* and *priority* are described in `syslog.conf(4)`. `[ID 123456 kern.notice]` is an example of an identifier when message ID generation is enabled.

If the message originated in a loadable kernel module or driver, the kernel module's name (for example, `ufs`) will be displayed instead of `unix`. See `EXAMPLES` for sample output from `syslogd` with and without message ID generation enabled.

In an effort to reduce visual clutter, message IDs are not displayed when writing to the console; message IDs are only written to the log file. See `EXAMPLES`.

The `/etc/default/syslogd` file contains the default parameter settings, which are in effect if neither the `-t` nor `-T` option is selected.

The recommended way to allow or disallow message logging is through the use of the service management facility (`smf(5)`) property:

```
svc:/system/system-log/config/log_from_remote
```

This property specifies whether remote messages are logged. `log_from_remote=true` is equivalent to the `-t` command-line option and `false` is equivalent to the `-T` command-line option. The default value for `-log_from_remote` is `false`. See NOTES, below.

`LOG_FROM_REMOTE`
Specifies whether remote messages are logged. `LOG_FROM_REMOTE=NO` is equivalent to the `-t` command-line option. The default value for `LOG_FROM_REMOTE` is `YES`.

Options The following options are supported:

`-d`
Turn on debugging. This option should only be used interactively in a root shell once the system is in multi-user mode. It should *not* be used in the system start-up scripts, as this will cause the system to hang at the point where `syslogd` is started.

`-f` *configfile*
Specify an alternate configuration file.

`-m` *markinterval*
Specify an interval, in minutes, between mark messages.

`-p` *path*
Specify an alternative log device name. The default is `/dev/log`.

`-T`
Enable the `syslogd` UDP port to turn on logging of remote messages. This is the default behavior.

`-t`
Disable the `syslogd` UDP port to turn off logging of remote messages.

Examples **EXAMPLE 1** `syslogd` Output Without Message ID Generation Enabled

The following example shows the output from `syslogd` when message ID generation *is not* enabled:

```
Sep 29 21:41:18 cathy unix: alloc /: file system full
```

EXAMPLE 2 `syslogd` Output with ID generation Enabled

The following example shows the output from `syslogd` when message ID generation *is* enabled. The message ID is displayed when writing to log file`/var/adm/messages`.

```
Sep 29 21:41:18 cathy ufs: [ID 845546 kern.notice]
                                    alloc /: file system full
```

EXAMPLE 3 `syslogd` Output with ID Generation Enabled

The following example shows the output from `syslogd` when message ID generation *is* enabled when writing to the console. Even though message ID is enabled, the message ID is not displayed at the console.

```
Sep 29 21:41:18 cathy ufs: alloc /: file system full
```

EXAMPLE 4 Enabling Acceptance of UDP Messages from Remote Systems

The following commands enable `syslogd` to accept entries from remote systems.

```
# svccfg -s svc:/system/system-log setprop config/log_from_remote = true
# svcadm restart svc:/system/system-log
```

Files

`/etc/syslog.conf`
Configuration file

`/var/run/syslog.pid`
Process ID

`/etc/default/syslogd`
Contains default settings. You can override some of the settings by command-line options.

`/dev/log`
STREAMS log driver

`/etc/netconfig`
Transport providers available on the system

`/etc/net/transport/hosts`
Network hosts for each transport

`/etc/net/transport/services`
Network services for each transport

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also

`logger(1)`, `svcs(1)`, `msgid(1M)`, `svcadm(1M)`, `svccfg(1M)`, `syslog(3C)`, `syslog.conf(4)`, `attributes(5)`, `signal.h(3HEAD)`, `smf(5)`, `log(7D)`

Notes

The `mark` message is a system time stamp, and so it is only defined for the system on which `syslogd` is running. It can not be forwarded to other systems.

When `syslogd` receives a `HUP` signal, it attempts to complete outputting pending messages, and close all log files to which it is currently logging messages. If, for some reason, one (or more) of these files does not close within a generous grace period, `syslogd` discards the pending messages, forcibly closes these files, and starts reconfiguration. If this shutdown procedure is disturbed by an unexpected error and `syslogd` cannot complete reconfiguration, `syslogd` sends a mail message to the superuser on the current system stating that it has shut down, and exits.

Care should be taken to ensure that each window displaying messages forwarded by `syslogd` (especially console windows) is run in the system default locale (which is `syslogd`'s locale). If

this advice is not followed, it is possible for a `syslog` message to alter the terminal settings for that window, possibly even allowing remote execution of arbitrary commands from that window.

The `syslogd` service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/system/system-log:default
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. The service's status can be queried using the `svcs(1)` command.

When `syslogd` is started by means of `svcadm(1M)`, if a value is specified for `LOG_FROM_REMOTE` in the `/etc/defaults/syslogd` file, the SMF property `svc:/system/system-log/config/log_from_remote` is set to correspond to the `LOG_FROM_REMOTE` value and the `/etc/default/syslogd` file is modified to replace the `LOG_FROM_REMOTE` specification with the following comment:

```
# LOG_FROM_REMOTE is now set using svccfg(1m), see syslogd(1m).
```

If neither `LOG_FROM_REMOTE` nor `svc:/system/system-log/config/log_from_remote` are defined, the default is to log remote messages.

On installation, the initial value of `svc:/system/system-log/config/log_from_remote` is `false`.

Name tapes – creates /dev entries for tape drives attached to the system

Synopsis `/usr/sbin/tapes [-r root_dir]`

Description devfsadm(1M) is now the preferred command for `/dev` and `/devices` and should be used instead of `tapes`.

`tapes` creates symbolic links in the `/dev/rmt` directory to the actual tape device special files under the `/devices` directory tree. `tapes` searches the kernel device tree to see what tape devices are attached to the system. For each equipped tape drive, the following steps are performed:

1. The `/dev/rmt` directory is searched for a `/dev/rmt/`*n* entry that is a symbolic link to the `/devices` special node of the current tape drive. If one is found, this determines the logical controller number of the tape drive.
2. The rest of the special devices associated with the drive are checked, and incorrect symbolic links are removed and necessary ones added.
3. If none are found, a new logical controller number is assigned (the lowest-unused number), and new symbolic links are created for all the special devices associated with the drive.

`tapes` does not remove links to non-existent devices; these must be removed by hand.

`tapes` is run each time a reconfiguration-boot is performed, or when add_drv(1M) is executed.

Notice to Driver Writers tapes(1M) considers all devices with the node type `DDI_NT_TAPE` to be tape devices; these devices must have their minor name created with a specific format. The minor name encodes operational modes for the tape device and consists of an ASCII string of the form [`l`,`m`,`h`,`c`,`u`][`b`][`n`].

The first character set is used to specify the tape density of the device, and are named low (`l`), medium (`m`), high (`h`), compressed (`c`), and ultra (`u`). These specifiers only express a relative density; it is up to the driver to assign specific meanings as needed. For example, 9 track tape devices interpret these as actual bits-per-inch densities, where `l` means 800 BPI, `m` means 1600 BPI , and `h` means 6250 BPI, whereas 4mm DAT tapes defines `l` as standard format, and `m`, `h`, `c` and `u` as compressed format. Drivers may choose to implement any or all of these format types.

During normal tape operation (non-BSD behavior), once an EOF mark has been reached, subsequent reads from the tape device return an error. An explicit IOCTL must be issued to space over the EOF mark before the next file can be read. `b` instructs the device to observe BSD behavior, where reading at EOF will cause the tape device to automatically space over the EOF mark and begin reading from the next file.

`n` or no-rewind-on-close instructs the driver to not rewind to the beginning of tape when the device is closed. Normal behavior for tape devices is to reposition to BOT when closing. See mtio(7I).

The minor number for tape devices should be created by encoding the device's instance number using the tape macro `MTMINOR` and ORing in the proper combination of density, BSD behavior, and no-rewind flags. See `mtio(7I)`.

To prevent `tapes` from attempting to automatically generate links for a device, drivers must specify a private node type and refrain from using the node type string `DDI_NT_TAPE` when calling`ddi_create_minor_node(9F)`.

Options The following options are supported:

`-r` *root_dir* Causes `tapes` to presume that the `/dev/rmt` directory tree is found under *root_dir*, not directly under `/`.

Errors If `tapes` finds entries of a particular logical controller linked to different physical controllers, it prints an error message and exits without making any changes to the `/dev` directory, since it cannot determine which of the two alternative logical to physical mappings is correct. The links should be manually corrected or removed before another reconfiguration boot is performed.

Examples **EXAMPLE 1** Creating Tape Device Nodes From Within the Driver's `attach()` Function

This example demonstrates creating tape device nodes from within the `xktape` driver's `attach(9E)` function.

```
#include <sys/mtio.h>
struct tape_minor_info {
    char *minor_name;
    int   minor_mode;
};
/*
 * create all combinations of logical tapes
*/
static struct tape_minor_info example_tape[] = {
   {"",    0},                    /* default tape */
   {"l",   MT_DENSITY1},
   {"lb",  MT_DENSITY1 | MT_BSD},
   {"lbn", MT_DENSITY1 | MT_BSD | MT_NOREWIND},
   {"m",   MT_DENSITY2},
   {"mb",  MT_DENSITY2 | MT_BSD},
   {"mbn", MT_DENSITY2 | MT_BSD | MT_NOREWIND},
   {"h",   MT_DENSITY3},
   {"hb",  MT_DENSITY3 | MT_BSD},
   {"hbn", MT_DENSITY3 | MT_BSD | MT_NOREWIND},
   {"c",   MT_DENSITY4},
   {"cb",  MT_DENSITY4 | MT_BSD},
   {"cbn", MT_DENSITY4| MT_BSD | MT_NOREWIND},
   {NULL,  0},
};
```

EXAMPLE 1 Creating Tape Device Nodes From Within the Driver's `attach()` Function *(Continued)*

```
int
xktapeattach(dev_info_t *dip, ddi_attach_cmd_t cmd)
{
   int instance;
   struct tape_minor_info *mdp;
      /* other stuff in attach... */
   instance = ddi_get_instance(dip);

   for (mdp = example_tape; mdp->minor_name != NULL; mdp++) {
            ddi_create_minor_node(dip, mdp->minor_name, S_IFCHR,
                 (MTMINOR(instance) | mdp->minor_mode), DDI_NT_TAPE, 0);
 }
```

Installing the `xktape` driver on a Sun Fire 4800, with the driver controlling a SCSI tape (target 4 attached to an `isp(7D)` SCSI HBA) and performing a reconfiguration-boot creates the following special files in `/devices`.

```
# ls -l /devices/ssm@0,0/pci@18,700000/pci@1/SUNW,isptwo@4
crw-rw-rw-   1 root sys   33,136 Aug 29 00:02  xktape@4,0:
crw-rw-rw-   1 root sys   33,200 Aug 29 00:02  xktape@4,0:b
crw-rw-rw-   1 root sys   33,204 Aug 29 00:02  xktape@4,0:bn
crw-rw-rw-   1 root sys   33,152 Aug 29 00:02  xktape@4,0:c
crw-rw-rw-   1 root sys   33,216 Aug 29 00:02  xktape@4,0:cb
crw-rw-rw-   1 root sys   33,220 Aug 29 00:02  xktape@4,0:cbn
crw-rw-rw-   1 root sys   33,156 Aug 29 00:02  xktape@4,0:cn
crw-rw-rw-   1 root sys   33,144 Aug 29 00:02  xktape@4,0:h
crw-rw-rw-   1 root sys   33,208 Aug 29 00:02  xktape@4,0:hb
crw-rw-rw-   1 root sys   33,212 Aug 29 00:02  xktape@4,0:hbn
crw-rw-rw-   1 root sys   33,148 Aug 29 00:02  xktape@4,0:hn
crw-rw-rw-   1 root sys   33,128 Aug 29 00:02  xktape@4,0:l
crw-rw-rw-   1 root sys   33,192 Aug 29 00:02  xktape@4,0:lb
crw-rw-rw-   1 root sys   33,196 Aug 29 00:02  xktape@4,0:lbn
crw-rw-rw-   1 root sys   33,132 Aug 29 00:02  xktape@4,0:ln
crw-rw-rw-   1 root sys   33,136 Aug 29 00:02  xktape@4,0:m
crw-rw-rw-   1 root sys   33,200 Aug 29 00:02  xktape@4,0:mb
crw-rw-rw-   1 root sys   33,204 Aug 29 00:02  xktape@4,0:mbn
crw-rw-rw-   1 root sys   33,140 Aug 29 00:02  xktape@4,0:mn
crw-rw-rw-   1 root sys   33,140 Aug 29 00:02  xktape@4,0:n
```

`/dev/rmt` will contain the logical tape devices (symbolic links to tape devices in `/devices`).

```
# ls -l /dev/rmt
/dev/rmt/0    -> ../../devices/[....]/xktape@4,0:
/dev/rmt/0b   -> ../../devices/[....]/xktape@4,0:b
/dev/rmt/0bn  -> ../../devices/[....]/xktape@4,0:bn
```

EXAMPLE 1 Creating Tape Device Nodes From Within the Driver's `attach()` Function *(Continued)*

```
/dev/rmt/0c   -> ../../devices/[....]/xktape@4,0:c
/dev/rmt/0cb  -> ../../devices/[....]/xktape@4,0:cb
/dev/rmt/0cbn -> ../../devices/[....]/xktape@4,0:cbn
/dev/rmt/0cn  -> ../../devices/[....]/xktape@4,0:cn
/dev/rmt/0h   -> ../../devices/[....]/xktape@4,0:h
/dev/rmt/0hb  -> ../../devices/[....]/xktape@4,0:hb
/dev/rmt/0hbn -> ../../devices/[....]/xktape@4,0:hbn
/dev/rmt/0hn  -> ../../devices/[....]/xktape@4,0:hn
/dev/rmt/0l   -> ../../devices/[....]/xktape@4,0:l
/dev/rmt/0lb  -> ../../devices/[....]/xktape@4,0:lb
/dev/rmt/0lbn -> ../../devices/[....]/xktape@4,0:lbn
/dev/rmt/0ln  -> ../../devices/[....]/xktape@4,0:ln
/dev/rmt/0m   -> ../../devices/[....]/xktape@4,0:m
/dev/rmt/0mb  -> ../../devices/[....]/xktape@4,0:mb
/dev/rmt/0mbn -> ../../devices/[....]/xktape@4,0:mbn
/dev/rmt/0mn  -> ../../devices/[....]/xktape@4,0:mn
/dev/rmt/0n   -> ../../devices/[....]/xktape@4,0:n
```

Files

`/dev/rmt/*` logical tape devices

`/devices/*` tape device nodes

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also `add_drv(1M)`, `devfsadm(1M)`, `attributes(5)`, `isp(7D)`, `devfs(7FS)`, `mtio(7I)`, `attach(9E)`, `ddi_create_minor_node(9F)`

Writing Device Drivers

Bugs `tapes` silently ignores malformed minor device names.

Name th_define – create fault injection test harness error specifications

Synopsis

```
th_define [-n name -i instance| -P path] [-a acc_types]
     [-r reg_number] [-l offset [length]]
     [-c count [failcount]] [-o operator [operand]]
     [-f acc_chk] [-w max_wait_period [report_interval]]

or

th_define [-n name -i instance| -P path]
     [-a log [acc_types] [-r reg_number] [-l offset [length]]]
     [-c count [failcount]] [-s collect_time] [-p policy]
     [-x flags] [-C comment_string]
     [-e fixup_script [args]]

or

th_define [-h]
```

Description The `th_define` utility provides an interface to the `bus_ops` fault injection `bofi` device driver for defining error injection specifications (referred to as errdefs). An errdef corresponds to a specification of how to corrupt a device driver's accesses to its hardware. The command line arguments determine the precise nature of the fault to be injected. If the supplied arguments define a consistent errdef, the `th_define` process will store the errdef with the `bofi` driver and suspend itself until the criteria given by the errdef become satisfied (in practice, this will occur when the access counts go to zero).

You use the `th_manage(1M)` command with the `start` option to activate the resulting errdef. The effect of `th_manage` with the `start` option is that the `bofi` driver acts upon the errdef by matching the number of hardware accesses—specified in *count*, that are of the type specified in *acc_types*, made by instance number *instance*—of the driver whose name is *name*, (or by the driver instance specified by *path*) to the register set (or DMA handle) specified by *reg_number*, that lie within the range *offset* to *offset* + *length* from the beginning of the register set or DMA handle. It then applies *operator* and *operand* to the next *failcount* matching accesses.

If *acc_types* includes `log`, `th_define` runs in automatic test script generation mode, and a set of test scripts (written in the Korn shell) is created and placed in a sub-directory of the current directory with the name <*driver*>`.test.`<*id*> (for example, `glm.test.978177106`). A separate, executable script is generated for each access handle that matches the logging criteria. The log of accesses is placed at the top of each script as a record of the session. If the current directory is not writable, file output is written to standard output. The base name of each test file is the driver name, and the extension is a number that discriminates between different access handles. A control script (with the same name as the created test directory) is generated that will run all the test scripts sequentially.

Executing the scripts will install, and then activate, the resulting error definitions. Error definitions are activated sequentially and the driver instance under test is taken offline and brought back online before each test (refer to the `-e` option for more information). By default, logging applies to all `PIO` accesses, all interrupts, and all DMA accesses to and from areas

mapped for both reading and writing. You can constrain logging by specifying additional *acc_types*, *reg_number*, *offset* and *length*. Logging will continue for *count* matching accesses, with an optional time limit of *collect_time* seconds.

Either the `-n` or `-P` option must be provided. The other options are optional. If an option (other than `-a`) is specified multiple times, only the final value for the option is used. If an option is not specified, its associated value is set to an appropriate default, which will provide maximal error coverage as described below.

Options The following options are available:

`-n` *name*
: Specify the name of the driver to test. (String)

`-i` *instance*
: Test only the specified driver instance (-1 matches all instances of driver). (Numeric)

`-P` *path*
: Specify the full device path of the driver to test. (String)

`-r` *reg_number*
: Test only the given register set or DMA handle (-1 matches all register sets and DMA handles). (Numeric)

`-a` *acc_types*
: Only the specified access types will be matched. Valid values for the *acc_types* argument are `log`, `pio`, `pio_r`, `pio_w`, `dma`, `dma_r`, `dma_w` and `intr`. Multiple access types, separated by spaces, can be specified. The default is to match all hardware accesses.

  If *acc_types* is set to `log`, logging will match all `PIO` accesses, interrupts and DMA accesses to and from areas mapped for both reading and writing. `log` can be combined with other *acc_types*, in which case the matching condition for logging will be restricted to the specified addional *acc_types*. Note that `dma_r` will match only DMA handles mapped for reading only; `dma_w` will match only DMA handles mapped for writing only; `dma` will match only DMA handles mapped for both reading and writing.

`-l` *offset* [*length*]
: Constrain the range of qualifying accesses. The *offset* and *length* arguments indicate that any access of the type specified with the `-a` option, to the register set or DMA handle specified with the `-r` option, lie at least *offset* bytes into the register set or DMA handle and at most *offset* + *length*

bytes into it. The default for *offset* is 0. The default for *length* is the maximum value that can be placed in an `offset_t` C data type (see `types.h`). Negative values are converted into unsigned quantities. Thus, `th_define -l 0 -1` is maximal.

`-c` *count* [*failcount*]

Wait for *count* number of matching accesses, then apply an operator and operand (see the `-o` option) to the next *failcount* number of matching accesses. If the access type (see the `-a` option) includes logging, the number of logged accesses is given by *count* + *failcount* - 1. The -1 is required because the last access coincides with the first faulting access.

Note that access logging may be combined with error injection if *failcount* and *operator* are nonzero and if the access type includes logging and any of the other access types (`pio`, `dma` and `intr`) See the description of access types in the definition of the `-a` option, above.

When the *count* and *failcount* fields reach zero, the status of the errdef is reported to standard output. When all active errdefs created by the `th_define` process complete, the process exits. If *acc_types* includes `log`, *count* determines how many accesses to log. If *count* is not specified, a default value is used. If *failcount* is set in this mode, it will simply increase the number of accesses logged by a further *failcount* - 1.

`-o` *operator* [*operand*]

For qualifying PIO read and write accesses, the value read from or written to the hardware is corrupted according to the value of *operator*:

`EQ` *operand* is returned to the driver.

`OR` *operand* is bitwise ORed with the real value.

`AND` *operand* is bitwise ANDed with the real value.

`XOR` *operand* is bitwise XORed with the real value.

For PIO write accesses, the following operator is allowed:

`NO` Simply ignore the driver's attempt to write to the hardware.

Note that a driver performs PIO via the `ddi_get`*X*`()`, `ddi_put`*X*`()`, `ddi_rep_get`*X*`()` and `ddi_rep_put`*X*`()` routines (where *X* is 8, 16, 32 or 64). Accesses made using `ddi_get`*X*`()` and `ddi_put`*X*`()` are treated as a single access, whereas an access made using the `ddi_rep_*`(9F) routines are broken down into their respective number of accesses, as given by the *repcount* parameter to these DDI calls. If the access is performed via a DMA handle, *operator* and *value* are applied to every access that comprises the DMA request. If interference with interrupts has been requested then the operator may take any of the following values:

`DELAY` After *count* accesses (see the `-c` option), delay delivery of the next *failcount* number of interrupts for *operand* number of microseconds.

`LOSE` After *count* number of interrupts, fail to deliver the next *failcount* number of real interrupts to the driver.

`EXTRA` After *count* number of interrupts, start delivering *operand* number of extra interrupts for the next *failcount* number of real interrupts.

The default value for *operand* and *operator* is to corrupt the data access by flipping each bit (XOR with -1).

`-f` *acc_chk*

If the *acc_chk* parameter is set to 1 or `pio`, then the driver's calls to `ddi_check_acc_handle(9F)` return `DDI_FAILURE` when the access count goes to 1. If the *acc_chk* parameter is set to 2 or `dma`, then the driver's calls to `ddi_check_dma_handle(9F)` return `DDI_FAILURE` when the access count goes to 1.

`-w` *max_wait_period* [*report_interval*]

Constrain the period for which an error definition will remain active. The option applies only to non-logging errdefs. If an error definition remains active for *max_wait_period* seconds, the test will be aborted. If *report_interval* is set to a nonzero value, the current status of the error definition is reported to standard output every *report_interval* seconds. The default value is zero. The status of the errdef is reported in parsable format (eight fields, each separated by a colon (`:`) character, the last of which is a string enclosed by double quotes and the remaining seven fields are integers):

ft:*mt*:*ac*:*fc*:*chk*:*ec*:*s*:*"message"* which are defined as follows:

ft	The UTC time when the fault was injected.
mt	The UTC time when the driver reported the fault.
ac	The number of remaining non-faulting accesses.
fc	The number of remaining faulting accesses.
chk	The value of the *acc_chk* field of the errdef.
ec	The number of fault reports issued by the driver against this errdef (*mt* holds the time of the initial report).
s	The severity level reported by the driver.
"message"	Textual reason why the driver has reported a fault.

`-h`

Display the command usage string.

`-s` *collect_time*

If *acc_types* is given with the `-a` option and includes `log`, the errdef will log accesses for *collect_time* seconds (the default is to log until the log becomes full). Note that, if the errdef

specification matches multiple driver handles, multiple logging errdefs are registered with the `bofi` driver and logging terminates when all logs become full or when *collect_time* expires or when the associated errdefs are cleared. The current state of the log can be checked with the `th_manage(1M)` command, using the `broadcast` parameter. A log can be terminated by running `th_manage(1M)` with the `clear_errdefs` option or by sending a `SIGALRM` signal to the `th_define` process. See `alarm(2)` for the semantics of `SIGALRM`.

`-p` *policy*

Applicable when the *acc_types* option includes `log`. The parameter modifies the policy used for converting from logged accesses to errdefs. All policies are inclusive:

- Use `rare` to bias error definitions toward rare accesses (default).
- Use `operator` to produce a separate error definition for each operator type (default).
- Use `common` to bias error definitions toward common accesses.
- Use `median` to bias error definitions toward median accesses.
- Use `maximal` to produce multiple error definitions for duplicate accesses.
- Use `unbiased` to create unbiased error definitions.
- Use `onebyte`, `twobyte`, `fourbyte`, or `eightbyte` to select errdefs corresponding to 1, 2, 4 or 8 byte accesses (if chosen, the `-xr` option is enforced in order to ensure that `ddi_rep_*()` calls are decomposed into *multiple single accesses*).
- Use `multibyte` to create error definitions for multibyte accesses performed using `ddi_rep_get*()` and `ddi_rep_put*()`.

Policies can be combined by adding together these options. See the NOTES section for further information.

-x *flags*

Applicable when the *acc_types* option includes `log`. The *flags* parameter modifies the way in which the `bofi` driver logs accesses. It is specified as a string containing any combination of the following letters:

`w` Continuous logging (that is, the log will wrap when full).

`t` Timestamp each log entry (access times are in seconds).

`r` Log repeated I/O as individual accesses (for example, a `ddi_rep_get16(9F)` call which has a repcount of *N* is logged *N* times with each transaction logged as size 2 bytes. Without this option, the default logging behavior is to log this access once only, with a transaction size of twice the *repcount*).

-C *comment_string*

Applicable when the *acc_types* option includes `log`. It provides a comment string to be placed in any generated test scripts. The string must be enclosed in double quotes.

-e *fixup_script* [*args*]

Applicable when the *acc_types* option includes `log`. The output of a logging errdefs is to generate a test script for each driver access handle. Use this option to embed a command in the resulting script before the errors are injected. The generated test scripts will take an instance offline and bring it back online before injecting errors in order to bring the instance into a known fault–free state. The executable *fixup_script* will be called twice with the set of optional *args*— once just before the instance is taken offline and again after the instance has been brought online. The following variables are passed into the environment of the called executable:

`DRIVER_PATH` Identifies the device path of the instance.

`DRIVER_INSTANCE` Identifies the instance number of the device.

`DRIVER_UNCONFIGURE`	Has the value 1 when the instance is about to be taken offline.
`DRIVER_CONFIGURE`	Has the value 1 when the instance has just been brought online.

Typically, the executable ensures that the device under test is in a suitable state to be taken offline (unconfigured) or in a suitable state for error injection (for example configured, error free and servicing a workload). A minimal script for a network driver could be:

```
#!/bin/ksh

driver=xyznetdriver
ifnum=$driver$DRIVER_INSTANCE

if [[ $DRIVER_CONFIGURE = 1 ]]; then
    ifconfig $ifnum plumb
    ifconfig $ifnum ...
    ifworkload start $ifnum
elif [[ $DRIVER_UNCONFIGURE = 1 ]]; then
    ifworkload stop $ifnum
    ifconfig $ifnum down
    ifconfig $ifnum unplumb
fi
exit $?
```

The `-e` option must be the last option on the command line.

If the `-a log` option is selected but the `-e` option is not given, a default script is used. This script repeatedly attempts to detach and then re-attach the device instance under test.

Examples

Examples of Error Definitions

```
th_define -n foo -i 1 -a log
```

Logs all accesses to all handles used by instance 1 of the `foo` driver while running the default workload (attaching and detaching the instance). Then generates a set of test scripts to inject appropriate errdefs while running that default workload.

```
th_define -n foo -i 1 -a log pio
```

Logs PIO accesses to each PIO handle used by instance 1 of the `foo` driver while running the default workload (attaching and detaching the instance). Then generates a set of test scripts to inject appropriate errdefs while running that default workload.

```
th_define -n foo -i 1 -p onebyte median -e fixup arg -now
```

Logs all accesses to all handles used by instance 1 of the `foo` driver while running the workload defined in the fixup script `fixup` with arguments `arg` and `-now`. Then generates a set of test scripts to inject appropriate errdefs while running that workload. The resulting error definitions are requested to focus upon single byte accesses to locations that are accessed a *median* number of times with respect to frequency of access to I/O addresses.

```
th_define -n se -l 0x20 1 -a pio_r -o OR 0x4 -c 10 1000
```

Simulates a stuck serial chip command by forcing 1000 consecutive read accesses made by any instance of the `se` driver to its command status register, thereby returning status busy.

```
th_define -n foo -i 3 -r 1 -a pio_r -c 0 1 -f 1 -o OR 0x100
```

Causes 0x100 to be ORed into the next physical I/O read access from any register in register set 1 of instance 3 of the `foo` driver. Subsequent calls in the driver to `ddi_check_acc_handle()` return `DDI_FAILURE`.

```
th_define -n foo -i 3 -r 1 -a pio_r -c 0 1 -o OR 0x0
```

Causes 0x0 to be ORed into the next physical I/O read access from any register in register set 1 of instance 3 of the `foo` driver. This is of course a no-op.

```
th_define -n foo -i 3 -r 1 -l 0x8100 1 -a pio_r -c 0 10 -o EQ 0x70003
```

Causes the next ten next physical I/O reads from the register at offset 0x8100 in register set 1 of instance 3 of the `foo` driver to return 0x70003.

```
th_define -n foo -i 3 -r 1 -l 0x8100 1 -a pio_w -c 100 3 -o AND
0xfffffffffffefff
```

The next 100 physical I/O writes to the register at offset 0x8100 in register set 1 of instance 3 of the `foo` driver take place as normal. However, on each of the three subsequent accesses, the 0x1000 bit will be cleared.

```
th_define -n foo -i 3 -r 1 -l 0x8100 0x10 -a pio_r -c 0 1 -f 1 -o XOR 7
```

Causes the bottom three bits to have their values toggled for the next physical I/O read access to registers with offsets in the range 0x8100 to 0x8110 in register set 1 of instance 3 of the `foo` driver. Subsequent calls in the driver to `ddi_check_acc_handle()` return `DDI_FAILURE`.

```
th_define -n foo -i 3 -a pio_w -c 0 1 -o NO 0
```

Prevents the next physical I/O write access to any register in any register set of instance 3 of the `foo` driver from going out on the bus.

```
th_define -n foo -i 3 -l 0 8192 -a dma_r -c 0 1 -o OR 7
```

Causes 0x7 to be ORed into each `long long` in the first 8192 bytes of the next DMA read, using any DMA handle for instance 3 of the `foo` driver.

```
th_define -n foo -i 3 -r 2 -l 0 8 -a dma_r -c 0 1 -o OR 0x7070707070707070
```

Causes 0x70 to be ORed into each byte of the first `long long` of the next DMA read, using the DMA handle with sequential allocation number 2 for instance 3 of the `foo` driver.

```
th_define -n foo -i 3 -l 256 256 -a dma_w -c 0 1 -f 2 -o OR 7
```

Causes 0x7 to be ORed into each `long long` in the range from offset 256 to offset 512 of the next DMA write, using any DMA handle for instance 3 of the `foo` driver. Subsequent calls in the driver to `ddi_check_dma_handle()` return `DDI_FAILURE`.

```
th_define -n foo -i 3 -r 0 -l 0 8 -a dma_w -c 100 3 -o AND 0xfffffffffffefff
```

The next 100 DMA writes using the DMA handle with sequential allocation number 0 for instance 3 of the `foo` driver take place as normal. However, on each of the three subsequent accesses, the 0x1000 bit will be cleared in the first `long long` of the transfer.

```
th_define -n foo -i 3 -a intr -c 0 6 -o LOSE 0
```

Causes the next six interrupts for instance 3 of the `foo` driver to be lost.

```
th_define -n foo -i 3 -a intr -c 30 1 -o EXTRA 10
```

When the thirty-first subsequent interrupt for instance 3 of the `foo` driver occurs, a further ten interrupts are also generated.

```
th_define -n foo -i 3 -a intr -c 0 1 -o DELAY 1024
```

Causes the next interrupt for instance 3 of the `foo` driver to be delayed by 1024 microseconds.

Notes The policy option in the `th_define -p` syntax determines how a set of logged accesses will be converted into the set of error definitions. Each logged access will be matched against the chosen policies to determine whether an error definition should be created based on the access.

Any number of policy options can be combined to modify the generated error definitions.

Bytewise Policies These select particular I/O transfer sizes. Specifing a byte policy will exclude other byte policies that have not been chosen. If none of the byte type policies is selected, all transfer sizes are treated equally. Otherwise, only those specified transfer sizes will be selected.

`onebyte`	Create errdefs for one byte accesses (`ddi_get8()`)
`twobyte`	Create errdefs for two byte accesses (`ddi_get16()`)
`fourbyte`	Create errdefs for four byte accesses (`ddi_get32()`)
`eightbyte`	Create errdefs for eight byte accesses (`ddi_get64()`)
`multibyte`	Create errdefs for repeated byte accesses (`ddi_rep_get*()`)

Frequency of Access Policies

The frequency of access to a location is determined according to the access type, location and transfer size (for example, a two-byte read access to address A is considered distinct from a four-byte read access to address A). The algorithm is to count the number of accesses (of a given type and size) to a given location, and find the locations that were most and least accessed (let *maxa* and *mina* be the number of times these locations were accessed, and *mean* the total number of accesses divided by total number of locations that were accessed). Then a rare access is a location that was accessed less than

(mean - mina) / 3 + mina

times. Similarly for the definition of common accesses:

maxa - (maxa - mean) / 3

A location whose access patterns lies within these cutoffs is regarded as a location that is accessed with median frequency.

`rare`	Create errdefs for locations that are rarely accessed.
`common`	Create errdefs for locations that are commonly accessed.
`median`	Create errdefs for locations that are accessed a median frequency.

Policies for Minimizing errdefs

If a transaction is duplicated, either a single or multiple errdefs will be written to the test scripts, depending upon the following two policies:

`maximal`	Create multiple errdefs for locations that are repeatedly accessed.
`unbiased`	Create a single errdef for locations that are repeatedly accessed.
`operators`	For each location, a default operator and operand is typically applied. For maximal test coverage, this default may be modified using the `operators` policy so that a separate errdef is created for each of the possible corruption operators.

See Also `kill(1)`, `th_manage(1M)`, `alarm(2)`, `ddi_check_acc_handle(9F)`, `ddi_check_dma_handle(9F)`

Name th_manage – manage the fault injection test harness

Synopsis

```
th_manage name instance command

th_manage path command
```

Description `th_manage` applies the action specified by *command* to the instance specified by *instance* of the driver specified by *name* (or the driver instance specified by *path*). The driver instance must be running fault injection specifications (errdefs) defined by `th_define(1M)`.

`th_manage` supports several commands that operate on the driver instance specified by *name* and *instance* (or *path*). The commands are:

`broadcast`	Awaken all `th_define` processes, causing them to display their current status and exit if the errdef is now defunct (that is, if *count*, *failcount* , and *acc_chk* are all zero).
`clear_acc_chk`	Awaken all `th_define` processes. If *count* and *failcount* are already zero, then set *acc_chk* to zero, so that `th_define` exits once it has displayed its status.
`clear_errdefs`	Awaken all `th_define processes`. *count*, *failcount* and *acc_chk* are all set to zero so that all `th_define` commands exit once they have displayed their status.
`clear_errors`	Awaken all `th_define` processes. If *count* is already zero, set *failcount* and *acc_chk* to zero, so that `th_define` exits once it has displayed its status.
`get_handles`	List all the access handles.
`start`	Begin or resume execution of all errdefs.
`stop`	Suspend all errdefs for this *name* and *instance* (or *path).*

Examples **EXAMPLE 1** Useful Commands

To begin the tests, enter:

```
# th_manage foo 0 start
```

To check the status of the errdefs, enter:

```
# th_manage foo 0 broadcast
```

This causes each `th_define` process to print out its current status.

If the driver has reported a fatal error, you can take the driver offline using `libdevice`, clear the error condition by entering:

```
# th_manage foo 0 clear_acc_chk
```

EXAMPLE 1 Useful Commands *(Continued)*

or

```
# th_manage foo 0 clear_errors
```

and bring the driver online again using `libdevice`.

To terminate testing, enter:

```
# th_manage foo 0 clear_errdefs
```

See Also `th_define(1M)`

Name tic – terminfo compiler

Synopsis

```
tic [-v [n]] [-c] file
```

Description The command `tic` translates a `terminfo` file from the source format into the compiled format. The results are placed in the directory `/usr/share/lib/terminfo`. The compiled format is necessary for use with the library routines in `curses(3CURSES)`.

If the environment variable `TERMINFO` is set, the compiled results are placed there instead of `/usr/share/lib/terminfo`.

Total compiled entries cannot exceed 4096 bytes. The name field cannot exceed 128 bytes. Terminal names exceeding 14 characters will be truncated to 14 characters and a warning message will be printed.

Options The following options are supported:

`-c` Specifies to check only `file` for errors. Errors in `use=` links are not detected.

`-v`[*n*] Specify that (verbose) output be written to standard error trace information showing `tic`'s progress. The optional integer *n* is a number from 1 to 10, indicating the desired level of detail of information. If *n* is omitted, the default level is 1. If *n* is specified and greater than 1, the level of detail is increased.

Operands `file` Contains one or more `terminfo` terminal descriptions in source format [see `terminfo(4)`]. Each description in the file describes the capabilities of a particular terminal. When a `use=`*entry-name* field is discovered in a terminal entry currently being compiled, `tic` reads in the binary from `/usr/share/lib/terminfo` to complete the entry. (Entries created from `file` will be used first. If the environment variable `TERMINFO` is set, that directory is searched instead of `/usr/share/lib/terminfo`.) `tic` duplicates the capabilities in *entry-name* for the current entry, with the exception of those capabilities that are explicitly defined in the current entry.

Files `/usr/share/lib/terminfo/?/*` Compiled terminal description database

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also `captoinfo(1M)`, `infocmp(1M)`, `curses(3CURSES)`, `terminfo(4)`, `attributes(5)`

Notes When an entry, for example, `entry_name_1`, contains a `use=`*entry_name_2* field, any canceled capabilities in *entry_name_2* must also appear in `entry_name_1` before `use=` for these capabilities to be canceled in `entry_name_1`.

Name tncfg – configure trusted networking properties

Synopsis

```
tncfg [-t template] [-e|-S [files|ldap]] [subcommand]
tncfg [-t template] [-e|-S [files|ldap]] -f command_file
tncfg -z zone [-e] [subcommand]
tncfg -z zone [-e] -f command_file
tncfg help
```

Description The `tncfg` utility creates, modifies, and displays the configuration of various networking properties related to Trusted Extensions. The command requires that the SMF service, `svc:/system/labeld` is enabled. It can be executed only in the global zone.

Templates A template is a collection of network security properties that define the rules for labeling packets received from remote hosts. Four host types are supported: `cipso`, `unlabeled`, `adaptive`, and `netif`. Each template must specify one of these four host types. Hosts that are trusted to specify their own labels are assigned to `cipso` templates. Otherwise, hosts can be assigned to `unlabeled` or `adaptive` templates. When using an `unlabeled` template, a single default label must be specified. When using an `adaptive` template, the default label is not specified. Instead, the label is derived from the default label of the IP interface on which the packet is received.

The labeling properties of an IP interface are specified using the `netif` template type. For `netif` templates, the host address properties correspond to local interfaces, instead of remote hosts. These templates should be used when separate single-level networks are connected to corresponding labeled zones. Therefore, the default label of a `netif` template must equal the label of every zone with a network interface whose IP address matches a host address in that template. Futhermore, the lower-link corresponding to any such matching zone interfaces can only be assigned to other zones sharing the same label.

Hosts can be specified using hostnames, IP addresses, or masks. When masks are used, a prefix length that specifies how many bits are required for a match must be appended . Hosts cannot be assigned to more than one template. When masks are used, the entry with the longest matching prefix is used to associate a host with a template. Packets from hosts without a matching template are dropped.

Each template must include an upper and lower bound specifying the accreditation range of accepted labels. Additionally, up to four auxiliary labels can be specified to enumerate labels outside of this range. Services bound to multilevel ports can accept packets from hosts whose labels are within the accreditation range, or match one of the auxiliary labels.

Normally, the template settings and their corresponding hosts are persistently maintained in local files or by means of an LDAP directory, depending on the `-S` option. These settings are automatically loaded into kernel memory when the user commits the updates. If the `-e` (ephemeral) option is specified, only the current in-memory properties are displayed and updated. However, the list of hosts associated with an in-memory template is generally incomplete. To view the matching template for a specific host, use the `get` subcommand.

By default, an unlabeled template, `admin_low`, is installed with a default label of `ADMIN_LOW`, and two mask entries matching any IPv4 or IPv6 address, so that the global zone is initially able to contact any `unlabeled` hosts. It is recommended to remove these two mask entries once your network security policy is established. An additional template, `cipso`, is installed with no matching hosts. By default all local IP addresses are implicitly associated with this template, but it is recommended that they should be explicitly added to this or a new `cipso`-type template.

Searches for template and host entries are resolved in the order specified by means of the name service configuration file, `/etc/nsswitch.conf`. The keywords, `tnrhdb` and `tnrhtp`, are used to specify the search order for hosts and templates, respectively. Both the `files` and `ldap` repositories are supported, but it is recommended to specify `files` first.

Creating or modifying a template requires the authorization `solaris.label.network.manage`, which is included in the Object Label Management rights profile.

Zones

Zones are isolated execution environments described in `zones(5)`. Trusted Extensions requires a zone brand called `labeled` to which special properties apply. Each labeled zone must have a unique label property at which it executes processes. This is also the label at which it will accept packets from remote hosts for services bound to single-level ports. Explicit multilevel port can also be specified. Services with the privilege `net_bindmlp` can bind to these ports, and accept packets within the accreditation range or the auxiliary label set associated with the remote host.

Non-global zones must be configured using `zonecfg(1M)` prior to configuring these properties. In general, updates to each zone's properties, including the global zone, are applied when it is booted. However, the multilevel port properties of running zones are reloaded into kernel memory when updates are committed. If the `-e` (ephemeral) option is specified, the zone must be in the ready or running state, and only its multilevel port properties can be updated.

Creating or modifying a zone's trusted networking properties requires the authorization `solaris.label.zone.manage`, which is included in the Object Label Management rights profile.

Properties

The set of valid properties depends on whether the `-t` or `-z` option was used. The two sets are referred to as the template context and the zone context.

Only a single property value can be specified at a time. Values containing white space must be quoted. An equal sign is required between the property and its value.

The values that can be specified in the template context properties are described below.

`name`=*template_name*

The initial value for the name is specified using `-t` option using the command line. If the name is changed, the current template properties are applied to the newly named template.

In this way an existing template can be cloned for subsequent editing. However, to avoid conflicts, the host entries from the initial template are not copied to the new template. The specified name must not match an existing template.

`host_type=cipso|unlabeled|adaptive|netif`
When the `unlabeled` or `netif` host types are used, the value specified using the the `def_label` property is implicitly applied to the received packets. The `cipso` host type is used for hosts that are trusted to explicitly label their packets. The `adaptive` host type is used for hosts whose label is derived from the interface on which their packets are received. The default is `unlabeled`.

`def_label=`*sensitivity_label*
The default label assigned to IP packets that are not explicitly labeled by means of `cipso` or IPsec.

`doi=`*integer*
A positive integer specifying the Domain of Interpretation for the binary representation of the labels. The default is `1`.

`min_label=`*sensitivity_label*
The minimum label in the accreditation range for IP packets that are accepted by multilevel services.

`max_label=`*sensitivity_label*
The maximum label in the accreditation range for IP packets that are accepted by multilevel services.

`aux_label=`*sensitivity_label*
Additional labels, outside of the accreditation range, for IP packets that are accepted by multilevel services. Up to four labels may be specified, using the `add` subcommand repetitively.

`host=`*hostname*|*IP address*[/*prefix*]
A hostname or an IP address to which the template properties apply. For IP addresses, both IPv4 and IPv6 formats can be used, followed by an optional slash and prefix length specifying the number of bits to match again IP addresses. The IPv4 address `0.0.0.0` has an implied prefix length of zero, and matches any IPv4 address. Multiple host values can be specified, using the `add` subcommand repetitively. There is no specific limit.

The values that may be specified in the zone context properties are described below.

`name=`*zone_name*
The name of the zone, which must have previously been configured using `zonecfg(1M)`. The initial value for the name is specified using `-z` option on the command line. If the name is changed, the current zone properties are applied to the newly named zone. In this way an existing template can be cloned for subsequent editing. However, to avoid conflicts, the initial label value is not copied to the new zone configuration. The specified name must correspond to an existing zone without a trusted networking configuration.

`primary=yes|no`
Although multiple zones can share a single sensitivity label, at most one zone for each label can have its primary property set to `yes`. This indicates that the zone should be selected as the target of any operation that specifies only a label instead of a zone name, such as choosing the label of a desktop workspace, sharing an IP address, or relabeling a file.

By default all zones are created with their primary property set to `yes`, unless an existing primary zone with a matching label already exists. Primary zones are not required for any label except `admin_low`, which is reserved for the global zone. When primary is set to `no`, the desktop packages are not installed by default.

`label=`*sensitivity_label*
The sensitivity label of the zone. It must be unique if the zone's primary property is set to `yes`. Otherwise, if the session is interactive, the user is given the option to set the zone's primary property to `no`. The global zone value must be `admin_low`.

`visible=yes|no`
Specifies whether the zone responds to ping requests from hosts whose labels don't match the zone's label. The default is `no`.

`mlp_private=`*port*`|[-`*port2*`]/tcp|udp`
A single port number, or a range of ports that privileged services can bind to and then accept requests from clients whose labels are with the accreditation range or the set of auxiliary labels specified in their matching templates. The port specification must be followed by a a protocol, either `tcp` or `udp`. This value applies to all interfaces that are private to the zone. Multiple `mlp_private` values can be specified, using the `add` subcommand repetitively. This is only limited by the number of available ports.

`mlp_shared=`*port*`|[-`*port2*`]/tcp|udp`
A single port number, or a range of ports that privileged services can bind to and then accept requests from clients whose labels are with the accreditation range or the set of auxiliary labels specified in their matching templates. The port specification must be followed by a protocol, either `tcp` or `udp`. This value applies to any `all-zones` interfaces, and must not overlap with the `mlp_shared` ports specifications for other zones. Multiple `mlp_shared` values can be specified, using the `add` subcommand repetitively. This is only limited by the number of available ports.

Options The following options are supported:

`-e`
Specifies that the data is ephemeral, affecting only what is currently loaded into kernel memory.

`-f` *command_file*
Specifies the name of `tncfg` command file. *command_file* is a text file of `tncfg` subcommands, one per line.

`-t` *template*
Specifies the template name. If the named template does not exist, a new template is created. If neither `-t` nor `-z` is specified, the template context is assumed using `cipso` as the default template name.

`-S` *repository*
The valid repositories are `files` and `ldap`. The repository specifies which name service will be updated. The default repository is files.

`-z` *zone*
Specifies the zone name. The zone must have previously been configured by means of `zonecfg`(1M).

Sub-commands Subcommands can be provided on the command line or interactively. Multiple subcommands, separated by semicolons, can be specified on the command line by enclosing the entire set in quotation marks. The lack of subcommands implies an interactive session, during which auto-completion of subcommands can be invoked using the tab key.

The add, clear, and remove subcommands are used for properties that can accept multiple values. However, only one value can be specified at a time.

Subcommands which can result in destructive actions or loss of work have an `-F` option to force the action. If input is from a terminal device, the user is prompted when appropriate if such a subcommand is given without the `-F` option. Otherwise, the action is disallowed, with a diagnostic message written to standard error.

The following subcommands are supported:

`add` *property-name=property-value*
Adds the specified value to the current property values. This subcommand can only be applied to properties that accept multiple values. Use the `set` subcommand for single-value properties.

`clear` *property-name*
Clears all of the values for the property. Only those properties that accept multiple assignments, using the `add` subcommand, can be cleared.

`commit`
Commits the current configuration from memory to stable storage and into the kernel. The configuration must be committed for the changes to take effect. Until the in-memory configuration is committed, you can remove changes with the `revert` subcommand. The commit operation is attempted automatically upon completion of a `tncfg` session. Since a configuration must be correct to be committed, this operation automatically does a `verify`.

`delete` [`-F`]
Deletes the specified template or zone configuration from the current name service.

Specify the `-F` option to force the action. If the deletion is allowed, its action is instantaneous and the session is terminated.

`export` [`-f` *output-file*]
Displays configuration to standard output. Use the `-f` option to display the configuration to *output-file*. This option produces output in a form suitable for use in a command file.

`get` host=*hostname* | *IP address*[/*prefix*]
Displays the template name corresponding to the specified host using the kernel's in-memory mapping.

`help` [`usage`] [`subcommands`] [`properties`] [*subcommand*] [*property*]
Displays general help or help about a given topic.

`info` *property-name*
Displays information about the current template or zone, or the specified property in a parseable format.

`list`
Lists the names of the templates or zones that have been configured.

`remove` *property-name=property-value*
Removes the specified value from the property. Only those properties that accept multiple assignments, using the `add` subcommand, can be removed.

`set` *property-name=property-value*
Sets a given property name to the given value. Properties that can take multiple values are assigned using the `add` subcommand, instead of `set`.

`verify`
Verifies the current configuration for correctness:

- The required properties are specified;
- the values are valid for each key word;
- the user is authorized to specify the values.

`revert` [`-F`]
Causes the configuration to revert to the last committed state. The `-F` option can be used to force the action.

`exit` [`-F`]
Exits the `tncfg` session. A `commit` is automatically attempted if needed. You can also use an EOF character to exit `tncfg`. The `-F` option can be used to force the action.

Examples

EXAMPLE 1 Using the `info` Subcommand

The command below displays the properties of a `cipso` template are displayed. The subcommand is specified on the command line.

```
example% tncfg -t cipso info
            name=cipso
            host_type=cipso
            doi=1
            min_label=ADMIN_LOW
```

EXAMPLE 1 Using the `info` Subcommand *(Continued)*

```
        max_label=ADMIN_LOW
        host=10.5.233.74
```

EXAMPLE 2 Using the `export` Subcommand

The following example shows an interactive session that exports the configuration of a zone in a format the could be imported to another machine with an equivalent zone.

```
example% tncfg -t public
tncfg:public> export
set name=public
set host_type=cipso
set doi=1
set def_label="PUBLIC"
set min_label="PUBLIC"
set max_label="CONFIDENTIAL : NEED TO KNOW"
add aux_label="SANDBOX PLAYGROUND"
add host=myserver.oracle.com
add host=10.5.0.0/16
tncfg:public> exit
```

EXAMPLE 3 Assigning Properties to a Zone

In the following example, the public zone is configured to be a multi-level NFS server.

```
example% tncfg -z public
tncfg:public> info
     name=public
     label=PUBLIC
     visible=no
tncfg:public> add mlp_private=111/tcp
tncfg:public> add mlp_private=111/ucp
tncfg:public> add mlp_private=2049/tcp
tncfg:public> commit
tncfg:public> exit
```

Exit Status

`0`

Successful completion.

`1`

An error occurred.

Files

- `/etc/security/tsol/tnrhtp`
- `/etc/security/tsol/tnrhdb`
- `/etc/security/tsol/tnzonecfg`

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/trusted
Interface Stability	See below.

The invocation and subcommands are committed. Output, except for the `export` and `info` subcommands, is Not-an-Interface.

See Also tnctl(1M), tnd(1M), tninfo(1M), txzonemgr(1M), zonecfg(1M), nsswitch.conf(4), attributes(5), labels(5), zones(5)

Notes The Labeled Zone Manager, txzonemgr(1M), is an alternative application for configuring Trusted Extensions. It invokes the `tncfg` command internally, and provides an interactive GUI-based user interface.

Name tnchkdb – check file syntax of trusted network databases

Synopsis

```
/usr/sbin/tnchkdb  [-h path] [-t path] [-z path]
```

Description `tnchkdb` checks the syntax of the `tnrhtp`, `tnrhdb`, and `tnzonecfg` databases. By default, the *path* for each file is:

- `/etc/security/tsol/tnrhtp`
- `/etc/security/tsol/tnrhdb`
- `/etc/security/tsol/tnzonecfg`

You can specify an alternate path for any or all of the files by specifying that path on the command line by using the `-h` (`tnrhdb`), `-t` (`tnrhtp`) and `-z` (`tnzonecfg`) options. The options are useful when testing a set of modified files before installing the files as new system databases.

All three database files are checked for integrity. `tnchkdb` returns an exit status of `0` if all of the files are syntactically and, to the extent possible, semantically correct. If one or more files have errors, then an exit status of `1` is returned. If there are command line problems, such as an unreadable file, an exit status of `2` is returned. Errors are written to standard error.

To avoid cascading errors, when there are errors in `tnrhtp`, the template names in `tnrhdb` are not validated.

`tnchkdb` can be run at any label, but the standard `/etc/security/tsol` files are visible only in the global zone.

Options

`-h` [*path*] Check *path* for proper `tnrhdb` syntax. If *path* is not specified, then check `/etc/security/tsol/tnrhdb`.

`-t` [*path*] Check *path* for proper `tnrhtp` syntax. If *path* is not specified, then check `/etc/security/tsol/tnrhtp`.

`-z` [*path*] Check *path* for proper `tnzonecfg` syntax. If *path* is not specified, then check `/etc/security/tsol/tnzonecfg`.

Examples **EXAMPLE 1** Sample Error Message

The `tnchkdb` command checks for CIPSO errors. In this example, the `admin_low` template has an incorrect value of `ADMIN_HIGH` for its default label.

```
# tnchkdb
checking /etc/security/tsol/tnrhtp ...
tnchkdb: def_label classification 7fff is invalid for cipso labels:
line 14 entry admin_low
tnchkdb: def_label compartments 241-256 must be zero for cipso labels:
line 14 entry admin_low
checking /etc/security/tsol/tnrhdb ...
checking /etc/security/tsol/tnzonecfg ...
```

Files

`/etc/security/tsol/tnrhdb`	Trusted network remote-host database
`/etc/security/tsol/tnrhtp`	Trusted network remote-host templates
`/etc/security/tsol/tnzonecfg`	Trusted zone configuration database

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/trusted
Interface Stability	See below.

The command line is Committed. The output is Uncommitted.

See Also `tnd(1M)`, `tnctl(1M)`, `attributes(5)`

Trusted Extensions Configuration and Administration

Notes The functionality described on this manual page is available only if the system is configured with Trusted Extensions.

It is possible to have inconsistent but valid configurations of `tnrhtp` and `tnrhdb` when LDAP is used to supply missing templates.

Name tnctl – configure Trusted Extensions network parameters

Synopsis

```
/usr/sbin/tnctl [-dfv] [-h host [/prefix] [:template]]
     [-m zone:mlp:shared-mlp][-t template [:key=val [;key=val]]]
     [-HTz] file]
```

Description `tnctl` provides an interface to manipulate trusted network parameters in the Solaris kernel.

As part of Solaris Trusted Extensions initialization, `tnctl` is run in the global zone by an `smf(5)` script during system boot. The `tnctl` command is not intended to be used during normal system administration. Instead, if a local trusted networking database file is modified, the administrator first issues `tnchkdb(1M)` to check the syntax, and then refreshes the kernel copy with this command:

```
# svcadm restart svc:/network/tnctl
```

See `WARNINGS` about the risks of changing remote host and template information on a running system.

Options

`-d`
Delete matching entries from the kernel. The entries will be deleted from kernel cache table that matches the template host type. For example, if the template host type is `NETIF`, the entry will be deleted from the kernel interface cache; otherwise, the entry will be deleted from kernel host cache. If a template is not specified, `tnctl` attempts to delete the entry from kernel cache table.

When deleting MLPs, the MLP range must match exactly. MLPs are specified in the form:

port`[`*-port*`]`*/protocol*

Where *port* can be a number in the range 1 to 65535. or any known service (see `services(4)`), and protocol can be a number in the range 1 to 255, or any known protocol (see `protocols(4)`).

`-f`
Flush all kernel entries before loading the entries that are specified on the command line. The flush does not take place unless at least one entry parsed successfully. Both host cache and interface cache entries are flushed.

`-v`
Turn on verbose mode.

`-h` *host*`[`*/prefix*`][`*:template*`]`
Update the kernel remote-host cache on the local host for the specified *host* or, if a template name is given, change the kernel's cache to use the specified *template*. If *prefix* is not specified, then an implied prefix length is determined according to the rules used for interpreting the `tnrhdb`. If `-d` is specified, then a template name cannot be specified.

`-m` *zone*:*mlp*:*shared-mlp*
Modify the kernel's multilevel port (MLP) configuration cache for the specified *zone*. *zone* specifies the zone to be updated. *mlp* and *shared-mlp* specify the MLPs for the zone-specific and shared IP addresses. The *shared-mlp* field is effective in the global zone only.

`-t` *template*[*key*=*val*[;*key*=*val*]]
Update the kernel template cache for *template* or, if a list of *key*=*val* pairs is given, change the kernel's cache to use the specified entry. If `-d` is specified, then *key*=*val* pairs cannot be specified.

`-T` *file*
Load all template entries in *file* into the kernel cache.

`-H` *file*
Load all remote host entries in *file* into the kernel cache.

`-z` *file*
Load just the global zone's MLPs from *file* into the kernel cache. To reload MLPs for a non-global zone, reboot the zone:

```
# zoneadm -z non-global zone reboot
```

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/trusted
Interface Stability	Uncommitted

Files

`/etc/security/tsol/tnrhdb`	Trusted network remote-host database
`/etc/security/tsol/tnrhtp`	Trusted network remote-host templates
`/etc/security/tsol/tnzonecfg`	Trusted zone configuration database
`/etc/nsswitch.conf`	Configuration file for the name service switch

See Also `svcs(1)`, `svcadm(1M)`, `tninfo(1M)`, `tnd(1M)`, `tnchkdb(1M)`, `zoneadm(1M)`, `nsswitch.conf(4)`, `protocols(4)`, `services(4)`, `attributes(5)`, `smf(5)`

Trusted Extensions Configuration and Administration

Warnings Changing a template while the network is up can change the security view of an undetermined number of hosts.

Notes The functionality described on this manual page is available only if the system is configured with Trusted Extensions.

The `tnctl` service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/network/tnctl
```

The service's status can be queried by using `svcs(1)`. Administrative actions on this service, such as refreshing the kernel cache, can be performed using `svcadm(1M)`, as in:

```
svcadm restart svc:/network/tnctl
```

Name tnd – trusted network daemon

Synopsis `/usr/sbin/tnd [-p` *poll-interval*`]`

Description The `tnd` (trusted network daemon) initializes the kernel with trusted network databases and also reloads the databases on demand from an LDAP server and local files. `tnd` follows the order specified in the `nsswitch.conf(4)` file when loading configuration databases.

`tnd` is intended to be started from the `svc:/network/tnd` `smf(5)` service during the boot process in the global zone only when the system has been configured as an LDAP client. Systems that use only local files for the trusted network databases use `tnctl(1M)` instead of `tnd`.

`tnd` loads the following databases into the kernel: the remote host database, `tnrhdb`, and the remote-host template database, `tnrhtp`. `tnd` also periodically scans for changes in the associated LDAP database or local databases and updates the kernel cache accordingly.

If a local trusted networking database file is modified, the administrator should run `tnchkdb(1M)` to check the syntax, and should also run `svcadm refresh svc:/network/tnd` to initiate an immediate database scan by `tnd`.

`tnd` is intended to be started from an `smf(5)` script and to run in the global zone. The following `svcadm` commands signal `tnd` to perform specific actions:

`svcadm refresh svc:/network/tnd`	Initiates a rescan of the local and LDAP `tnrhdb` and `tnrhtp` databases. `tnd` updates the kernel database with any changes found.
`svcadm disable svc:/network/tnd`	Terminates the `tnd` daemon. No changes are made to the kernel database.

Running `tnd` in debug mode is determined by the value of the following service management facility (SMF) property:

```
tnd/debug_level = 0
```

A value of `0`, as above, prevents debug information from being collected; `1` turns on debugging. The default value is `0`. Debug output is sent to the `/var/tsol/tndlog` log file.

Options

`-p` *poll-interval*	Set poll interval to *poll-interval* seconds. The default *poll-interval* is 1800 seconds (30 minutes).

Examples **EXAMPLE 1** Changing the Poll Interval

The following command changes the polling interval to one hour, and puts this interval in the SMF repository. At the next boot, the `tnd` poll interval will be one hour.

```
# svccfg -s network/tnd setprop tnd/poll_interval=3600
```

EXAMPLE 1 Changing the Poll Interval *(Continued)*

The following command changes the polling interval, but does not update the repository. At the next boot, the `tnd` poll interval remains the default, 30 minutes.

```
# tnd -p 3600
```

Files

`/etc/security/tsol/tnrhdb`	Trusted network remote-host database
`/etc/security/tsol/tnrhtp`	Trusted network remote-host templates
`/etc/security/tsol/tnzonecfg`	Trusted zone configuration database
`/etc/nsswitch.conf`	Configuration file for the name service switch

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/trusted
Interface Stability	See below.

The command invocation is Committed. The service is Private.

See Also `svcs(1)`, `svcadm(1M)`, `tninfo(1M)`, `tnctl(1M)`, `tnchkdb(1M)`, `nsswitch.conf(4)`, `attributes(5)`, `smf(5)`

Trusted Extensions Configuration and Administration

Notes The functionality described on this manual page is available only if the system is configured with Trusted Extensions.

The `tnd` service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/network/tnd
```

The service's status can be queried by using `svcs(1)`. Administrative actions on this service, such as requests to restart the daemon, can be performed using `svcadm(1M)`, as in:

```
svcadm restart svc:/network/tnd
```

Name tninfo – print kernel-level network information and statistics

Synopsis
```
/usr/sbin/tninfo [-h hostname] [-i interface_ip_address]
     [-m zone-name] [-t template]
```

Description `tninfo` provides an interface to retrieve and display kernel-level network information and statistics.

Options

`-h` *hostname*
Display the security structure for the specified host in the remote-host cache. The output should reflect what is specified in the `tnrhdb` database.

`-i` *interface_ip_address*
Display the template associated with the IP interface that hosts the IP address.

`-m` *zone-name*
Display the MLP configuration associated with the specified zone. The output should reflect what is specified in the `tnzonecfg` database.

`-t` *template*
Display the structure associated with the specified *template*. The output should reflect what is specified in the `tnrhtp` database.

Examples

EXAMPLE 1 Displaying Remote Host Structures Cached in the Kernel

This example shows the remote host structures cached in the kernel. The output reflects the definition in the `tnrhdb` database.

```
# tninfo -h machine1
   IP address= 192.168.8.61
   Template = cipso
```

EXAMPLE 2 Displaying Multilevel Ports for the Global Zone

This example shows the kernel-cached MLPs for the global zone. The output reflects the definition in the `tnzonecfg` database, plus any dynamically allocated MLPs. `private` indicates zone-specific MLPs.

```
# tninfo -m global
private:23/tcp;111/tcp;111/udp;515/tcp;2049/tcp;6000-6003/tcp;
        32812/tcp;36698/ip;38634/tcp;64365/ip
shared: 6000-6003/tcp
```

EXAMPLE 3 Displaying the cipso and netif Template Definitions

This example shows the kernel-cached `cipso` template definition. The output reflects the definition in the `tnrhtp` database.

```
# tninfo -t cipso
====================================
   Remote Host Template Table Entries:

   ___________________________
```

EXAMPLE 3 Displaying the cipso and netif Template Definitions *(Continued)*

```
   template: cipso
   host_type: CIPSO
   doi: 1
   min_sl: ADMIN_LOW
   hex: ADMIN_LOW
   max_sl: ADMIN_HIGH
   hex: ADMIN_HIGH

# tninfo -t netif
====================================
  Remote Host Template Table Entries:

  ____________________________
  template: netif
  host_type: NETIF
  doi: 1
  def_label: ADMIN_LOW
  hex: ADMIN_LOW
  min_sl: ADMIN_LOW
  hex: ADMIN_LOW
  max_sl: ADMIN_LOW
  hex: ADMIN_LOW
```

EXAMPLE 4 Displaying Template Name Associated with an IP Interface

The example command below shows the kernel-cached template name associated with the interface on which the IP address is hosted.

```
# tninfo -i 192.168.3.10
IP address= 192.168.3.10
Template = netif
```

Files

`/etc/security/tsol/tnrhdb`	Trusted network remote-host database
`/etc/security/tsol/tnrhtp`	Trusted network remote-host templates
`/etc/security/tsol/tnzonecfg`	Trusted zone configuration database

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/trusted
Interface Stability	See below.

The command line is Committed. The output is Uncommitted.

See Also `tnd`(1M), `tnctl`(1M), `attributes`(5)

Trusted Extensions Configuration and Administration

Notes The functionality described on this manual page is available only if the system is configured with Trusted Extensions.

Name tpmadm – administer Trusted Platform Module

Synopsis

```
tpmadm status

tpmadm init

tpmadm clear [owner | lock]

tpmadm auth

tpmadm keyinfo [uuid]

tpmadm deletekey uuid

tpmadm migrate export UUID [MigDataFile MigKeyfile]

tpmadm migrate import UUID [MigDataFile MigKeyfile [ParentUUID]
     [NewKeyUUID]]

tpmadm pcrextend pcr [filename]

tpmadm pcrreset pcr
```

Description A Trusted Platform Module (TPM) is a hardware component that provides for protected key storage and reliable measurements of software used to boot the operating system. The `tpmadm` utility is used to initialize and administer the TPM so that it can be used by the operating system and other programs.

The TPM subsystem can store and manage an unlimited number of keys for use by the operating system and by users. Each key is identified by a Universally Unique Identifier, or UUID.

Although the TPM can hold only a limited number of keys at any given time, the supporting software automatically loads and unloads keys as needed. When a key is stored outside the TPM, it is always encrypted or "wrapped" by its parent key so that the key is never exposed in readable form outside the TPM.

Before the TPM can be used, it must be initialized by the platform owner. This process involves setting an owner password which is used to authorize privileged operations.

Although the TPM owner is similar to a traditional superuser, there are two important differences. First, process privilege is irrelevant for access to TPM functions. All privileged operations require knowledge of the owner password, regardless of the privilege level of the calling process. Second, the TPM owner is not able to override access controls for data protected by TPM keys. The owner can effectively destroy data by re-initializing the TPM, but he cannot access data that has been encrypted using TPM keys owned by other users.

Sub-commands The following subcommands are used in the form:

```
# tpamadm <subcommand> [operand]
```

`status`
Report status information about the TPM. Output includes basic information about whether ownership of the TPM has been established, current PCR contents, and the usage of TPM resources such as communication sessions and loaded keys.

`init`
Initialize the TPM for use. This involves taking ownership of the TPM by setting the owner authorization password. Taking ownership of the TPM creates a new storage root key, which is the ancestor of all keys created by this TPM. Once this command is issued, the TPM must be reset using BIOS operations before it can be re-initialized.

`auth`
Change the owner authorization password for the TPM.

`clear lock`
Clear the count of failed authentication attempts. After a number of failed authentication attempts, the TPM responds more slowly to subsequent attempts, in an effort to thwart attempts to find the owner password by exhaustive search. This command, which requires the correct owner password, resets the count of failed attempts.

`clear owner`
Deactivate the TPM and return it to an unowned state. This operation, which requires the current TPM owner password, invalidates all keys and data tied to the TPM. Before the TPM can be used again, the system must be restarted, the TPM must be reactivated from the BIOS or ILOM pre-boot environment, and the TPM must be re-initialized using the `tpmadm init` command.

`keyinfo` [*uuid*]
Report information about keys stored in the TPM subsystem. Without additional arguments, this subcommand produces a brief listing of all keys. If the UUID of an individual key is specified, detailed information about that key is displayed.

`deletekey` *uuid*
Delete the key with the specified UUID from the TPM subsystem's persistent storage.

`migrate export` *UUID* [*MigDataFile MigKeyfile*]
Create the initial migration blob and key for the persistent key *UUID*. If necessary, the user will be prompted for a password to access the key being migrated. Additionally, the user will be prompted to create an authorization password for the migration key. This operation creates two files: a migration blob (wrapped key) and a migration key to be used in future migrations. The output files will be named `tpm-migration.dat` and `tpm-migration.key`, unless they are specified on the command line. This operation will require TPM owner authorization as well as authorization passwords for any parent keys that must be loaded in order to load the key being exported. The user will be prompted for all authorization passwords as needed.

`migrate import` [*MigDataFile MigKeyFile* [*ParentUUID*] [*NewKeyUUID*]]
Import a key into the user's persistent key DB. The key will be made a child of the given *ParentUUID*. If *ParentUUID* is not given, the imported key will be a child of the system

MRK UUID. If *NewKeyUUID* is not given, the system will generate a new UUID and report it to the user upon completion of the command. The user will be prompted for the migration password used in the "export" step. When the `migrate import` command is given with no arguments, the import operation will attempt the migration of the `SYSTEM MRK` UUID to the current SRK in the system key `db`. When importing an `MRK`, the user must have the TPM Administration rights (see `prof_attr(4)`) or have root privilege (`euid == 0`). This operation will require TPM owner authorization as well as authorization passwords for any parent keys that must be loaded in order to load the key being exported. The user will be prompted for all authorization passwords as needed.

`pcrextend` *pcr* [*filename*]
Create an SHA-1 hash of the contents of *filename* and perform a PCR Extend operation on the indicated PCR using the hash value as the data to be extended. If a filename is not specified, the data is read from stdin.

`pcrreset` *pcr*
Reset the indicated PCR to its initial state (all zeros).

Exit Status After completing the requested operation, `tpmadm` exits with one of the following status values.

`0`
Successful termination.

`1`
Failure. The requested operation could not be completed.

`2`
Usage error. The `tpmadm` command was invoked with invalid arguments.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os
Interface Stability	Committed

See Also `prof_attr(4)`, `attributes(5)`

See also the `tcsd(8)` man page, available in the `SUNWtss` package.

TCG Software Stack (TSS) Specifications:
`https://www.trustedcomputinggroup.org/specs/TSS` (as of the date of publication)

Notes `tpmadm` communicates with the TPM device through the `tcsd` service. `tcsd` must be running before using the `tpmadm` command. If `tcsd` is not running, `tpmadm` will generate the following error:

```
Connect context: Communication failure (0x3011)
```

See `tcsd(8)` for more details.

Name traceroute – print the route packets take to network host

Synopsis

```
traceroute [-adFIlnSvx] [-A addr_family] [-c traffic_class]
     [-f first_hop] [-g gateway [-g gateway...] | -r]
     [-i iface] [-L flow_label] [-m max_hop]
     [-P pause_sec] [-p port] [-Q max_timeout]
     [-q nqueries] [-s src_addr] [-t tos] [-w wait_time] host
     [packetlen]
```

Description The Internet is a large and complex aggregation of network hardware, connected by gateways. Tracking the route a packet follows can be difficult. The utility `traceroute` traces the route that an IP packet follows to another internet host.

The `traceroute` utility utilizes the both the IPv4 and IPv6 protocols. Use the `-A` option to override the default behavior. `traceroute` uses the IPv4 protocol *ttl* (time to live) field or the IPv6 field *hop limit*. It attempts to elicit an ICMP or ICMP6 `TIME_EXCEEDED` response from each *gateway* along the path, and a `PORT_UNREACHABLE`(or `ECHO_REPLY` if `-I` is used) response from the destination host. It starts by sending probes with a *ttl* or *hop limit* of `1` and increases by `1` until it either gets to the host, or it hits the maximum *max_hop*. The default maximum *max_hop* is 30 hops, but this can be set by the `-m` option.

Three probes are sent at each *ttl* (*hop limit*) setting, and a line is printed showing the *ttl* (*hop limit*), the hostname and the address of the gateway, and the *rtt* (round trip time) of each probe. The number of probes may be specifically set using the `-q` option. If the probe answers come from different gateways, the hostname and the address of each responding system will be printed. If there is no response within a 5 second timeout interval, an asterisk (`*`) is displayed for that probe. The `-w` option may be used to set the timeout interval. Other possible annotations that may appear after the time are:

`!`
: the *ttl* (*hop limit*) value in the received packet is `<= 1`.

`!H`
: host unreachable.

`!X`
: communication administratively prohibited.

`<!N>`
: ICMP (ICMP6) unreachable code N.

The following annotations appear only for IPv4:

`!F`
: fragmentation needed. This should never occur. If this is seen, the associated gateway is broken.

`!N`
: network unreachable.

`!P`
protocol unreachable.

`!S`
source route failed. It is likely that the gateway does not support source routing.

`!T`
unreachable for the specified tos (type-of-service).

`!U`
source host isolated or precedence problem.

The following annotations appear only for IPv6:

`!A`
host unreachable for a reason other than lack of an entry in the routing table.

`!B`
packet too big.

`!E`
destination is not a neighbor.

`!R`
unrecognized next header.

If almost all the probes result in some kind of unreachable code, then `traceroute` gives up and exits.

The destination *host* is not supposed to process the UDP probe packets, so the destination *port* default is set to an unlikely value. However, if some application on the destination is using that value, the value of *port* can be changed with the `-p` option.

The only mandatory parameter is the destination *host* name or IP number. The default probe datagram length is 40 bytes (60 bytes for IPv6), but this may be increased by specifying a packet length (in bytes) after the destination *host* name.

All integer arguments to `traceroute` can be specified in either decimal or hexadecimal notation. For example, *packetlen* can be specified either as `256` or `0x100`.

Options `-A` *addr_family*
Specify the address family of the target host. *addr_family* can be either `inet` or `inet6`. Address family determines which protocol to use. For an argument of `inet`, IPv4 is used. For `inet6`, IPv6 is used.

By default, if the name of a host is provided, not the literal IP address, and a valid IPv6 address exists in the name service database, `traceroute` will use this address. Otherwise, if the name service database contains an IPv4 address, it will try the IPv4 address.

Specify the address family `inet` or `inet6` to override the default behavior. If the argument specified is `inet`, `traceroute` will use the IPv4 address associated with the hostname. If none exists, `traceroute` will state that the host is unknown and exit. It will not try to determine if an IPv6 address exists in the name service database.

If the specified argument is `inet6`, `traceroute` will use the IPv6 address that is associated with the hostname. If none exists, `traceroute` will state that the host is unknown and exit.

`-a`
Probe all of the addresses of a multi-homed destination. The output looks like `traceroute` has been run once for each IP address of the destination. If this option is used together with `-A`, `traceroute` probes only the addresses that are of the specified address family. While probing one of the addresses of the destination, user can skip to the next address by sending a `SIGINT`, or exit `traceroute` by sending a `SIGQUIT` signal. See `signal(3C)`

`-c` *traffic_class*
Specify the traffic class of probe packets. The value must be an integer in the range from 0 to 255. Gateways along the path may route the probe packet differently depending upon the value of *traffic_class* set in the probe packet. This option is valid only on IPv6.

`-d`
Set the `SO_DEBUG` socket option.

`-F`
Turn off fragmentation. For IPv4, this means setting the Don't Fragment bit. For IPv4 and IPv6, this means do not allow fragmentation as the datagrams are sent. If the packetlen exceeds the MTU, then traceroute may report that sending failed due to Message too long.

`-f` *first_hop*
Set the starting *ttl* (*hop limit*) value to *first_hop*, to override the default value `1`. `traceroute` skips processing for those intermediate gateways which are less than *first_hop* hops away.

`-g` *gateway*
Specify a loose source route *gateway*. The user can specify more than one *gateway* by using `-g` for each gateway. The maximum number of gateways is 8 for IPv4 and 127 for IPv6. Note that some factors such as the link MTU can further limit the number of gateways for IPv6. This option cannot be used with the `-r` option.

Only users with the `{PRIV_NET_RAWACCESS}` privilege can specify a loose source route with this option.

`-I`
Use ICMP (ICMP6) `ECHO` instead of UDP datagrams.

`-i` *iface*
For IPv4, this option specifies a network interface to obtain the source IP address. This is normally only useful on a multi-homed host. The `-s` option is also another way to do this.

For IPv6, it specifies the network interface on which probe packets are transmitted. The argument can be either an interface index, for example, `1`, `2`, or an interface name, for example, `eri0`, `hme0`.

`-L` *flow_label*
Specify the flow label of probe packets. The value must be an integer in the range from 0 to 1048575. This option is valid only on IPv6.

`-l`
Print the value of the *ttl* (*hop limit*) field in each packet received.

`-m` *max_hop*
Set the maximum *ttl* (*hop limit*) used in outgoing probe packets. The default is 30 hops, which is the same default used for TCP connections.

`-n`
Print hop addresses numerically rather than symbolically and numerically. This saves a nameserver address-to-name lookup for each gateway found on the path.

`-P` *pause_sec*
Specify a delay, in seconds, to pause between probe packets. This may be necessary if the final destination does not accept undeliverable packets in bursts. By default, `traceroute` sends the next probe as soon as it has received a reply. Note that *pause_sec* is a real number.

`-p` *port*
Set the base UDP *port* number used in probes.The default is 33434. `traceroute` hopes that nothing is listening on UDP *ports* `(base+(nhops-1)*nqueries)` to `(base+(nhops*nqueries)-1)` at the destination host, so that an ICMP (ICMP6) `PORT_UNREACHABLE` message will be returned to terminate the route tracing. If something is listening on a *port* in the default range, this option can be used to select an unused *port* range. `nhops` is defined as the number of hops between the source and the destination.

`-Q` *max_timeout*
Stop probing this hop after *max_timeout* consecutive timeouts are detected. The default value is 5. Useful in combination with the `-q` option if you have specified a large *nqueries* probe count.

`-q` *nqueries*
Set the desired number of probe queries. The default is 3.

`-r`
Bypass the normal routing tables and send directly to a host on an attached network. If the host is not on a directly-attached network, an error is returned. This option can be used to send probes to a local host through an interface that has been dropped by the router daemon. See `in.routed(1M)`. You cannot use this option if the `-g` option is used.

`-S`
Display a summary of how many probes were not answered for each hop.

`-s` *src_addr*
Use the following address, which usually is given as a literal IP address, not a hostname, as the source address in outgoing probe packets. On multi-homed hosts, those with more than one IP address, this option can be used to force the source address to be something other than the IP address `traceroute` picks by default. If the IP address is not one of this machine's interface addresses, an error is returned and nothing is sent. For IPv4, when used together with the `-i` option, the given IP address should be configured on the specified interface. Otherwise, an error will be returned. In the case of IPv6, the interface name and the source address do not have to match.

`-t` *tos*
Set the *tos*(type-of-service) in probe packets to the specified value. The default is zero. The value must be an integer in the range from 0 to 255. Gateways along the path may route the probe packet differently depending upon the *tos* value set in the probe packet. This option is valid only on IPv4.

`-v`
Verbose output. For each hop, the size and the destination of the response packets is displayed. Also ICMP (ICMP6) packets received other than `TIME_EXCEEDED` and `UNREACHABLE` are listed as well.

`-w` *waittime*
Set the time, in seconds, to wait for a response to a probe. The default is 5 seconds.

`-x`
Prevent `traceroute` from calculating checksums. Checksums are usually required for the last hop when using ICMP `ECHO` probes. This option is valid only on IPv4. See the `-I` option.

When specified from within a shared-IP zone, this option has no effect as the checksum is always calculated by the operating system in this case.

Operands The following operands are supported:

host
The network host.

Examples **EXAMPLE 1** Sample Output From the `traceroute` Utility

Some sample output from the `traceroute` utility might be:

```
istanbul% traceroute london
traceroute: Warning: london has multiple addresses;  \
  using 4::114:a00:20ff:ab3d:83ed
traceroute: Warning: Multiple interfaces found; \
  using 4::56:a00:20ff:fe93:8dde @ eri0:2
traceroute to london (4::114:a00:20ff:ab3d:83ed), 30 hops max, \
  60 byte packets
1  frbldg7c-86 (4::56:a00:20ff:fe1f:65a1)  1.786 ms  1.544 ms  1.719 ms
```

EXAMPLE 1 Sample Output From the `traceroute` Utility *(Continued)*

```
2  frbldg7b-77 (4::255:0:0:c0a8:517)  2.587 ms 3.001 ms  2.988 ms
3  london (4::114:a00:20ff:ab3d:83ed)  3.122 ms  2.744 ms  3.356 ms
```

The target host, london, has both IPv4 and IPv6 addresses in the name service database. According to the default behavior, `traceroute` uses IPv6 address of the destination host.

EXAMPLE 2 Using the `traceroute` Utility For a Host Which has Only IPv4 Addresses

In the following examples, `traceroute` is tracking the route to host sanfrancisco, which has only IPv4 addresses in the name service database. Therefore traceroute uses only IPv4 addresses. The following shows the 7-hop path that a packet would follow from the host istanbul to the host sanfrancisco.

```
istanbul% traceroute sanfrancisco
traceroute: Warning: Multiple interfaces found; using 172.31.86.247 @eri0
traceroute to sanfrancisco (172.29.64.39), 30 hops max, 40 byte packets
1  frbldg7c-86 (172.31.86.1)  1.516 ms  1.283 ms  1.362 ms
2  bldg1a-001 (172.31.1.211)  2.277 ms  1.773 ms  2.186 ms
3  bldg4-bldg1 (172.30.4.42)  1.978 ms  1.986 ms  13.996 ms
4  bldg6-bldg4 (172.30.4.49)  2.655 ms  3.042 ms  2.344 ms
5  ferbldg11a-001 (172.29.1.236)  2.636 ms  3.432 ms  3.830 ms
6  frbldg12b-153 (172.29.153.72)  3.452 ms  3.146 ms  2.962 ms
7  sanfrancisco (172.29.64.39)  3.430 ms  3.312 ms  3.451 ms
```

EXAMPLE 3 Using the `traceroute` Utility With Source Routing

The following example shows the path of a packet that goes from istanbul to sanfrancisco through the hosts cairo and paris, as specified by the `-g` option. The `-I` option makes `traceroute` send ICMP `ECHO` probes to the host sanfrancisco. The `-i` options sets the source address to the IP address configured on the interface `qe0`.

```
istanbul% traceroute -g cairo -g paris -i qe0 -q 1 -I sanfrancisco
traceroute to sanfrancisco (172.29.64.39), 30 hops max, 56 byte packets
1  frbldg7c-86 (172.31.86.1)  2.012 ms
2  flrbldg7u (172.31.17.131)  4.960 ms
3  cairo (192.168.163.175) 4.894 ms
4  flrbldg7u (172.31.17.131)  3.475 ms
5  frbldg7c-017 (172.31.17.83)  4.126 ms
6  paris (172.31.86.31) 4.086 ms
7  frbldg7b-82 (172.31.82.1)  6.454 ms
8  bldg1a-001 (172.31.1.211)  6.541 ms
9  bldg6-bldg4 (172.30.4.49)  6.518 ms
10  ferbldg11a-001 (172.29.1.236)  9.108 ms
11  frbldg12b-153 (172.29.153.72)  9.634 ms
12  sanfrancisco (172.29.64.39)  14.631 ms
```

Exit Status The following exit values are returned:

`0`
Successful operation.

`>0`
An error occurred.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also `netstat(1M)`, `signal(3C)`, `ping(1M)`, `attributes(5)`, `privileges(5)`, `zones(5)`

Warnings This utility is intended for use in network testing, measurement and management. It should be used primarily for manual fault isolation. Because of the load it could impose on the network, it is unwise to use `traceroute`(1M) during normal operations or from automated scripts.

Name trapstat – report trap statistics

Synopsis

```
/usr/sbin/trapstat [-t | -T | -e entry]
     [-C processor_set_id | -c cpulist] [-P] [-a]
     [-A cor|soc|bins] [-k keys] [-o num] [-m]
     [-I statfile] [-O statfile]
     [-r rate] [ [interval [count]] | command | [args]]

/usr/sbin/trapstat -l [-t | -T]
```

Description The `trapstat` utility gathers and displays run-time trap statistics on UltraSPARC-based systems. The default output is a table of trap types and CPU IDs, with each row of the table denoting a trap type and each column of the table denoting a CPU. If standard output is a terminal, the table contains as many columns of data as can fit within the terminal width; if standard output is not a terminal, the table contains at most six columns of data. By default, data is gathered and and displayed for all CPUs; if the data cannot fit in a single table, it is printed across multiple tables. The set of CPUs for which data is gathered and displayed can be optionally specified with the `-c` or `-C` option.

Unless the `-r` option or the `-a` option is specified, the value displayed in each entry of the table corresponds to the number of traps per second. If the `-r` option is specified, the value corresponds to the number of traps over the interval implied by the specified sampling rate; if the `-a` option is specified, the value corresponds to the accumulated number of traps since the invocation of `trapstat`.

By default, `trapstat` displays data once per second, and runs indefinitely; both of these behaviors can be optionally controlled with the *interval* and *count* parameters, respectively. The *interval* is specified in seconds; the *count* indicates the number of intervals to be executed before exiting. Alternatively, `command` can be specified, in which case `trapstat` executes the provided command and continues to run until the command exits. A positive integer is assumed to be an *interval*; if the desired *command* cannot be distinguished from an integer, the full path of *command* must be specified.

UltraSPARC I (obsolete), II, and III handle translation lookaside buffer (TLB) misses by trapping to the operating system. TLB miss traps can be a significant component of overall system performance for some workloads; the `-t` option provides in-depth information on these traps. When run with this option, `trapstat` displays both the rate of TLB miss traps and the percentage of time spent processing those traps. Additionally, TLB misses that hit in the translation storage buffer (TSB) are differentiated from TLB misses that further miss in the TSB. (The TSB is a software structure used as a translation entry cache to allow the TLB to be quickly filled; it is discussed in detail in the *UltraSPARC II User's Manual*.) The TLB and TSB miss information is further broken down into user- and kernel-mode misses.

Workloads with working sets that exceed the TLB reach may spend a significant amount of time missing in the TLB. To accommodate such workloads, the operating system supports multiple page sizes: larger page sizes increase the effective TLB reach and thereby reduce the number of TLB misses. To provide insight into the relationship between page size and TLB

miss rate, `trapstat` optionally provides in-depth TLB miss information broken down by page size using the `-T` option. The information provided by the `-T` option is a superset of that provided by the `-t` option; only one of `-t` and `-T` can be specified.

Options The following options are supported:

`-a`
Displays the number of traps as accumulating, monotonically increasing values instead of per-second or per-interval rates.

`-A cor`
Aggregate output by core ID. Data rows having the same core ID are aggregated into one row. The columns are replaced with subtotals, by default. The `-m` option prints column averages, instead.

`-A soc`
Aggregate output by socket ID. Data rows having the same socket ID are aggregated into one row. The columns are replaced with subtotals, by default. The `-m` option prints column averages, instead.

`-A bins`
Aggregate the columns into a lesser number of bins within each sampling period, grouping them in the order in which they appear. The `-m` option may be used in order to compute the arithmetic mean instead of the subtotal. The `-k` sorting option may be used to change the column order prior to the binning step.

Aggregation by ID (`-A cor|soc`) is processed before sorting (`-k`). Grouping by bins (`-A bins`) is done next. Finally, the number of output lines printed per interval may be limited by `-o`.

`-c` *cpulist*
Enables `trapstat` only on the CPUs specified by *cpulist*.

cpulist can be a single processor ID (for example, `4`), a range of processor IDs (for example, `4-6`), or a comma separated list of processor IDs or processor ID ranges (for example, `4,5,6` or `4,6-8`).

`-C` *processor_set_id*
Enables `trapstat` only on the CPUs in the processor set specified by *processor_set_id*.

`trapstat` modifies its output to always reflect the CPUs in the specified processor set. If a CPU is added to the set, `trapstat` modifies its output to include the added CPU; if a CPU is removed from the set, `trapstat` modifies its output to exclude the removed CPU. At most one processor set can be specified.

`-e` *entrylist*
Enables `trapstat` only for the trap table entry or entries specified by *entrylist*. A trap table entry can be specified by trap number or by trap name (for example, the level–10 trap can be specified as `74`, `0x4A`, `0x4a`, or `level-10`).

entrylist can be a single trap table entry or a comma separated list of trap table entries. If the specified trap table entry is not valid, `trapstat` prints a table of all valid trap table entries and values. A list of valid trap table entries is also found in *The SPARC Architecture Manual, Version 9* and the *Sun Microelectronics UltraSPARC II User's Manual*. If the parsable option (`-P`) is specified in addition to the `-e` option, the format of the data is as follows:

Field	Contents
1	Timestamp (nanoseconds since start)
2	CPU ID
3	Trap number (in hexadecimal)
4	Trap name
5	Trap rate per interval

Each field is separated with whitespace. If the format is modified, it will be modified by adding potentially new fields beginning with field 6; extant fields will remain unchanged.

`-I` *statfile*
Replay data previously saved in *statfile*. Create data files for replay by specifying `-O`. This option is especially useful for analyzing statistics on machines with large numbers of CPUs. The file may be reprocessed multiple times using different sorting and aggregation options.

The `-I` option is incompatible with an interval and count specification.

`-k` *key1*,...
Sort rows within each sampling period from highest to lowest by *key1*, then *key2*, and so on. Each key may be any of the row headers in the `trapstat` output, such as `level-10`, `u-itlb-miss`, and so forth.

Use `trapstat -l` to list all event names. Use `-lt` or `-lT` to list key names for the TLB formats.

`-l`
Lists trap table entries. By default, a table is displayed containing all valid trap numbers, their names and a brief description. The trap name is used in both the default output and in the *entrylist* parameter for the `-e` argument. If the parsable option (`-P`) is specified in addition to the `-l` option, the format of the data is as follows:

Field	Contents
1	Trap number in hexadecimal
2	Trap number in decimal

Field	Contents
3	Trap name
Remaining	Trap description

`-m`
Display the arithmetic mean value rather than the sum when the `-b` or `-i` is used to aggregate data over multiple CPUs.

`-o` *num*
Display only the first *num* rows within each sampling period, after applying sorting and aggregation options.

`-O` *statfile*
Save all data to *statfile*. This data may be replayed at a later time using `-I`.

Write to the standard output if the file name is – (hyphen).

The purpose of `-O` is to capture all available data. It is incompatible with the data reduction options: `-A`, `-k`, `-m`, and `-o`.

`-P`
Generates parsable output. When run without other data gathering modifying options (that is, `-e`, `-t` or `-T`), `trapstat`'s the parsable output has the following format:

Field	Contents
1	Timestamp (nanoseconds since start)
2	CPU ID
3	Trap number (in hexadecimal)
4	Trap name
5	Trap rate per interval

Each field is separated with whitespace. If the format is modified, it will be modified by adding potentially new fields beginning with field 6; extant fields will remain unchanged.

`-r` *rate*
Explicitly sets the sampling rate to be *rate* samples per second. If this option is specified, `trapstat`'s output changes from a traps-per-second to traps-per-sampling-interval.

`-t`
Enables TLB statistics.

A table is displayed with four principal columns of data: *itlb-miss*, *itsb-miss*, *dtlb-miss*, and *dtsb-miss*. The columns contain both the rate of the corresponding event and the percentage of CPU time spent processing the event. The percentage of CPU time is given only in terms of a single CPU. The rows of the table correspond to CPUs, with each CPU consuming two rows: one row for user-mode events (denoted with `u`) and one row for kernel-mode events (denoted with `k`). For each row, the percentage of CPU time is totalled and displayed in the rightmost column. The CPUs are delineated with a solid line. If the parsable option (`-P`) is specified in addition to the `-t` option, the format of the data is as follows:

Field	Contents
1	Timestamp (nanoseconds since start)
2	CPU ID
3	Mode (`k` denotes kernel, `u` denotes user.
4	I-TLB misses
5	Percentage of time in I-TLB miss handler
6	I-TSB misses
7	Percentage of time in I-TSB miss handler
8	D-TLB misses
9	Percentage of time in D-TLB miss handler
10	D-TSB misses
11	Percentage of time in D-TSB miss handler

Each field is separated with whitespace. If the format is modified, it will be modified by adding potentially new fields beginning with field 12; extant fields will remain unchanged.

`-T`

Enables TLB statistics, with page size information. As with the `-t` option, a table is displayed with four principal columns of data: *itlb-miss*, *itsb-miss*, *dtlb-miss*, and *dtsb-miss*. The columns contain both the absolute number of the corresponding event, and the percentage of CPU time spent processing the event. The percentage of CPU time is given only in terms of a single CPU. The rows of the table correspond to CPUs, with each CPU consuming two sets of rows: one set for user-level events (denoted with `u`) and one set for kernel-level events (denoted with `k`). Each set, in turn, contains as many rows as there are page sizes supported (see `getpagesizes(3C)`). For each row, the percentage of CPU time is totalled and displayed in the right-most column. The two sets are delineated with a dashed line; CPUs are delineated with a solid line. If the parsable option (`-P`) is specified in addition to the `-T` option, the format of the data is as follows:

Field	Contents
1	Timestamp (nanoseconds since start)
2	CPU ID
3	Mode k denotes kernel, u denotes user)
4	Page size, in decimal
5	I-TLB misses
6	Percentage of time in I-TLB miss handler
7	I-TSB misses
8	Percentage of time in I-TSB miss handler
9	D-TLB misses
10	Percentage of time in D-TLB miss handler
11	D-TSB misses
12	Percentage of time in D-TSB miss handler

Each field is separated with whitespace. If the format is modified, it will be modified by adding potentially new fields beginning with field 13; extant fields will remain unchanged.

Examples

EXAMPLE 1 Using `trapstat` Without Options

When run without options, `trapstat` displays a table of trap types and CPUs. At most six columns can fit in the default terminal width; if (as in this example) there are more than six CPUs, multiple tables are displayed:

```
example# trapstat
vct  name                |     cpu0     cpu1     cpu4     cpu5     cpu8     cpu9
-------------------------+-----------------------------------------------------
 24 cleanwin             |     6446     4837     6368     2153     2623     1321
 41 level-1              |      100        0        0        0        1        0
 44 level-4              |        0        1        1        1        0        0
 45 level-5              |        0        0        0        0        0        0
 47 level-7              |        0        0        0        0        9        0
 49 level-9              |      100      100      100      100      100      100
 4a level-10             |      100        0        0        0        0        0
 4d level-13             |        6       10        7       16       13       11
 4e level-14             |      100        0        0        0        1        0
 60 int-vec              |     2607     2740     2642     2922     2920     3033
 64 itlb-miss            |     3129     2475     3167     1037     1200      569
 68 dtlb-miss            |   121061    86162   109838    37386    45639    20269
 6c dtlb-prot            |      997      847     1061      379      406      184
 84 spill-user-32        |     2809     2133     2739   200806   332776   454504
 88 spill-user-64        |    45819   207856    93487   228529    68373    77590
```

EXAMPLE 1 Using `trapstat` Without Options *(Continued)*

```
 8c spill-user-32-cln   |      784      561      767      274      353      215
 90 spill-user-64-cln   |        9       37       17       39       12       13
 98 spill-kern-64       |    62913    50145    63869    21916    28431    11738
 a4 spill-asuser-32     |     1327      947     1288      460      572      335
 a8 spill-asuser-64     |       26       48       18       54       10       14
 ac spill-asuser-32-cln |     4580     3599     4555     1538     1978      857
 b0 spill-asuser-64-cln |       26        0        0        2        0        0
 c4 fill-user-32        |     2862     2161     2798   191746   318115   435850
 c8 fill-user-64        |    45813   197781    89179   217668    63905    74281
 cc fill-user-32-cln    |     3802     2833     3733    10153    16419    19475
 d0 fill-user-64-cln    |      329    10105     4873    10603     4235     3649
 d8 fill-kern-64        |    62519    49943    63611    21824    28328    11693
108 syscall-32          |     2285     1634     2278      737      957      383
126 self-xcall          |      100        0        0        0        0        0

vct  name               |    cpu12    cpu13    cpu14    cpu15
------------------------+-----------------------------------
 24 cleanwin            |     5435     4232     6302     6104
 41 level-1             |        0        0        0        0
 44 level-4             |        2        0        0        1
 45 level-5             |        0        0        0        0
 47 level-7             |        0        0        0        0
 49 level-9             |      100      100      100      100
 4a level-10            |        0        0        0        0
 4d level-13            |       15       11       22       11
 4e level-14            |        0        0        0        0
 60 int-vec             |     2813     2833     2738     2714
 64 itlb-miss           |     2636     1925     3133     3029
 68 dtlb-miss           |    90528    70639   107786   103425
 6c dtlb-prot           |      819      675      988      954
 84 spill-user-32       |   175768    39933     2811     2742
 88 spill-user-64       |        0   241348    96907   118298
 8c spill-user-32-cln   |      681      513      753      730
 90 spill-user-64-cln   |        0       42       16       20
 98 spill-kern-64       |    52158    40914    62305    60141
 a4 spill-asuser-32     |     1113      856     1251     1208
 a8 spill-asuser-64     |        0       64       16       24
 ac spill-asuser-32-cln |     3816     2942     4515     4381
 b0 spill-asuser-64-cln |        0        0        0        0
 c4 fill-user-32        |   170744    38444     2876     2784
 c8 fill-user-64        |        0   230381    92941   111694
 cc fill-user-32-cln    |     8550     3790     3612     3553
 d0 fill-user-64-cln    |        0    10726     4495     5845
 d8 fill-kern-64        |    51968    40760    62053    59922
108 syscall-32          |     1839     1495     2144     2083
126 self-xcall          |        0        0        0        0
```

EXAMPLE 2 Using `trapset` with CPU Filtering

The `-c` option can be used to limit the CPUs on which `trapstat` is enabled. This example limits CPU 1 and CPUs 12 through 15.

```
example# trapstat -c 1,12-15


vct  name                |     cpu1    cpu12    cpu13    cpu14    cpu15
-------------------------+-----------------------------------------
 24 cleanwin             |     6923     3072     2500     3518     2261
 44 level-4              |        3        0        0        1        1
 49 level-9              |      100      100      100      100      100
 4d level-13             |       23        8       14       19       14
 60 int-vec              |     2559     2699     2752     2688     2792
 64 itlb-miss            |     3296     1548     1174     1698     1087
 68 dtlb-miss            |   114788    54313    43040    58336    38057
 6c dtlb-prot            |     1046      549      417      545      370
 84 spill-user-32        |    66551    29480   301588    26522   213032
 88 spill-user-64        |        0   318652   111239   299829   221716
 8c spill-user-32-cln    |      856      347      331      416      293
 90 spill-user-64-cln    |        0       55       21       59       39
 98 spill-kern-64        |    66464    31803    24758    34004    22277
 a4 spill-asuser-32      |     1423      569      560      698      483
 a8 spill-asuser-64      |        0       74       32       98       46
 ac spill-asuser-32-cln  |     4875     2250     1728     2384     1584
 b0 spill-asuser-64-cln  |        0        2        0        1        0
 c4 fill-user-32         |    64193    28418   287516    27055   202093
 c8 fill-user-64         |        0   305016   106692   288542   210654
 cc fill-user-32-cln     |     6733     3520    15185     2396    12035
 d0 fill-user-64-cln     |        0    13226     3506    12933    11032
 d8 fill-kern-64         |    66220    31680    24674    33892    22196
108 syscall-32           |     2446      967      817     1196      755
```

EXAMPLE 3 Using `trapstat` with TLB Statistics

The `-t` option displays in-depth TLB statistics, including the amount of time spent performing TLB miss processing. The following example shows that the machine is spending 14.1 percent of its time just handling D-TLB misses:

```
example# trapstat -t
cpu m| itlb-miss %tim itsb-miss %tim | dtlb-miss %tim dtsb-miss %tim |%tim
-----+-------------------------------+-------------------------------+----
  0 u|      2571  0.3         0  0.0 |     10802  1.3         0  0.0 | 1.6
  0 k|         0  0.0         0  0.0 |    106420 13.4       184  0.1 |13.6
-----+-------------------------------+-------------------------------+----
  1 u|      3069  0.3         0  0.0 |     10983  1.2       100  0.0 | 1.6
  1 k|        27  0.0         0  0.0 |    106974 12.6        19  0.0 |12.7
-----+-------------------------------+-------------------------------+----
  2 u|      3033  0.3         0  0.0 |     11045  1.2       105  0.0 | 1.6
```

EXAMPLE 3 Using `trapstat` with TLB Statistics *(Continued)*

```
  2 k|        43  0.0         0  0.0 |    107842 12.7       108  0.0 |12.8
-----+------------------------------+------------------------------+----
  3 u|      2924  0.3         0  0.0 |     10380  1.2       121  0.0 | 1.6
  3 k|        54  0.0         0  0.0 |    102682 12.2        16  0.0 |12.2
-----+------------------------------+------------------------------+----
  4 u|      3064  0.3         0  0.0 |     10832  1.2       120  0.0 | 1.6
  4 k|        31  0.0         0  0.0 |    107977 13.0       236  0.1 |13.1
=====+==============================+==============================+====
 ttl |     14816  0.3         0  0.0 |    585937 14.1      1009  0.0 |14.5
```

EXAMPLE 4 Using `trapstat` with TLB Statistics and Page Size Information

By specifying the `-T` option, `trapstat` shows TLB misses broken down by page size. In this example, CPU 0 is spending 7.9 percent of its time handling user-mode TLB misses on 8K pages, and another 2.3 percent of its time handling user-mode TLB misses on 64K pages.

```
example# trapstat -T -c 0
cpu m size| itlb-miss %tim itsb-miss %tim | dtlb-miss %tim dtsb-miss %tim |%tim
----------+------------------------------+------------------------------+----
  0 u   8k|      1300  0.1        15  0.0 |    104897  7.9        90  0.0 | 8.0
  0 u  64k|         0  0.0         0  0.0 |     29935  2.3         7  0.0 | 2.3
  0 u 512k|         0  0.0         0  0.0 |      3569  0.2         2  0.0 | 0.2
  0 u   4m|         0  0.0         0  0.0 |       233  0.0         2  0.0 | 0.0
- - - - - + - - - - - - - - - - - - - - - + - - - - - - - - - - - - - - - + - -
  0 k   8k|        13  0.0         0  0.0 |     71733  6.5       110  0.0 | 6.5
  0 k  64k|         0  0.0         0  0.0 |         0  0.0         0  0.0 | 0.0
  0 k 512k|         0  0.0         0  0.0 |         0  0.0       206  0.1 | 0.1
  0 k   4m|         0  0.0         0  0.0 |         0  0.0         0  0.0 | 0.0
==========+==============================+==============================+====
      ttl |      1313  0.1        15  0.0 |    210367 17.1       417  0.2 |17.5
```

EXAMPLE 5 Using `trapstat` with Entry Filtering

By specifying the `-e` option, `trapstat` displays statistics for only specific trap types. Using this option minimizes the probe effect when seeking specific data. This example yields statistics for only the *dtlb-prot* and *syscall-32* traps on CPUs 12 through 15:

```
example# trapstat -e dtlb-prot,syscall-32 -c 12-15
vct  name                |    cpu12    cpu13    cpu14    cpu15
-------------------------+-----------------------------------
 6c dtlb-prot            |      817      754     1018      560
108 syscall-32           |     1426     1647     2186     1142

vct  name                |    cpu12    cpu13    cpu14    cpu15
-------------------------+-----------------------------------
 6c dtlb-prot            |     1085      996      800      707
108 syscall-32           |     2578     2167     1638     1452
```

EXAMPLE 6 Using `trapstat` with a Higher Sampling Rate

The following example uses the `-r` option to specify a sampling rate of 1000 samples per second, and filter only for the level-10 trap. Additionally, specifying the `-P` option yields parsable output.

Notice the timestamp difference between the level-10 events: 9,998,000 nanoseconds and 10,007,000 nanoseconds. These level-10 events correspond to the system clock, which by default ticks at 100 hertz (that is, every 10,000,000 nanoseconds).

```
example# trapstat -e level-10 -P -r 1000
1070400 0 4a level-10 0
2048600 0 4a level-10 0
3030400 0 4a level-10 1
4035800 0 4a level-10 0
5027200 0 4a level-10 0
6027200 0 4a level-10 0
7027400 0 4a level-10 0
8028200 0 4a level-10 0
9026400 0 4a level-10 0
10029600 0 4a level-10 0
11028600 0 4a level-10 0
12024000 0 4a level-10 0
13028400 0 4a level-10 1
14031200 0 4a level-10 0
15027200 0 4a level-10 0
16027600 0 4a level-10 0
17025000 0 4a level-10 0
18026000 0 4a level-10 0
19027800 0 4a level-10 0
20025600 0 4a level-10 0
21025200 0 4a level-10 0
22025000 0 4a level-10 0
23035400 0 4a level-10 1
24027400 0 4a level-10 0
25026000 0 4a level-10 0
26027000 0 4a level-10 0
```

EXAMPLE 7 Display Three CPUs with Highest cpu_mondo Rate

The following command displays the three CPUs with the highest `cpu_mondo` rate.

```
example% trapstat -k cpu_mondo -o 3 10 1
vct name                | cpu0 cpu1 cpu61
------------------------+----------------
9   immu-miss           |    0    0     0
24  cleanwin            |    0    0     0
31  dmmu-miss           |    0    0     0
41  level-1             |    0    0     0
46  level-6             |    0    0     0
```

EXAMPLE 7 Display Three CPUs with Highest cpu_mondo Rate *(Continued)*

```
49  level-9             |    0    0     0
4a  level-10            |  100   31    16
4d  level-13            |   23   15     8
4e  level-14            |  100   32    18
6c  dtlb-prot           |    0    0     0
7c  cpu_mondo           |   24   16     9
7d  dev_mondo           |    0    0     0
84  spill-user-32       |    0    0     0
8c  spill-user-32-cln   |    0    0     0
98  spill-kern-64       |  423  180   102
a4  spill-asuser-32     |    0    0     0
ac  spill-asuser-32-cln |    0    0     0
c4  fill-user-32        |    0    0     0
cc  fill-user-32-cln    |    0    1     0
d8  fill-kern-64        |  295  165    94
103 flush-wins          |    0    0     0
108 syscall-32          |    0    0     0
122 get-psr             |    0    0     0
127 gethrtime           |    0    0     0
```

EXAMPLE 8 Aggregating Multiple CPUs into Quartiles

The following commands aggregate 96 CPUs into quartiles by level-10 rate.

```
example% trapstat -O /tmp/t1 -e level-10 10 1
example% trapstat -I /tmp/t1 -A 4
vct name     | bin0 bin1 bin2 bin3
-------------+--------------------
4a  level-10 |  440  340  305  306
```

EXAMPLE 9 Aggregating and Sorting Multiple CPUs

The following command aggregates 96 CPUs by core ID and sorts for the highest four.

```
example% trapstat -A cor -e level-10 -k level-10 -o 4 10 1
vct name     |  cor514  cor549  cor542  cor521
-------------+--------------------------------
4a  level-10 |     197     120     111     106
```

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os
Interface Stability Human Readable Output	 Uncommitted

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Parsable Output	Committed

See Also `lockstat(1M)`, `pmap(1)`, `psrset(1M)`, `psrinfo(1M)`, `pbind(1M)`, `ppgsz(1)`, `getpagesizes(3C)`

Sun Microelectronics UltraSPARC II User's Manual, January 1997, STP1031,

The SPARC Architecture Manual, Version 9, 1994, Prentice-Hall.

Notes When enabled, `trapstat` induces a varying probe effect, depending on the type of information collected. While the precise probe effect depends upon the specifics of the hardware, the following table can be used as a rough guide:

Option	Approximate probe effect
default	3-5% per trap
`-e`	3-5% per specified trap
`-t`, `-T`	40-45% per TLB miss trap hitting in the TSB, 25-30% per TLB miss trap missing in the TSB

These probe effects are *per trap* not for the system as a whole. For example, running `trapstat` with the default options on a system that spends 7% of total time handling traps induces a performance degradation of less than one half of one percent; running `trapstat` with the `-t` or `-T` option on a system spending 5% of total time processing TLB misses induce a performance degradation of no more than 2.5%.

When run with the `-t` or `-T` option, `trapstat` accounts for its probe effect when calculating the *%tim* fields. This assures that the *%tim* fields are a reasonably accurate indicator of the time a given workload is spending handling TLB misses — regardless of the perturbing presence of `trapstat`.

While the *%tim* fields include the explicit cost of executing the TLB miss handler, they do *not* include the implicit costs of TLB miss traps (for example, pipeline effects, cache pollution, etc). These implicit costs become more significant as the trap rate grows; if high *%tim* values are reported (greater than 50%), you can accurately infer that much of the balance of time is being spent on the implicit costs of the TLB miss traps.

Due to the potential system wide degradation induced, only the super-user can run `trapstat`.

Due to the limitation of the underlying statistics gathering methodology, only one instance of `trapstat` can run at a time.

Name ttymon – port monitor for terminal ports

Synopsis

```
/usr/sbin/ttymon -g [-d device] [-h] [-t timeout]
     [-l ttylabel] [-p prompt] [-m modules] [-T termtype]
```

Description `ttymon` is a STREAMS-based TTY port monitor. Its function is to monitor ports, to set terminal modes, baud rates, and line disciplines for the ports, and to connect users or applications to services associated with the ports. Each instance of `ttymon` monitors one port, specified at startup. When an instance of `ttymon` is started, `ttymon` first initializes the line disciplines, if they are specified, and the speed and terminal settings. For ports with entries in `/etc/logindevperm`, device owner, group and permissions are set. (See `logindevperm(4)`.) The values used for initialization are taken from the appropriate entry in the TTY settings file. This file is maintained by the `sttydefs(1M)` command. Default line disciplines on ports are usually set up by the `autopush(1M)` command of the Autopush Facility.

`ttymon` then writes the prompt and waits for user input. If the user indicates that the speed is inappropriate by pressing the BREAK key, `ttymon` tries the next speed and writes the prompt again. When valid input is received, `ttymon` creates a `utmpx` entry (see `utmpx(4)`), and execs the login service for the port. Valid input consists of a string of at least one non-newline character, terminated by a carriage return.

If *autobaud* is enabled for a port, `ttymon` will try to determine the baud rate on the port automatically. Users must enter a carriage return before `ttymon` can recognize the baud rate and print the prompt. Currently, the baud rates that can be determined by *autobaud* are `110`, `1200`, `2400`, `4800`, and `9600`.

SMF Service Description The primary `smf(5)` service which invokes `ttymon` is `svc:/system/console-login`, which may have multiple service instances. Instances are described in greater detail below. The service provides a number of properties within the property group `ttymon` to control the invocation, as follows:

NAME	TYPE	TTYMON OPTION
device	astring	[-d device]
nohangup	boolean	[-h]
label	astring	[-l label]
modules	astring	[-m module1,module2]
prompt	astring	[-p prompt]
timeout	count	[-t timeout]
terminal_type	astring	[-T termtype]

If any value is the empty string or an integer set to zero, then the option is not passed to the `ttymon` invocation.

`svc:/system/console-login:default`

The default instance always represents the `ttymon` that offers login on the system hardware console.

See EXAMPLES for an example of how to modify settings for the system console.

`svc:/system/console-login:{vt2, vt3, vt4, vt5, vt6}`
Additional service instances are provided for the system's virtual consoles. If virtual consoles are not available, these services will automatically disable themselves. See `vtdaemon(1M)`.

`svc:/system/console-login:{terma, termb}`
`svc:/system/console-login:terma` and `svc:/system/console-login:termb` are provided as a convenience and can assist the user in setting up login services for additional ports `/dev/term/a` and `/dev/term/b`. These services are disabled by default.

Creating Additional Instances

The user can configure additional service instances for additional devices. This can be accomplished in any of these ways:

- Manually creating the service instance using `svccfg(1M)`.
- Creating the service in a service profile (see `smf(5)`).
- Creating a service manifest for additional service instance(s).

See EXAMPLES for an example of manually configuring the service using `svccfg`.

SMF Service Errors

In most cases when an instance of the console-login service is misconfigured, it will transition itself to the maintenance state. Use `svcs -l` (see `svcs(1)`) to determine the location of the service's log file and consult the log for additional information.

In some error cases, the service may respawn indefinitely. Disable the service using `svcadm(1M)`, then consult the service log for additional messages or information to help resolve the problem.

Security

`ttymon` uses `pam(3PAM)` for session management. The PAM configuration policy, specified in `/etc/pam.conf` or per-service files in `/etc/pam.d/`, specifies the modules to be used for `ttymon`. Here is a partial `pam.conf` file with an entry for `ttymon` using the UNIX session management module:

```
ttymon session required /usr/lib/security/pam_unix_session.so.1
```

The equivalent PAM configuration using `/etc/pam.d/` would be the following entry in `/etc/pam.d/ttymon`:

```
session required /usr/lib/security/pam_unix_session.so.1
```

If there are no entries for the `ttymon` service in `/etc/pam.conf` and the `/etc/pam.d/ttymon` file does not exist, then the entries for the "other" service in `/etc/pam.conf` will be used. If there are not any entries in `/etc/pam.conf` for the "other" service, then the entries in `/etc/pam.d/other` will be used.

Options

The following options are supported:

`-g` The `-g` option is required for historical reasons.

`-d`*device*	*device* is the full path name of the port to which `ttymon` is to attach. If this option is not specified, file descriptor `0` must be set up by the invoking process to a TTY port.
`-h`	If the -h flag is not set, `ttymon` will force a hangup on the line by setting the speed to zero before setting the speed to the default or specified speed.
`-l`*ttylabel*	*ttylabel* is a link to a speed and TTY definition in the `ttydefs` file. This definition tells `ttymon` at what speed to run initially, what the initial TTY settings are, and what speed to try next if the user indicates that the speed is inappropriate by pressing the BREAK key. The default speed is 9600 baud.
`-m`*modules*	When initializing the port, `ttymon` will pop all modules on the port, and then push *modules* in the order specified. *modules* is a comma-separated list of pushable modules. Default modules on the ports are usually set up by the Autopush Facility.
`-p`*prompt*	Allows the user to specify a prompt string. The default prompt is `Login:`.
`-t`*timeout*	Specifies that `ttymon` should exit if no one types anything in *timeout* seconds after the prompt is sent.
`-T`*termtype*	Sets the `TERM` environment variable to *termtype*.
`-v`	Enables verbose messaging.

Examples

EXAMPLE 1 Setting the Terminal Type for the System Console

The following example sets the value of the terminal type (`-T`) option for the system console `ttymon` invocation:

```
# svccfg -s svc:/system/console-login:default \
       "setprop ttymon/terminal_type = xterm"
# svcadm refresh svc:/system/console-login:default
```

EXAMPLE 2 Creating a Service Instance for an Additional Serial Device

In this example, the user wishes to configure an additional instance of the `svc:/system/console-login` service in order to offer login services over a terminal connected by means of a USB serial adapter. Assume that the USB serial port is present as `/dev/term/1`, and the user plans to connect a `vt100` terminal to it. In this case, the service instance can be named `term1` (or any other name) and defined as follows:

```
# svccfg -s svc:/system/console-login "add term1"
# SVC=svc:/system/console-login:term1
# svccfg -s $SVC "addpg ttymon application"
# svccfg -s $SVC "setprop ttymon/device = /dev/term/1"
# svccfg -s $SVC "setprop ttymon/terminal_type = vt100"
# svcadm refresh $SVC
# svcadm enable $SVC
```

Environment Variables

If any of the LC_* variables (LC_CTYPE, LC_MESSAGES, LC_TIME, LC_COLLATE, LC_NUMERIC, and LC_MONETARY) (see environ(5)) are not set in the environment, the operational behavior of ttymon for each corresponding locale category is determined by the value of the LANG environment variable. If LC_ALL is set, its contents are used to override both the LANG and the other LC_* variables. If none of the above variables is set in the environment, the "C" (U.S. style) locale determines how ttymon behaves.

LC_CTYPE — Determines how ttymon handles characters. When LC_CTYPE is set to a valid value, ttymon can display and handle text and filenames containing valid characters for that locale. ttymon can display and handle Extended Unix Code (EUC) characters where any individual character can be 1, 2, or 3 bytes wide. ttymon can also handle EUC characters of 1, 2, or more column widths. In the "C" locale, only characters from ISO 8859-1 are valid.

Files

/etc/logindevperm
Contains information that is used by login(1) and ttymon to change the owner, group, and permissions of devices upon logging into or out of a console device.

Attributes

See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os
Interface Stability	Committed

See Also

svcs(1), ct(1C), cu(1C), autopush(1M), sttydefs(1M), svcadm(1M), svccfg(1M), uucico(1M), vtdaemon(1M), pam(3PAM), logindevperm(4), pam.conf(4), utmpx(4), attributes(5), environ(5), pam_authtok_check(5), pam_authtok_get(5), pam_authtok_store(5), pam_dhkeys(5), pam_passwd_auth(5), pam_unix_account(5), pam_unix_auth(5), pam_unix_session(5), smf(5)

Oracle Solaris Administration: Common Tasks

Notes

Service Access Facility (SAF and SAC)

ttymon was formerly a component of the Service Access Facility and was invoked by sac, the Service Access Controller. This facility has been removed in this release of Solaris, and a conversion to SMF of relevant portions was performed.

Competition for Ports

If a port is monitored by more than one ttymon, it is possible for the ttymons to send out prompt messages in such a way that they compete for input.

It is possible that two svc:/system/console-login service instances could refer to the same underlying device. For example, if the system's hardware console is connected (due to settings or autodetection in firmware) to serial port A, then both the svc:/system/console-login:default and svc:/system/console-login:terma services

will refer to same underlying hardware device. Care should be taken when defining or enabling additional service instances to avoid this situation, or the two `ttymon`s will compete for input.

Name tunefs – tune an existing UFS file system

Synopsis

```
tunefs [-a maxcontig] [-d rotdelay] [-e maxbpg]
     [-m minfree] [-o space | time] special | filesystem
```

Description `tunefs` is designed to change the dynamic parameters of a file system that affect the layout policies. When using `tunefs` with *filesystem*, *filesystem* must be in `/etc/vfstab`. The parameters that can be changed are indicated by the options given below.

Options The following options are supported:

`-a` *maxcontig* The maximum number of logical blocks, belonging to one file, that is allocated contiguously. The default is calculated as follows:

`maxcontig =` *disk drive maximum transfer size / disk block size*

If the disk drive's maximum transfer size cannot be determined, the default value for `maxcontig` is calculated from kernel parameters as follows:

If `maxphys` is less than `ufs_maxmaxphys`, which is 1 Mbyte, then `maxcontig` is set to `maxphys`. Otherwise, `maxcontig` is set to `ufs_maxmaxphys`.

You can set `maxcontig` to any positive integer value.

The actual value will be the lesser of what has been specified and what the hardware supports.

`-d` *rotdelay* This parameter is obsolete as of the Solaris 10 release. The value is always set to `0`, regardless of the input value.

`-e` *maxbpg* Indicates the maximum number of contiguous logical blocks any single file can allocate from a cylinder group before it is forced to begin allocating blocks from another cylinder group. Typically this value is set to approximately one quarter of the total contiguous logical blocks in a cylinder group. The intent is to prevent any single file from using up all the blocks in a single cylinder group, thus degrading access times for all files subsequently allocated in that cylinder group.

The effect of this limit is to cause big files to do long seeks more frequently than if they were allowed to allocate all the blocks in a cylinder group before seeking elsewhere. For file systems with exclusively large files, this parameter should be set higher.

`-m` *minfree* Specifies the minimum free space threshold, or the percentage of space held back from normal users. This value can be set to `0`. However, up to a factor of three in throughput will be lost over the performance obtained at a 10% threshold. *Note:* If the value is raised above the current usage level, users will be unable to allocate files until enough files have been deleted to get under the higher threshold.

`-o` *space* | *time* The file system can either be instructed to try to minimize the *time* spent allocating blocks, or to try to minimize the *space* fragmentation on the disk. The default is *time*.

Generally, you should optimize for time unless the file system is over 90% full.

Usage See `largefile(5)` for the description of the behavior of `tunefs` when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also `mkfs_ufs(1M)`, `newfs(1M)`, `attributes(5)`, `largefile(5)`

Name txzonemgr – Trusted Extensions Zone Manager configuration utility

Synopsis

```
/usr/sbin/txzonemgr [-c | -d[f] | -Z file]
```

Description The `txzonemgr` shell script provides a simple, menu-based GUI wizard for creating, installing, initializing, and booting labeled zones on a system on which Trusted Extensions is enabled. The script also provides menu items for networking options, name services options, and making the global zone a client of an existing LDAP server. By default, all zones are configured to use the same name service and IP address as the global zone.

`txzonemgr` can also perform a limited set of commands in command-line mode instead of GUI mode, by using appropriate options, described below.

`txzonemgr` is run by roles granted in the Zone Management Rights Profile, or by root in the global zone.

Options Without options, `txzonemgr` operates as a menu-based GUI. Using any options will result in command-line operation only. Except as specified below, options cannot be combined.

The following options are supported:

`-c`
: Create default zones. Requires that no zones already exist. The number, names, and attributes of the zones created is subject to change. However, typically a "public" zone and at least one other zone is created.

`-d`
: Destroy all zones. All zones will be halted, uninstalled, and deleted. Prompts for confirmation unless the `-f` option is also specified.

`-f`
: Force. Can be used with `-d` to override prompt for confirmation.

`-Z` *file*
: Create a set of zones using the list specified in the file. Each line in the file must contain a zone name and its corresponding label, separated by whitespace. The label should not be quoted even if it contains whitespace. If an existing zone named "snapshot" exists, all the specified zones are cloned from it. Otherwise the first zone in the list is installed, and then a snapshot zone is cloned from it. The remaining zones are then cloned from the snapshot. The zones are configured to require manual booting. However, when an authorized user assigns a valid label to a GNOME workspace, the corresponding zone is booted automatically.

Exit Status No values are returned for GUI mode. For command-line operation, the following exit values are returned:

`0`
: Successful completion.

1
An error occurred.

2
Invalid usage.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Command Interface Stability	Committed
Interactive Dialogue	Not an Interface

See Also zenity(1) (not a SunOS man page), ipadm(1M), zoneadm(1M), zonecfg(1M), attributes(5), rbac(5), zones(5)

Solaris Trusted Extensions Administrator's Procedures

Notes If administering zones from JDS, use txzonemgr rather than CDE actions.

txzonemgr uses the zenity command. For details, see the zenity(1) man page, which is not part of the SunOS collection.

Name tzreload – notify timezone update

Synopsis

```
/usr/sbin/tzreload
```

Description The `tzreload` command notifies active (running) processes to reread timezone information. The timezone information is cached in each process, absent a `tzreload` command, is never reread until a process is restarted. In response to a `tzreload` command, active processes reread the current timezone information at the next call to `ctime(3C)` and `mktime(3C)`. The `tzreload` notification is sent to processes within the current zone.

`tzreload` causes processes which are using the system timezone (in `/etc/default/init`) to reread the contents of that file.

In addition to notifying active processes, the `tzreload` command also notifies `cron(1M)`, to reinitialize the job scheduler with the new timezone information.

Files

`/usr/share/lib/zoneinfo`
Standard zone information directory.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os
Interface Stability	Committed

See Also `cron(1M)`, `zdump(1M)`, `zic(1M)`, `ctime(3C)`, `mktime(3C)`, `attributes(5)`, `smf(5)`

Notes Although `tzreload` reinitializes `cron(1M)`, applications that are affected by timezone changes still need to be restarted or reinitialized if they do not reread the new timezone information before timezone changes take place.

The `timezone` update service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/system/timezone:default
```

Administrative actions on this service, such as enabling, disabling, or requesting refresh, can be performed using `svcadm(1M)`. The service's status can be queried using the `svcs(1)` command.

Refreshing this service causes the `tzreload` command to be run, notifying running processes to reload their `timezone` caches. Disabling this service is not recommended.

Name tzselect – select a time zone

Synopsis

```
/usr/bin/tzselect
```

Description The `tzselect` program asks you a series of questions about the current location and outputs the resulting time zone description to standard output. The output is suitable as a value for the `TZ` environment variable.

All user interaction is through standard input and standard error.

Options The `tzselect` command has no options.

Exit Status The following exit values are returned:

`0` Timezone information was successfully obtained.

`>0` An error occurred.

Files `/usr/share/lib/zoneinfo` directory containing timezone data files

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os
Interface Stability	Committed

See Also `zdump(1M)`, `zic(1M)`, `ctime(3C)`, `attributes(5)`

Name uadmin – administrative control

Synopsis `/usr/sbin/uadmin` *cmd fcn* [*mdep*]

Description The `uadmin` command provides control for basic administrative functions. This command is tightly coupled to the system administration procedures and is not intended for general use. It may be invoked only by the super-user.

Both the *cmd* (command) and *fcn* (function) arguments are converted to integers and passed to the `uadmin` system call. The optional *mdep* (machine dependent) argument is only available for the *cmd* values of `1` (`A_REBOOT`), `2` (`A_SHUTDOWN`), or `5` (`A_DUMP`). For any other *cmd* value, no *mdep* command-line argument is allowed.

When passing an *mdep* value that contains whitespaces, the string must be grouped together as a single argument enclosed within quotes, for example:

```
uadmin 1 1 “-s kernel/unix”
```

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also `halt(1M)`, `reboot(1M)`, `uadmin(2)`, `attributes(5)`

Warnings Shutting down or halting the system by means of `uadmin` does not update the boot archive. Avoid using this command after

- editing of files such as `/etc/system`
- installing new driver binaries or kernel binaries
- updating existing driver binaries or kernel binaries.

Use `reboot(1M)` or `halt(1M)` instead.

Name ucodeadm – update processor microcode

Synopsis

```
/usr/sbin/ucodeadm -v

/usr/sbin/ucodeadm -umicrocode-text-file

/usr/sbin/ucodeadm -i [-R path] microcode-text-file
```

Description The `ucodeadm` utility can be used to report running microcode revision on the processors, update microcode, or install microcode on the target system to be used during the boot process.

The *microcode-text-file* can be obtained from processor vendors.

Options

`-v` Report microcode revision.

`-u` *microcode-text-file* Update microcode on all cross-call interrupt ready processors.

`-i` *microcode-text-file* Install microcode files on target system to be used during the next boot cycle. The text file name must have the vendor name prefix, such as "`intel`" or "`amd`".

By default the microcode files will be installed at:

```
/platform/i86pc/ucode/$VENDORSTR/
```

where `VENDORSTR` is either "`GenuineIntel`" or "`AuthenticAMD`".

`-R` *alternate path* Install *microcode* path in the *alternate path*.

Examples

EXAMPLE 1 Reporting the Microcode Revision

The following example displays the microcode revision that is currently running:

```
# ucodeadm -v
```

EXAMPLE 2 Updating the Processor Microcode

The following example updates the processor microcode to `intel-ucode.txt`:

```
# ucodeadm -u intel-ucode.txt
```

EXAMPLE 3 Installing the Microcode on the Target System

The following example installs the microcode on the target system, `/export/ucode-path`:

```
# ucodeadm -i -R /export/ucode-path intel-ucode.txt
```

If an alternate path is used when installing the microcode on the target system, the installed microcode file is not used on the next boot cycle.

Exit Status The following exit values are returned:

0 Successful completion.

>0 An error occurred.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os
Interface Stability	Committed

See Also psradm(1M), psrinfo(1M), attributes(5)

Name ufsdump – incremental file system dump

Synopsis `/usr/sbin/ufsdump [`*options*`] [`*arguments*`]` *files_to_dump*

Description `ufsdump` backs up all files specified by *files_to_dump* (usually either a whole file system or files within a file sytem changed after a certain date) to magnetic tape or disk file.

The `ufsdump` command can only be used on unmounted file systems, or those mounted read-only. Attempting to dump a mounted, read-write file system might result in a system disruption or the inability to restore files from the dump. Consider using the `fssnap(1M)` command to create a file system snapshot if you need a point-in-time image of a file system that is mounted.

options is a single string of one-letter `ufsdump` options.

arguments may be multiple strings whose association with the options is determined by order. That is, the first argument goes with the first option that takes an argument; the second argument goes with the second option that takes an argument, and so on.

files_to_dump is required and must be the last argument on the command line. See `OPERANDS` for more information.

With most devices `ufsdump` can automatically detect the end-of-media. Consequently, the `d`, `s`, and `t` options are not necessary for multi-volume dumps, unless `ufsdump` does not understand the way the device detects the end-of-media, or the files are to be restored on a system with an older version of the `restore` command.

Options The following options are supported:

`0`–`9`
The "dump level." All files specified by *files_to_dump* that have been modified since the last `ufsdump` at a lower dump level are copied to the *dump_file* destination (normally a magnetic tape device). For instance, if a "level `2`" dump was done on Monday, followed by a "level `4`" dump on Tuesday, a subsequent "level `3`" dump on Wednesday would contain all files modified or added since the "level `2`" (Monday) backup. A "level `0`" dump copies the entire file system to the *dump_file*.

`a` *archive_file*
Archive file. Archive a dump table-of-contents in the specified *archive_file* to be used by `ufsrestore(1M)` to determine whether a file is in the dump file that is being restored.

`b` *factor*
Blocking factor. Specify the blocking factor for tape writes. The default is 20 blocks per write for tapes of density less than 6250BPI (bytes-per-inch). The default blocking factor for tapes of density 6250BPI and greater is 64. The default blocking factor for cartridge tapes (`c` option) is 126. The highest blocking factor available with most tape drives is 126. Note: the blocking factor is specified in terms of 512-byte blocks, for compatibility with `tar(1)`.

`c`

Cartridge. Set the defaults for cartridge instead of the standard half-inch reel. This sets the density to 1000BPI and the blocking factor to 126. Since `ufsdump` can automatically detect the end-of-media, only the blocking parameter normally has an effect. When cartridge tapes are used, and this option is *not* specified, `ufsdump` will slightly miscompute the size of the tape. If the `b`, `d`, `s` or `t` options are specified with this option, their values will override the defaults set by this option.

`d` *bpi*

Tape density. Not normally required, as `ufsdump` can detect end-of-media. This parameter can be used to keep a running tab on the amount of tape used per reel. The default density is 6250BPI except when the `c` option is used for cartridge tape, in which case it is assumed to be 1000BPI per track. Typical values for tape devices are:

1/2 inch tape
6250 BPI

1/4 inch cartridge
1000 BPI The tape densities and other options are documented in the `st`(7D) man page.

`D`

Diskette. Obsolete option.

`f` *dump_file*

Dump file. Use *dump_file* as the file to dump to, instead of `/dev/rmt/0`. If *dump_file* is specified as –, dump to standard output.

If the name of the file is of the form *machine*:*device,* the dump is done from the specified machine over the network using `rmt`(1M). Since `ufsdump` is normally run by root, the name of the local machine must appear in the `/.rhosts` file of the remote machine. If the file is specified as *user*@*machine*:*device,* `ufsdump` will attempt to execute as the specified user on the remote machine. The specified user must have a `.rhosts` file on the remote machine that allows the user invoking the command from the local machine to access the remote machine.

`l`

Autoload. When the end-of-tape is reached before the dump is complete, take the drive offline and wait up to two minutes for the tape drive to be ready again. This gives autoloading (stackloader) tape drives a chance to load a new tape. If the drive is ready within two minutes, continue. If it is not, prompt for another tape and wait.

`L` *string*

Sets the tape label to *string*, instead of the default `none`. *string* may be no more than sixteen characters long. If it is longer, it is truncated and a warning printed; the dump will still be done. The tape label is specific to the `ufsdump` tape format, and bears no resemblance to IBM or ANSI-standard tape labels.

`n`

Notify all operators in the `sys` group that `ufsdump` requires attention by sending messages to their terminals, in a manner similar to that used by the `wall(1M)` command. Otherwise, such messages are sent only to the terminals (such as the console) on which the user running `ufsdump` is logged in.

`N` *device_name*

Use *device_name* when recording information in `/etc/dumpdates` (see the `u` option) and when comparing against information in `/etc/dumpdates` for incremental dumps. The *device_name* provided can contain no white space as defined in `scanf(3C)` and is case-sensitive.

`o`

Offline. Take the drive offline when the dump is complete or the end-of-media is reached and rewind the tape. In the case of some autoloading 8mm drives, the tape is removed from the drive automatically. This prevents another process which rushes in to use the drive, from inadvertently overwriting the media.

`s` *size*

Specify the *size* of the volume being dumped to. Not normally required, as `ufsdump` can detect end-of-media. When the specified size is reached, `ufsdump` waits for you to change the volume. `ufsdump` interprets the specified size as the length in feet for tapes and cartridges. The values should be a little smaller than the actual physical size of the media (for example, 425 for a 450-foot cartridge). Typical values for tape devices depend on the `c` option for cartridge devices:

1/2 inch tape
: 2300 feet

60-Mbyte 1/4 inch cartridge
: 425 feet

150-Mbyte 1/4 inch cartridge
: 700 feet

`S`

Size estimate. Determine the amount of space that is needed to perform the dump without actually doing it, and display the estimated number of bytes it will take. This is useful with incremental dumps to determine how many volumes of media will be needed.

`t` *tracks*

Specify the number of tracks for a cartridge tape. Not normally required, as `ufsdump` can detect end-of-media. The default is 9 tracks. The `t` option is not compatible with the `D` option. Values for Oracle-supported tape devices are:

60-Mbyte 1/4 inch cartridge
: 9 tracks

150-Mbyte 1/4 inch cartridge
: 18 tracks

T *time_wait*[hms]
Sets the amount of time to wait for an `autoload` command to complete. This option is ignored unless the `l` option has also been specified. The default time period to wait is two minutes. Specify time units with a trailing `h` (for hours), `m` (for minutes), or `s` (for seconds). The default unit is minutes.

u
Update the dump record. Add an entry to the file `/etc/dumpdates`, for each file system successfully dumped that includes the file system name (or *device_name* as specified with the `N` option), date, and dump level.

v
Verify. After each tape is written, verify the contents of the media against the source file system. If any discrepancies occur, prompt for new media, then repeat the dump/verification process. The file system *must* be unmounted. This option cannot be used to verify a dump to standard output.

w
Warning. List the file systems that have not been backed up within a day. This information is gleaned from the files `/etc/dumpdates` and `/etc/vfstab`. When the `w` option is used, all other options are ignored. After reporting, `ufsdump` exits immediately.

W
Warning with highlight. Similar to the `w` option, except that the `W` option includes all file systems that appear in `/etc/dumpdates`, along with information about their most recent dump dates and levels. File systems that have not been backed up within a day are highlighted.

Operands The following operand is supported:

files_to_dump
Specifies the files to dump. Usually it identifies a whole file system by its raw device name (for example, `/dev/rdsk/c0t3d0s6`). Incremental dumps (levels `1` to `9`) of files changed after a certain date only apply to a whole file system. Alternatively, *files_to_dump* can identify individual files or directories. All named directories that may be examined by the user running `ufsdump`, as well as any explicitly-named files, are dumped. This dump is equivalent to a level `0` dump of the indicated portions of the filesystem, except that `/etc/dumpdates` is not updated even if the `-u` option has been specified. In all cases, the files must be contained in the same file system, and the file system must be local to the system where `ufsdump` is being run.

files_to_dump is required and must be the last argument on the command line.

If no *options* are given, the default is `9uf /dev/rmt/0` *files_to_dump*.

Usage See largefile(5) for the description of the behavior of ufsdump when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

Examples **EXAMPLE 1** Using ufsdump

The following command makes a full dump of a root file system on c0t3d0, on a 150-MByte cartridge tape unit 0:

```
example# ufsdump 0cfu /dev/rmt/0 /dev/rdsk/c0t3d0s0
```

The following command makes and verifies an incremental dump at level 5 of the usr partition of c0t3d0, on a 1/2 inch reel tape unit 1,:

```
example# ufsdump 5fuv /dev/rmt/1 /dev/rdsk/c0t3d0s6
```

Exit Status While running, ufsdump emits many verbose messages. ufsdump returns the following exit values:

0
Normal exit.

1
Startup errors encountered.

3
Abort – no checkpoint attempted.

Files /dev/rmt/0
default unit to dump to

/etc/dumpdates
dump date record

/etc/group
to find group sys

/etc/hosts
to gain access to remote system with drive

/etc/vfstab
list of file systems

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also cpio(1), tar(1), dd(1M), devnm(1M), fssnap(1M), prtvtoc(1M), rmt(1M), shutdown(1M), ufsrestore(1M), volcopy(1M), wall(1M), scanf(3C), ufsdump(4), attributes(5), largefile(5), st(7D)

Notes

Read Errors Fewer than 32 read errors on the file system are ignored.

Process Per Reel Because each reel requires a new process, parent processes for reels that are already written hang around until the entire tape is written.

Operator Intervention `ufsdump` requires operator intervention on these conditions: end of volume, end of dump, volume write error, volume open error or disk read error (if there are more than a threshold of 32). In addition to alerting all operators implied by the `n` option, `ufsdump` interacts with the operator on `ufsdump`'s control terminal at times when `ufsdump` can no longer proceed, or if something is grossly wrong. All questions `ufsdump` poses *must* be answered by typing `yes` or `no`, as appropriate.

Since backing up a disk can involve a lot of time and effort, `ufsdump` checkpoints at the start of each volume. If writing that volume fails for some reason, `ufsdump` will, with operator permission, restart itself from the checkpoint after a defective volume has been replaced.

Suggested Dump Schedule It is vital to perform full, "level `0`", dumps at regular intervals. When performing a full dump, bring the machine down to single-user mode using shutdown(1M). While preparing for a full dump, it is a good idea to clean the tape drive and heads. Incremental dumps should be performed with the system running in single-user mode.

Incremental dumps allow for convenient backup and recovery of active files on a more frequent basis, with a minimum of media and time. However, there are some tradeoffs. First, the interval between backups should be kept to a minimum (once a day at least). To guard against data loss as a result of a media failure (a rare, but possible occurrence), capture active files on (at least) two sets of dump volumes. Another consideration is the desire to keep unnecessary duplication of files to a minimum to save both operator time and media storage. A third consideration is the ease with which a particular backed-up version of a file can be located and restored. The following four-week schedule offers a reasonable tradeoff between these goals.

	Sun	Mon	Tue	Wed	Thu	Fri
Week 1:	Full	5	5	5	5	3
Week 2:		5	5	5	5	3
Week 3:		5	5	5	5	3
Week 4:		5	5	5	5	3

Although the Tuesday through Friday incrementals contain "extra copies" of files from Monday, this scheme assures that any file modified during the week can be recovered from the previous day's incremental dump.

Process Priority of ufsdump

`ufsdump` uses multiple processes to allow it to read from the disk and write to the media concurrently. Due to the way it synchronizes between these processes, any attempt to run dump with a `nice` (process priority) of '–5' or better will likely make `ufsdump` run *slower* instead of faster.

Overlapping Partitions

Most disks contain one or more overlapping slices because slice 2 covers the entire disk. The other slices are of various sizes and usually do not overlap. For example, a common configuration places `root` on slice 0, `swap` on slice 1, `/opt` on slice 5 and `/usr` on slice 6.

It should be emphasized that `ufsdump` dumps one `ufs` file system at a time. Given the above scenario where slice 0 and slice 2 have the same starting offset, executing `ufsdump` on slice 2 with the intent of dumping the entire disk would instead dump only the `root` file system on slice 0. To dump the entire disk, the user must dump the file systems on each slice separately.

Bugs

The `/etc/vfstab` file does not allow the desired frequency of backup for file systems to be specified (as `/etc/fstab` did). Consequently, the `w` and `W` options assume file systems should be backed up daily, which limits the usefulness of these options.

Name ufsrestore – incremental file system restore

Synopsis

```
/usr/sbin/ufsrestore i | r | R | t | x [abcdfhlmostvyLT]
     [archive_file] [factor] [dumpfile] [n] [label]
     [timeout] [filename]...
```

Description The `ufsrestore` utility restores files from backup media created with the `ufsdump` command. `ufsrestores`'s actions are controlled by the *key* argument. The *key* is exactly one `function` *letter* (`i`, `r`, `R`, `t`, or `x`) and zero or more *function modifiers* (letters). The *key* string contains no SPACE characters. Function modifier arguments are listed on the command line in the same order as their corresponding function modifiers appear in the *key* string.

filename arguments which appear on the command line, or as arguments to an interactive command, are treated as shell `glob` patterns by the `x` and `t` functions; any files or directories matching the patterns are selected. The metacharacters `*`, `?`, and `[ ]` must be protected from the shell if they appear on the command line. There is no way to quote these metacharacters to explicitly match them in a filename.

The temporary files `rstdir*` and `rstmode*` are placed in `/tmp` by default. If the environment variable `TMPDIR` is defined with a non-empty value, that location is used instead of `/tmp`.

Options

Function Letters You must specify one (and only one) of the function letters listed below. Note that `i`, `x`, and `r` are intended to restore files into an empty directory. The `R` function is intended for restoring into a populated directory.

`i` Interactive. After reading in the directory information from the media, `ufsrestore` invokes a shell-like interface that allows you to browse through the dump file's directory hierarchy and select individual files to be extracted. Restoration has the same semantics as `x` (see below). See `Interactive Commands`, below, for a description of available commands.

`r` Recursive. Starting with an empty directory and a level 0 dump, the `r` function recreates the filesystem relative to the current working directory, exactly as it appeared when the dump was made. Information used to restore incremental dumps on top of the full dump (for example, `restoresymtable`) is also included. Several `ufsrestore` runs are typical, one for each higher level of dump (0, 1, ..., 9). Files that were deleted between the level 0 and a subsequent incremental dump will not exist after the final restore. To completely restore a file system, use the `r` function to restore the level 0 dump, and again for each incremental dump. Although this function letter is intended for a complete restore onto a new file system (one just created with `newfs(1M)`), if the file system contains files not on the backup media, they are preserved.

`R` Resume restoring. If an `r`-mode `ufsrestore` was interrupted, this function prompts for the volume from which to resume restoring and continues the restoration from where it was left off. Otherwise identical to `r`.

t Table of contents. List each *filename* that appears on the media. If no *filename* argument is given, the root directory is listed. This results in a list of all files on the media, unless the `h` function modifier is in effect. The table of contents is taken from the media or from the specified archive file, when the `a` function modifier is used. The `a` function modifier is mutually exclusive with the `x` and `r` function letters.

x Extract the named files from the media. Files are restored to the same relative locations that they had in the original file system.

If the *filename* argument matches a directory whose contents were written onto the media, and the `h` modifier is not in effect, the directory is recursively extracted, relative to the current directory, which is expected to be empty. For each file, the owner, modification time, and mode are restored (if possible).

If you omit the *filename* argument or specify `.`, the root directory is extracted. This results in the entire tape being extracted, unless the `h` modifier is in effect. . With the `x` function, existing files are overwritten and `ufsrestore` displays the names of the overwritten files. Overwriting a currently-running executable can have unfortunate consequences.

Use the `x` option to restore partial file system dumps, as they are (by definition) not entire file systems.

Function Modifiers

a *archive_file* Read the table of contents from *archive_file* instead of the media. This function modifier can be used in combination with the `t`, `i`, or `x` function letters, making it possible to check whether files are on the media without having to mount the media. When used with the `x` and interactive (`i`) function letters, it prompts for the volume containing the file(s) before extracting them.

b *factor* Blocking factor. Specify the blocking factor for tape reads. For variable length SCSI tape devices, unless the data was written with the default blocking factor, a blocking factor at least as great as that used to write the tape must be used; otherwise, an error will be generated. Note that a tape block is 512 bytes. Refer to the man page for your specific tape driver for the maximum blocking factor.

c Convert the contents of the media in 4.1BSD format to the new `ufs` file system format.

d Debug. Turn on debugging output.

f *dump_file* Use *dump_file* instead of `/dev/rmt/0` as the file to restore from. Typically *dump_file* specifies a tape drive. If *dump_file* is specified as '–', `ufsrestore` reads from the standard input. This allows `ufsdump(1M)` and `ufsrestore` to be used in a pipeline to copy a file system:

```
example# ufsdump 0f - /dev/rdsk/c0t0d0s7 \
 | (cd /home;ufsrestore xf -)
```

If the name of the file is of the form *machine*:*device*, the restore is done from the specified machine over the network using rmt(1M). Since `ufsrestore` is normally run by root, the name of the local machine must appear in the `/.rhosts` file of the remote machine. If the file is specified as *user*@*machine*:*device*, `ufsrestore` will attempt to execute as the specified user on the remote machine. The specified user must have a `.rhosts` file on the remote machine that allows the user invoking the command from the local machine to access the remote machine.

h — Extract or list the actual directory, rather than the files that it references. This prevents hierarchical restoration of complete subtrees from the tape.

l — Autoload. When the end-of-tape is reached before the restore is complete, take the drive off-line and wait up to two minutes (the default, see the `T` function modifier) for the tape drive to be ready again. This gives autoloading (stackloader) tape drives a chance to load a new tape. If the drive is ready within two minutes, continue. If it is not, prompt for another tape and wait.

L *label* — The label that should appear in the header of the dump file. If the labels do not match, `ufsrestore` issues a diagnostic and exits. The tape label is specific to the `ufsdump` tape format, and bears no resemblance to IBM or ANSI-standard tape labels.

m — Extract by inode numbers rather than by filename to avoid regenerating complete pathnames. Regardless of where the files are located in the dump hierarchy, they are restored into the current directory and renamed with their inode number. This is useful if only a few files are being extracted.

o — Offline. Take the drive off-line when the restore is complete or the end-of-media is reached and rewind the tape. In the case of some autoloading 8mm drives, the tape is removed from the drive automatically.

s *n* — Skip to the *n*th file when there are multiple dump files on the same tape. For example, the command:

```
example# ufsrestore xfs /dev/rmt/0hn 5
```

would position you to the fifth file on the tape when reading volume 1 of the dump. If a dump extends over more than one volume, all volumes except the first are assumed to start at position 0, no matter what "`s` *n*" value is specified.

If "`s` *n*" is specified, the backup media must be at BOT (beginning of tape). Otherwise, the initial positioning to read the table of contents will fail, as it is performed by skipping the tape forward *n*`-1` files rather than by using absolute positioning. This is because on some devices absolute positioning is very time consuming.

T timeout [hms] — Sets the amount of time to wait for an autoload command to complete. This function modifier is ignored unless the `l` function modifier has also been specified. The default timeout period is two minutes. The time units may be specified as a trailing `h` (hours), `m` (minutes), or `s` (seconds). The default unit is minutes.

v — Verbose. `ufsrestore` displays the name and inode number of each file it restores, preceded by its file type.

y — Do not ask whether to abort the restore in the event of tape errors. `ufsrestore` tries to skip over the bad tape block(s) and continue as best it can.

Interactive Commands

`ufsrestore` enters interactive mode when invoked with the `i` function letters. Interactive commands are reminiscent of the shell. For those commands that accept an argument, the default is the current directory. The interactive options are:

add [*filename*] — Add the named file or directory to the list of files to extract. If a directory is specified, add that directory and its files (recursively) to the extraction list (unless the `h` modifier is in effect).

cd *directory* — Change to *directory* (within the dump file).

delete [*filename*] — Delete the current directory, or the named file or directory from the list of files to extract. If a directory is specified, delete that directory and all its descendents from the extraction list (unless the `h` modifier is in effect). The most expedient way to extract a majority of files from a directory is to add that directory to the extraction list, and then delete specific files to omit.

extract — Extract all files on the extraction list from the dump media. `ufsrestore` asks which volume the user wishes to mount. The fastest way to extract a small number of files is to start with the last volume and work toward the first. If "`s` *n*" is given on the command line, volume 1 will automatically be positioned to file *n* when it is read.

help — Display a summary of the available commands.

ls [*directory*] — List files in *directory* or the current directory, represented by a '`.`' (period). Directories are appended with a '`/`' (slash). Entries marked for extraction are prefixed with a '`*`' (asterisk). If the verbose option is in effect, inode numbers are also listed.

marked [*directory*]	Like `ls`, except only files marked for extraction are listed.
pager	Toggle the pagination of the output from the `ls` and `marked` commands. The pager used is that defined by the `PAGER` environment variable, or more(1) if that envar is not defined. The `PAGER` envar may include white-space-separated arguments for the pagination program.
pwd	Print the full pathname of the current working directory.
quit	`ufsrestore` exits immediately, even if the extraction list is not empty.
setmodes	Prompts: `set owner/mode for '.'` (period). Type `y` for yes to set the mode (permissions, owner, times) of the current directory '.' (period) into which files are being restored equal to the mode of the root directory of the file system from which they were dumped. Normally, this is what you want when restoring a whole file system, or restoring individual files into the same locations from which they were dumped. Type `n` for no, to leave the mode of the current directory unchanged. Normally, this is what you want when restoring part of a dump to a directory other than the one from which the files were dumped.
setpager *command*	Sets the command to use for paginating output instead of the default or that inherited from the environment. The *command* string may include arguments in addition to the command itself.
verbose	Toggle the status of the `v` modifier. While `v` is in effect, the `ls` command lists the inode numbers of all entries, and `ufsrestore` displays information about each file as it is extracted.
what	Display the dump header on the media.

Operands The following operands are supported.

filename Specifies the pathname of files (or directories) to be restored to disk. Unless the `h` function modifier is also used, a directory name refers to the files it contains, and (recursively) its subdirectories and the files they contain. *filename* is associated with either the `x` or `t` function letters, and must come last.

Usage See `largefile(5)` for the description of the behavior of `ufsrestore` when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

Exit Status The following exit values are returned:

`0` Successful completion.

`1` An error occurred. Verbose messages are displayed.

Environment Variables PAGER The command to use as a filter for paginating output. This can also be used to specify the options to be used. Default is `more(1)`.

TMPDIR Selects the directory for temporary files. Defaults to `/tmp` if not defined in the environment.

Files

`/dev/rmt/0`	the default tape drive
`$TMPDIR/rstdir*`	file containing directories on the tape
`$TMPDIR/rstmode*`	owner, mode, and timestamps for directories
`./restoresymtable`	information passed between incremental restores

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also `more(1)`, `mkfs(1M)`, `mount(1M)`, `rmt(1M)`, `ufsdump(1M)`, `ufsdump(4)`, `attributes(5)`, `largefile(5)`

Diagnostics `ufsrestore` complains about bad option characters.

Read errors result in complaints. If `y` has been specified, or the user responds `y`, `ufsrestore` will attempt to continue.

If the dump extends over more than one tape, `ufsrestore` asks the user to change tapes. If the `x` or `i` function letter has been specified, `ufsrestore` also asks which volume the user wishes to mount. If the `s` modifier has been specified, and volume 1 is mounted, it is automatically positioned to the indicated file.

There are numerous consistency checks that can be listed by `ufsrestore`. Most checks are self-explanatory or can "never happen". Common errors are given below.

`Converting to new file system format`
A dump tape created from the old file system has been loaded. It is automatically converted to the new file system format.

filename`: not found on tape`
The specified file name was listed in the tape directory, but was not found on the tape. This is caused by tape read errors while looking for the file, using a dump tape created on an active file system, or restoring a partial dump with the `r` function.

`expected next file` *inumber*`, got` *inumber*
A file that was not listed in the directory showed up. This can occur when using a dump tape created on an active file system.

`Incremental tape too low`
When doing an incremental restore, a tape that was written before the previous incremental tape, or that has too low an incremental level has been loaded.

`Incremental tape too high`
When doing incremental restore, a tape that does not begin its coverage where the previous incremental tape left off, or one that has too high an incremental level has been loaded.

`media read error: invalid argument`
Blocking factor specified for read is smaller than the blocking factor used to write data.

`Tape read error while restoring`
`Tape read error while skipping over inode number`
`Tape read error while trying to resynchronize`
`A tape read error has occurred`
If a file name is specified, then its contents are probably partially wrong. If an inode is being skipped or the tape is trying to resynchronize, then no extracted files have been corrupted, though files may not be found on the tape.

`resync ufsrestore, skipped` *num*
After a tape read error, `ufsrestore` may have to resynchronize itself. This message lists the number of blocks that were skipped over.

`Incorrect tape label. Expected 'foo', got 'bar'.`
The `L` option was specified, and its value did not match what was recorded in the header of the dump file.

Notes `ufsrestore` can get confused when doing incremental restores from dump tapes that were made on active file systems.

A `level 0` dump must be done after a full restore. Because `ufsrestore` runs in user mode, it has no control over inode allocation. This means that `ufsrestore` repositions the files, although it does not change their contents. Thus, a full dump must be done to get a new set of directories reflecting the new file positions, so that later incremental dumps will be correct.

REFERENCE

System Administration Commands - Part 3

Name unshare – make local resource unavailable for mounting by remote systems

Synopsis

```
unshare [-F protocol] [-a | [-t] [pathname | sharename]]
```

Description The `unshare` command unpublishes a file system share, which makes a shared local file system unavailable for file sharing *protocol*. If the `-F` *protocol* option is omitted, then the first file sharing procotol listed in the `/etc/dfs/fstypes` file used as the default sharing protocol.

Options

`-F` *protocol*
Identify the file sharing protocol.

`-a`
Unpublish all active shares.

`-t`
Temporarily unpublish a share. By default, all shares are permanently unpublished.

Files

`/etc/dfs/fstypes`
List of file sharing protocols. NFS is the default file-sharing protocol.

`/etc/dfs/sharetab`
Contains a table of local resources published by the `share` command.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcs

See Also `share(1M)`, `shareall(1M)`, `attributes(5)`

Notes If *pathname* or *resourcename* is not found in the share information, an error message is sent to standard error.

When an `unshare` command completes successfully, a client mounting a file system specified in that `unshare` command no longer has access to that file system.

Name unshare_nfs – make local NFS file systems unavailable for mounting by remote systems

Synopsis

```
unshare [-F nfs] pathname
```

Description The `unshare` command makes local file systems unavailable for mounting by remote systems. The shared file system must correspond to a line with NFS as the *FSType* in the file `/etc/dfs/sharetab`.

Options The following options are supported:

`-F` This option may be omitted if NFS is the first file system type listed in the file `/etc/dfs/fstypes`.

Files

`/etc/dfs/fstypes`

`/etc/dfs/sharetab`

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	service/file-system/nfs

See Also `nfsd(1M)`, `share(1M)`, `attributes(5)`

Notes If the file system being unshared is a symbolic link to a valid pathname, the canonical path (the path which the symbolic link follows) will be unshared.

For example, if `/export/foo` is a symbolic link to `/export/bar` (`/export/foo -> /export/bar`), the following `unshare` command will result in `/export/bar` as the unshared pathname (and not `/export/foo`):

```
example#  unshare -F nfs /export/foo
```

For file systems that are accessed by NFS Version 4 clients, once the unshare is complete, all NFS Version 4 state (open files and file locks) are released and unrecoverable by the clients. If the intent is to share the file system after some administrative action, the NFS daemon (`nfsd`) should first be stopped and then the file system unshared. After the administrative action is complete, the file system would then be shared and the NFS daemon restarted. See `nfsd(1M)`

Name update_drv – modify device driver attributes

Synopsis

```
update_drv [-f | -v] [-n] driver_module

update_drv [-b basedir] [-f | -v] [-n] -a [-m 'permission']
     [-i 'identify-name'] [-P 'privilege'] [-p 'policy'] driver_module

update_drv [-b basedir] [-f | -v] [-n] -d [-m 'permission']
     [-i 'identify-name'] [-P 'privilege'] [-p 'policy'] driver_module
```

Description The `update_drv` command informs the system about attribute changes to an installed device driver. It can be used to re-read the `driver.conf(4)` file, or to add, modify, or delete a driver's minor node permissions or aliases.

Without options, `update_drv` reloads the `driver.conf` file.

Upon successfully updating the aliases, the driver binding takes effect upon reconfig boot or hotplug of the device.

Upon successfully updating the permissions, only the new driver minor nodes get created with the modified set of file permissions. Existing driver minor nodes do not get modified.

Options The following options are supported:

`-a`
Add a *permission*, *aliases*, *privilege* or *policy* entry.

With the `-a` option specified, a permission entry (using the `-m` option), or a driver's aliases entry (using the `-i` option), a device privilege (using the `-P` option) or a a device policy (using the `-p` option), can be added or updated. If a matching minor node permissions entry is encountered (having the same driver name and the minor node), it is replaced. If a matching aliases entry is encountered (having a different driver name and the same alias), an error is reported.

The `-a` and `-d` options are mutually exclusive.

`-b` *basedir*
Installs or modifies the driver on the system with a root directory of basedir rather than installing on the system executing `update_drv`.

Note – The root file system of any non-global zones must not be referenced with the `-b` option. Doing so might damage the global zone's file system, might compromise the security of the global zone, and might damage the non-global zone's file system. See `zones(5)`.

`-d`
Delete a *permission*, *aliases*, *privilege* or *policy* entry.

The `-m` *permission*, `-i` *identify-name*, `-P` *privilege* or the `-p` *policy* option needs to be specified with the `-d` option.

The `-d` and `-a` options are mutually exclusive.

If the entry doesn't exist `update_drv` returns an error.

`-f`
Force the system to reread the `driver.conf` file even if the driver module cannot be unloaded. See `NOTES` section for details.

Without this option, when removing an alias for a driver, `update_drv` updates the binding files for the next boot, but returns an error if one or more devices that reference the driver-alias binding remains. *With* the `-f` option, `update_drv` does not return an error if such devices remain.

`-i` '*identify-name*'
A white-space separated list of aliases for the driver. If `-a` or `-d` option is not specified then this option is ignored. The *identify-name* string is mandatory. If all aliases need to be removed, `rem_drv(1M)` is recommended.

`-m` '*permission*'
Specify a white-space separated list of file system permissions for the device node of the device driver. If `-a` or `-d` option is not specified then, this option is ignored. The permission string is mandatory.

`-n`
Do not try to load and attach *device_driver*, just modify the system configuration files for that driver.

`-p` '*policy*'
With the `-a` option, policy is a white-space separated list of complete device policies. For the `-d` option, policy is a white space separated list of minor device specifications. The minor device specifications are matched exactly against the entries in `/etc/security/device_policy`, that is., no wildcard matching is performed.

`-P` '*privilege*'
With the `-a` option, privilege is a comma separated list of additional driver privileges. For the `-d` option, privilege is a single privilege. The privileges are added to or removed from the `/etc/security/extra_privs` file.

`-v`
Verbose.

Examples

EXAMPLE 1 Adding or Modifying an Existing Minor Permissions Entry

The following command adds or modifies the exisitng minor permissions entry of the `clone` driver:

```
example# update_drv -a -m 'llc1 777 joe staff' clone
```

EXAMPLE 2 Removing All Minor Permissions Entries

The following command removes all minor permission entries of the `usbprn` driver, the USB printer driver:

```
example# update_drv -d -m '* 0666 root sys' usbprn
```

EXAMPLE 3 Adding a Driver Aliases Entry

The following command adds a driver aliases entry of the `ugen` driver with the identity string of `usb459,20`:

```
example# update_drv -a -i '"usb459,20"' ugen
```

EXAMPLE 4 Re-reading the `driver.conf` File For the ohci Driver

The following command re-reads the `driver.conf`(4) file.

```
 example# update_drv ohci
```

EXAMPLE 5 Requiring a Self-defined Privilege to Open a tcp Socket

The following command requires a self-defined privilege to open a tcp socket:

```
example#   update_drv -a -P net_tcp -p \
     'write_priv_set=net_tcp read_priv_set=net_tcp' tcp
```

EXAMPLE 6 Establishing a Path-oriented Alias

The following command establishes a path-oriented alias to force a specific driver, `qlt`, to be used for a particular device path:

```
example#   update_drv -a -i '"/pci@8,600000/SUNW,qlc@4"' qlt
```

Exit Status

The following exit values are returned:

`0`
: Successful completion.

`>0`
: An error occurred.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also

`add_drv(1M)`, `modunload(1M)`, `rem_drv(1M)`, `driver.conf(4)`, `attributes(5)`, `privileges(5)`

Notes If `-a` or `-d` options are specified, `update_drv` does not reread the `driver.conf` file.

A forced update of the `driver.conf` file reloads the `driver.conf` file without reloading the driver binary module. In this case, devices which cannot be detached reference driver global properties from the old `driver.conf` file, while the remaining driver instances reference global properties in the new `driver.conf` file.

It is possible to add an alias , which changes the driver binding of a device already being managed by a different driver. A force update with the `-a` option tries to bind to the new driver and report error if it cannot. If you specify more than one of the `-m`, `-i`, `-P` or `-p` options, a force flag tries to modify aliases or permissions. This is done even if the other operation fails and vice-versa.

Name useradd – administer a new user login on the system

Synopsis

```
useradd [-A authorization [,authorization...]]
      [-b base_dir] [-c comment] [-d dir] [-e expire]
      [-f inactive] [-g group] [-G group [,group]...]
      [-K key=value] [-m [-k skel_dir]] [-p projname]
      [-P profile [,profile...]] [-R role [,role...]]
      [-s shell] [-S repository] [-u uid [-o]] login

useradd -D [-A authorization [,authorization...]]
      [-b base_dir] [-s shell [-k skel_dir]] [-e expire]
      [-f inactive] [-g group] [-K key=value] [-p projname]
      [-P profile [,profile...]] [-R role [,role...]]
```

Description `useradd` adds a new user to the `passwd`, `shadow`, and `user_attr` databases in the `files` and `ldap` repositories. The `-A` and `-P` options respectively assign authorizations and profiles to the user. The `-R` option assigns roles to a user. The `-p` option associates a project with a user. The `-K` option adds a *key=value* pair to `user_attr` entry for the user. Multiple *key=value* pairs may be added with multiple `-K` options.

`useradd` also creates supplementary group memberships for the user (`-G` option) and creates the home directory (`-m` option) for the user if requested. The new login remains locked until the `passwd(1)` command is executed.

Specifying `useradd -D` with the `-s`, `-k`,`-g`, `-b`, `-f`, `-e`, `-A`, `-P`, `-p`, `-R`, or `-K` option (or any combination of these options) sets the default values for the respective fields. See the `-D` option, below. Subsequent `useradd` commands without the `-D` option use these arguments.

The system file entries created with this command have a limit of 2048 characters per line. Specifying long arguments to several options can exceed this limit.

`useradd` requires that usernames be in the format described in `passwd(4)`. A warning message is displayed if these restrictions are not met. See `passwd(4)` for the requirements for usernames.

An administrator must be granted the User Management Profile to be able to create a new user. The authorizations required to set the various fields in `passwd`, `shadow` and `user_attr` can be found in `passwd(4)`, `shadow(4)`, and `user_attr(4)`. The authorizations required to assign groups and projects can be found in `group(4)` and `project(4)`.

Options The following options are supported:

`-A` *authorization*
: One or more comma-separated authorizations defined in `auth_attr(4)`. Only a user or role who has `grant` rights to the authorization can assign it to an account.

`-b` *base_dir*
: The base directory for new login home directories (see the `-d` option below. When a new user account is being created, *base_dir* must already exist unless the `-m` option or the `-d` option is also specified.

`-c` *comment*
Any text string. It is generally a short description of the login, and is currently used as the field for the user's full name. This information is stored in the user's `passwd` entry.

`-d` *dir* | *server*:*dir*
Specifies the home directory path for the new user. If no server name is specified, the specified directory is maintained in the `passwd(4)` database.

The optional server name specifies the host on which the home directory resides. Entries in this form depend on the automounter, and are maintained in the `auto_home` map. The path `/home/`*username* is maintained in the `passwd(4)` database. When the user subsequently references `/home/`*username*, the automounter will mount the specified directory on `/home/`*username*.

`-D`
Display the default values for `group`, `base_dir`, `skel_dir`, `shell`, `inactive`, `expire`, `proj`, `projname` and `key=value` pairs. When used with the `-g`, `-b`, `-f`, `-e`, `-A`, `-P`, `-p`, `-R`, or `-K` options, the `-D` option sets the default values for the specified fields. The default values are:

group
`other` (GID of 1)

base_dir
`/export/home`

skel_dir
`/etc/skel`

shell
`/usr/bin/bash`

inactive
`0`

expire
null

auths
null

profiles
null

proj
`3`

projname
`default`

key=value (pairs defined in `user_attr(4)`
not present

roles
null

-e *expire*
Specify the expiration date for a login. After this date, no user will be able to access this login. The expire option argument is a date entered using one of the date formats included in the template file `/etc/datemsk`. See `getdate(3C)`.

If the date format that you choose includes spaces, it must be quoted. For example, you can enter `10/6/90` or `October 6, 1990`. A null value (`""`) defeats the status of the expired date. This option is useful for creating temporary logins.

-f *inactive*
The maximum number of days allowed between uses of a login ID before that ID is declared invalid. Normal values are positive integers. A value of `0` defeats the status.

-g *group*
An existing group's integer ID or character-string name. Without the `-D` option, it defines the new user's primary group membership and defaults to the default group. You can reset this default value by invoking `useradd -D -g` *group*. GIDs 0-99 are reserved for allocation by the Solaris Operating System.

-G *group*
An existing group's integer ID or character-string name. It defines the new user's supplementary group membership. Duplicates between *group* with the `-g` and `-G` options are ignored. No more than `NGROUPS_MAX` groups can be specified. GIDs 0-99 are reserved for allocation by the Solaris Operating System.

-K *key=value*
A *key=value* pair to add to the user's attributes. Multiple `-K` options may be used to add multiple *key=value* pairs. The generic `-K` option with the appropriate key may be used instead of the specific implied key options (`-A`, `-P`, `-R`, `-p`). See `user_attr(4)` for a list of valid *key=value* pairs. The “type” key is not a valid key for this option. Keys may not be repeated.

-k *skel_dir*
A directory that contains skeleton information (such as `.profile`) that can be copied into a new user's home directory. This directory must already exist. The system provides the `/etc/skel` directory that can be used for this purpose.

-m
Create the new user's home directory if it does not already exist. If the directory already exists, it must have read, write, and execute permissions by *group*, where *group* is the user's primary group. If the server name specified to the `-d` option is a remote host then the system will not attempt to create the home directory.

If the directory does not already exist and the parent directory is the mount point of a ZFS dataset, then a child of that dataset will be created and mounted at the specified location.

The user is delegated permissions to create ZFS snapshots and promote them. The newly created dataset will inherit the encryption setting from its parent. If it is encrypted, the user is granted permission to change its wrapping key.

`-o`
This option allows a UID to be duplicated (non-unique).

`-P` *profile*
One or more comma-separated execution profiles defined in `prof_attr(4)`.

`-p` *projname*
Name of the project with which the added user is associated. See the *projname* field as defined in `project(4)`.

`-R` *role*
One or more comma-separated execution profiles defined in `user_attr(4)`. Roles cannot be assigned to other roles.

`-s` *shell*
Full pathname of the program used as the user's shell on login. If unspecified, it will default to any value previously configured with the `-D -s` option. If no default has been set with `-D -s`, then `/usr/bin/bash` will be used. The value of *shell* must be a valid executable file.

`-S` *repository*
The valid repositories are `files`, `ldap`. The repository specifies which name service will be updated. The default repository is `files`. When the repository is `files`, the authorizations, profiles, and roles can be present in other name service repositories and can be assigned to a user in the `files` repository. When the repository is `ldap`, all the assignable attributes must be present in the `ldap` repository.

`-u` *uid*
The UID of the new user. This UID must be a non-negative decimal integer below `MAXUID` as defined in `<sys/param.h>`. The UID defaults to the next available (unique) number above the highest number currently assigned. For example, if UIDs 100, 105, and 200 are assigned, the next default UID number will be 201. UIDs `0-99` are reserved for allocation by the Solaris Operating System.

Exit Status In case of an error, `useradd` prints an error message and exits with one of the following values:

`1` No permission for attempted operation.

`2` The command syntax was invalid. A usage message for the `usermod` command is displayed.

`3` An invalid argument was provided to an option.

`4` The *gid* or *uid* given with the `-u` option is already in use.

`5` The `password` and `shadow` files are not consistent with each other. `pwconv(1M)` might be of use to correct possible errors. See `passwd(4)` and `shadow(4)`.

6 The login to be modified does not exist, the *gid* or the *uid* does not exist.

7 The `group`, `passwd`, or `shadow` file is missing.

9 A group or user name is already in use.

10 Cannot update the `passwd`, `shadow`, or `user_attr` file.

11 Insufficient space to move the home directory (`-m` option).

12 Unable to create, remove, or move the new home directory.

13 Requested login is already in use.

14 Unexpected failure.

16 Unable to update the group database.

17 Unable to update the project database.

18 Insufficient authorization.

19 Does not have role.

20 Does not have profile.

21 Does not have privilege.

22 Does not have label.

23 Does not have group.

24 System not running Trusted Extensions.

25 Does not have project.

26 Unable to update `auto_home`.

Files

`/etc/datemsk`

`/etc/passwd`

`/etc/shadow`

`/etc/group`

`/etc/skel`

`/usr/include/limits.h`

`/etc/user_attr`

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os
Interface Stability	Committed

See Also auths(1), passwd(1), profiles(1), roles(1), groupadd(1M), groupdel(1M), groupmod(1M), grpck(1M), logins(1M), pwck(1M), userdel(1M), usermod(1M), getdate(3C), auth_attr(4), group(4), passwd(4), prof_attr(4), project(4), user_attr(4), attributes(5)

Managing User Accounts and User Environments in Oracle Solaris 11.1

Diagnostics In case of an error, useradd displays an error message and exits with a non-zero status.

The following indicates that login specified is already in use:

```
UX: useradd: ERROR: login is already in use. Choose another.
```

The following indicates that the *uid* specified with the -u option is not unique:

```
UX: useradd: ERROR: uid uid is already in use. Choose another.
```

The following indicates that the *group* specified with the -g option has not yet been created:

```
UX: useradd: ERROR: group group does not exist. Choose another.
```

The following indicates that the *uid* specified with the -u option is in the range of reserved UIDs (from 0-99):

```
UX: useradd: WARNING: uid uid is reserved.
```

The following indicates that the *uid* specified with the -u option exceeds MAXUID as defined in <sys/param.h>:

```
UX: useradd: ERROR: uid uid is too big. Choose another.
```

The following indicates that the /etc/passwd or /etc/shadow files do not exist:

```
UX: useradd: ERROR: Cannot update system files - login cannot be created.
```

The following indicates that the user executing the command does not have sufficient authorization to perform the operation:

```
UX: roleadd: ERROR: Permission denied.
```

The following indicates that an invalid directory was specified in a useradd command:

```
UX: invalid_directory is not a valid directory. Choose another.
```

Notes The useradd utility adds definitions to the passwd, shadow, group, project, and user_attr databases in the scope (default or specified). It will verify the uniqueness of the user name (or role) and user id and the existence of any group names specified against the external name service.

Name userdel – delete a user's login from the system

Synopsis

```
userdel [-r] [-S repository] login
```

Description The `userdel` utility deletes a user account from the system and makes the appropriate account-related changes to the system file and file system.

An administrator must be granted the User Management Profile to be able to delete a user.

Options The following options are supported:

`-r` Remove the user's home directory from the system. This directory must exist. The files and directories under the home directory will no longer be accessible following successful execution of the command. A ZFS dataset that was created for the user's home directory will be removed. An `auto_home` entry that was added for the user will be deleted.

`-S` *repository* The valid repositories are `files`, `ldap`. The repository specifies which name service will be updated. The default repository is `files`.

Operands The following operands are supported:

login An existing login name to be deleted.

Exit Status The following exit values are returned:

`0` Successful completion.

`2` Invalid command syntax. A usage message for the `userdel` command is displayed.

`6` The account to be removed does not exist.

`8` The account to be removed is in use.

`10` Cannot update the `/etc/group` or `/etc/user_attr` file but the login is removed from the `/etc/passwd` file.

`12` Cannot remove or otherwise modify the home directory.

Files

`/etc/passwd` system password file

`/etc/shadow` system file contain users' encrypted passwords and related information

`/etc/group` system file containing group definitions

`/etc/user_attr` system file containing additional user attributes

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also auths(1), passwd(1), profiles(1), roles(1), users(1B), groupadd(1M), groupdel(1M), groupmod(1M), logins(1M), roleadd(1M), rolemod(1M), useradd(1M), usermod(1M), passwd(4), prof_attr(4), user_attr(4), attributes(5)

Notes The `userdel` utility deletes an account definition that is in the `group`, `passwd`, `shadow`, and `user_attr` databases in the `files` or `ldap` repository.

Name usermod – modify a user's login information on the system

Synopsis

```
usermod [-u uid [-o]] [-g group] [-G [+|-]group [, group...]]
     [-d dir [-m]] [-s shell] [-c comment] [-l new_name]
     [-f inactive] [-e expire]
     [-A [+|-]authorization [,authorization]]
     [-P [+|-]profile [,profile]] [-R [+|-]role [,role]]
     [-K key[+|-]=value] [-S repository] login
```

Description The `usermod` utility modifies a user's login definition on the system. It changes the definition of the specified login and makes the appropriate login-related system file and file system changes.

The system file entries created with this command have a limit of 512 characters per line. Specifying long arguments to several options might exceed this limit.

An administrator must be granted the User Security Profile to modify the security attributes for an existing user. To be able to modify the non-security attributes of an existing user requires the User Management Profile. The authorizations required to set the various fields in `passwd`, `shadow` and `user_attr` can be found in `passwd(4)`, `shadow(4)`, and `user_attr(4)`. The authorizations required to assign groups can be found in `group(4)`.

Options The following options are supported:

`-A` [+|-]*authorization*
: One or more comma separated authorizations as defined in `auth_attr(4)`. Only a user or role who has `grant` rights to the `authorization` can assign it to an account. This replaces any existing authorization setting. If no authorization list is specified, the existing setting is removed.

  A prefix + adds the authorization to the existing authorization; a prefix - removes the authorization from the existing authorization. With no prefix, *authorization* replaces the existing authorization.

`-c` *comment*
: Specify a comment string. *comment* can be any text string. It is generally a short description of the login, and is currently used as the field for the user's full name. This information is stored in the user's `passwd` entry.

`-d` *dir*
: Specify the new home directory of the user. It defaults to *base_dir/login*, where *base_dir* is the base directory for new login home directories, and `login` is the new login. This creates or modifies an `auto_home` entry for the user.

  The argument to the option can be specified as *server*:*dir* where *server* is the hostname of the machine on which the home directory resides and *dir* is the path to the user's home directory. If the server is a remote host then the home directory needs to be created on the

remote host for the system to mount it, when the user logs in. If no server name is specified then the home directory will be created on the host where the command is executed, when the `-m` option is used.

`-e` *expire*
Specify the expiration date for a login. After this date, no user will be able to access this login. The expire option argument is a date entered using one of the date formats included in the template file `/etc/datemsk`. See `getdate(3C)`.

For example, you may enter `10/6/90` or `October 6, 1990`. A value of "`''` defeats the status of the expired date.

`-f` *inactive*
Specify the maximum number of days allowed between uses of a login ID before that login ID is declared invalid. Normal values are positive integers. A value of `0` defeats the status.

`-g` *group*
Specify an existing group's integer ID or character-string name. It redefines the user's primary group membership.

`-G` [+|-]*group*
An existing group's integer ID or character-string name. It defines the new user's supplementary group membership. Duplicates between group with the `-g` and `-G` options are ignored. No more than `NGROUPS_MAX` groups can be specified. GIDs 0-99 are reserved for allocation by the Solaris Operating System.

A prefix + adds the group to the existing group; a prefix - removes the group from the existing group. With no prefix, *group* replaces the existing group.

`-K` *key*[+|-]=*value*
Replace existing or add to a user's *key*=*value* pair attributes. Multiple `-K` options can be used to replace or add multiple *key*=*value* pairs. However, keys must not be repeated. The generic `-K` option with the appropriate key can be used instead of the specific implied key options (`-A`, `-P`, `-R`, `-p`). See `user_attr(4)` for a list of valid *key*s. Values for these keys are usually found in man pages or other sources related to those keys. For example, see `project(4)` for guidance on values for the `project` key. Use the command `ppriv(1)` with the `-v` and `-l` options for a list of values for the keys `defaultpriv` and `limitpriv`. If no value is specified, the existing key is removed.

The keyword `type` can be specified with the value `role` or the value `normal`. When using the value `role`, the account changes from a normal user to a role; using the value `normal` keeps the account a normal user.

As a `role` account, no roles (`-R` or *roles*=*value*) can be present.

A prefix + adds the value to the existing value; a prefix - removes the value from the existing value. With no prefix, *value* replaces the existing value.

The prefix +/- operation is applicable only to the following keys: `auths`, `profiles`, `roles`, `project`, `limitpriv`, and `defaultpriv`.

`-l` *new_logname*
Specify the new login name for the user. See `passwd(4)` for the requirements for usernames.

`-m`
Move the user's home directory to the new directory specified with the `-d` option. If the directory already exists, it must have permissions read/write/execute by *group*, where *group* is the user's primary group. If the server name specified to the `-d` option is a remote host then the system will not attempt to create the home directory.

If the directory does not already exist, a new ZFS dataset will be created. In the global zone, the dataset is created as `rpool/export/home/`*rolename*. For non-global zones, the dataset will be created as *ROOT-dataset*`/export/home/`*rolename*. The mountpoint for the ZFS dataset is `/export/home/`*rolename* by default. If `-d` *path* is specified and it is a path on the local machine, the dataset will be mounted at the specified location. The role is delegated permissions to create ZFS snapshots and promote them. The newly created dataset will inherit the encryption setting from its parent. If it is encrypted, the role is granted permission to change its wrapping key.

`-o`
This option allows the specified UID to be duplicated (non-unique).

`-P` [+|-]*profile*
One or more comma-separated rights profiles defined in `prof_attr(4)`. This replaces any existing profile setting in `user_attr(4)`. If an empty profile list is specified, the existing setting is removed.

A prefix + adds the profile to the existing profile; a prefix - removes the profile from the existing profile. With no prefix, *profile* replaces the existing profile.

`-R` [+|-]*role*
One or more comma-separated roles (see `roleadd(1M)`). This replaces any existing role setting. If no role list is specified, the existing setting is removed.

A prefix + adds the role to the existing role; a prefix - removes the role from the existing role. With no prefix, *role* replaces the existing role.

`-s` *shell*
Specify the full pathname of the program that is used as the user's shell on login. The value of *shell* must be a valid executable file.

`-S` *repository*
The valid repositories are `files`, `ldap`. The repository specifies which name service will be updated. The default repository is `files`. When the repository is `files`, the authorizations, profiles, and roles can be present in other name service repositories and can be assigned to a user in the `files` repository. When the repository is `ldap`, all the assignable attributes must be present in the `ldap` repository.

`-u` *uid*
Specify a new UID for the user. It must be a non-negative decimal integer less than `MAXUID` as defined in `<param.h>`. The UID associated with the user's home directory is not modified with this option; a user will not have access to their home directory until the UID is manually reassigned using `chown(1)`.

Operands The following operands are supported:

`login`
An existing login name to be modified.

Examples **EXAMPLE 1** Assigning Privileges to a User

The following command adds the privilege that affects high resolution times to a user's initial, inheritable set of privileges.

```
# usermod -K defaultpriv=basic,proc_clock_highres jdoe
```

This command results in the following entry in `user_attr`:

```
jdoe::::type=normal;defaultpriv=basic,proc_clock_highres
```

EXAMPLE 2 Removing a Privilege from a User's Limit Set

The following command removes the privilege that allows the specified user to create hard links to directories and to unlink directories.

```
# usermod -K limitpriv=all,!sys_linkdir jdoe
```

This command results in the following entry in `user_attr`:

```
jdoe::::type=normal;defaultpriv=basic,limitpriv=all,!sys_linkdir
```

EXAMPLE 3 Removing a Privilege from a User's Basic Set

The following command removes the privilege that allows the specified user to examine processes outside the user's session.

```
# usermod -K defaultpriv=basic,!proc_session jdoe
```

This command results in the following entry in `user_attr`:

```
jdoe::::type=normal;defaultpriv=basic,!proc_session;limitpriv=all
```

EXAMPLE 4 Assigning a Role to a User

The following command assigns a role to a user. The role must have been created prior to this command through use of `roleadd(1M)`.

```
# usermod -R mailadm jdoe
```

This command results in the following entry in `user_attr`:

```
jdoe::::type=normal;roles=mailadm;defaultpriv=basic;limitpriv=all
```

EXAMPLE 5 Removing All Profiles from a User

The following command removes all profiles that were granted to a user directly. The user will still have any rights profiles that are granted by means of the PROFS_GRANTED key in policy.conf(4).

```
# usermod -P "" jdoe
```

Exit Status In case of an error, usermod prints an error message and exits with one of the following values:

1 No permission for attempted operation.

2 The command syntax was invalid. A usage message for the usermod command is displayed.

3 An invalid argument was provided to an option.

4 The *gid* or *uid* given with the -u option is already in use.

5 The password and shadow files are not consistent with each other. pwconv(1M) might be of use to correct possible errors. See passwd(4) and shadow(4).

6 The login to be modified does not exist, the *gid* or the *uid* does not exist.

7 The group, passwd, or shadow file is missing.

9 A group or user name is already in use.

10 Cannot update the passwd, shadow, or user_attr file.

11 Insufficient space to move the home directory (-m option).

12 Unable to create, remove, or move the new home directory.

13 Requested login is already in use.

14 Unexpected failure.

16 Unable to update the group database.

17 Unable to update the project database.

18 Insufficient authorization.

19 Does not have role.

20 Does not have profile.

21 Does not have privilege.

22 Does not have label.

23 Does not have group.

24 System not running Trusted Extensions.

25 Does not have project.

26 Unable to update `auto_home`.

Files

`/etc/group`
system file containing group definitions

`/etc/datemsk`
system file of date formats

`/etc/passwd`
system password file

`/etc/shadow`
system file containing users' encrypted passwords and related information

`/etc/user_attr`
system file containing additional user and role attributes

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os
Interface Stability	Committed

See Also `auths(1)`, `chown(1)`, `passwd(1)`, `profiles(1)`, `users(1B)`, `groupadd(1M)`, `groupdel(1M)`, `groupmod(1M)`, `logins(1M)`, `pwconv(1M)`, `roleadd(1M)`, `roledel(1M)`, `rolemod(1M)`, `useradd(1M)`, `userdel(1M)`, `getdate(3C)`, `auth_attr(4)`, `passwd(4)`, `policy.conf(4)`, `prof_attr(4)`, `user_attr(4)`, `attributes(5)`

Notes The `usermod` utility modifies definitions in the `passwd`, `shadow`, `group`, `project`, and `user_attr` databases in the scope (default or specified). The utility will verify the uniqueness of user name and user ID against the external name service.

The `usermod` utility uses the `/etc/datemsk` file, available with `SUNWaccr`, for date formatting.

Name utmpd – utmpx monitoring daemon

Synopsis

```
utmpd [-debug]
```

Description The `utmpd` daemon monitors the `/var/adm/utmpx` file. See `utmpx(4)` (and `utmp(4)` for historical information).

`utmpd` receives requests from `pututxline(3C)` by way of a named pipe. It maintains a table of processes and uses `poll(2)` on `/proc` files to detect process termination. When `utmpd` detects that a process has terminated, it checks that the process has removed its `utmpx` entry from `/var/adm/utmpx`. If the process' `utmpx` entry has not been removed, `utmpd` removes the entry. By periodically scanning the `/var/adm/utmpx` file, `utmpd` also monitors processes that are not in its table.

Options

`-debug`
: Run in debug mode, leaving the process connected to the controlling terminal. Write debugging information to standard output.

Exit Status The following exit values are returned:

`0`
: Successful completion.

`>0`
: An error occurred.

Files

`/etc/default/utmpd`
: You can set default values for the flags listed below. For example: `SCAN_PERIOD=600`

  `SCAN_PERIOD`
  : The number of seconds that `utmpd` sleeps between checks of `/proc` to see if monitored processes are still alive. The default is 300.

  `MAX_FDS`
  : The maximum number of processes that `utmpd` attempts to monitor. The default value is 4096.

  `WTMPX_UPDATE_FREQ`
  : The number of seconds that `utmpd` sleeps between read accesses of the `wtmpx` file. The `wtmpx` file's last access time is used by `init(1M)` on reboot to determine when the operating system became unavailable. The default is 60.

`/var/adm/utmpx`
: File containing user and accounting information for commands such as `who(1)`, `write(1)`, and `login(1)`.

`/proc`
: Directory containing files for processes whose `utmpx` entries are being monitored.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also svcs(1), init(1M), svcadm(1M), poll(2), pututxline(3C), proc(4), utmp(4), utmpx(4), attributes(5), smf(5)

Notes If the filesystem holding /var/adm/wtmpx is mounted with options which inhibit or defer access time updates, an unknown amount of error will be introduced into the utmp DOWN_TIME record's timestamp in the event of an uncontrolled shutdown (for example, a crash or loss of power). Controlled shutdowns will update the modify time of /var/adm/wtmpx, which will be used on the next boot to determine when the previous shutdown ocurred, regardless of access time deferral or inhibition.

The utmpd service is managed by the service management facility, smf(5), under the service identifier:

```
svc:/system/utmp:default
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using svcadm(1M). The service's status can be queried using the svcs(1) command.

Name uucheck – check the uucp directories and permissions file

Synopsis

```
/usr/lib/uucp/uucheck [-v] [-x debug-level]
```

Description `uucheck` checks for the presence of the `uucp` system required files and directories. `uucheck` also does error checking of the `Permissions` file (`/etc/uucp/Permissions`).

`uucheck` is executed during package installation. `uucheck` can only be used by the super-user or `uucp`.

Options The following options are supported:

`-v` Give a detailed (verbose) explanation of how the `uucp` programs will interpret the `Permissions` file.

`-x` *debug-level* Produce debugging output on the standard output. *debug-level* is a number from `0` to `9`. Higher numbers give more detailed debugging information.

Files

`/etc/uucp/Devices`

`/etc/uucp/Limits`

`/etc/uucp/Permissions`

`/etc/uucp/Systems`

`/var/spool/locks/*`

`/var/spool/uucp/*`

`/var/spool/uucppublic/*`

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	service/network/uucp

See Also `uucp(1C)`, `uustat(1C)`, `uux(1C)`, `uucico(1M)`, `uusched(1M)`, `attributes(5)`

Bugs The program does not check file/directory modes or some errors in the `Permissions` file such as duplicate login or machine name.

Name uucico – file transport program for the uucp system

Synopsis

```
/usr/lib/uucp/uucico [-f] [-c type] [-d spool-directory]
     [-i interface] [-r role-number] [-s system-name]
     [-x debug-level]
```

Description `uucico` is the file transport program for `uucp` work file transfers.

Options The following options are supported:

`-f` This option is used to "force execution" of `uucico` by ignoring the limit on the maximum number of uucicos defined in the `/etc/uucp/Limits` file.

`-c` *type* The first field in the `Devices` file is the "Type" field. The `-c` option forces `uucico` to only use entries in the "Type" field that match the user specified `type`. The specified `type` is usually the name of a local area network.

`-d` *spool-directory* This option specifies the directory *spool-directory* that contains the `uucp` work files to be transferred. The default spool directory is `/var/spool/uucp`.

`-i` *interface* This option defines the *interface* used with `uucico`. The interface only affects slave mode. Known interfaces are UNIX (default), TLI (basic Transport Layer Interface), and TLIS (Transport Layer Interface with Streams modules, read/write).

`-r` *role-number* The *role-number* `1` is used for master mode. *role-number* `0` is used for slave mode (default). When `uucico` is started by a program or `cron`, *role-number* `1` should be used for master mode.

`-s` *system-name* The `-s` option defines the remote system (*system-name*) that `uucico` will try to contact. It is required when the role is master; *system-name* must be defined in the `Systems` file.

`-x` *debug-level* Both `uux` and `uucp` queue jobs that will be transferred by `uucico`. These jobs are normally started by the `uusched` scheduler, for debugging purposes, and can be started manually. For example, the shell `Uutry` starts `uucico` with debugging turned on. The *debug-level* is a number between `0` and `9`. Higher numbers give more detailed debugging information.

Files `/etc/uucp/Devconfig`

`/etc/uucp/Devices`

`/etc/uucp/Limits`

`/etc/uucp/Permissions`

`/etc/uucp/Sysfiles`

`/etc/uucp/Systems`

`/var/spool/locks/*`

`/var/spool/uucp/*`

`/var/spool/uucppublic/*`

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	service/network/uucp

See Also `uucp(1C)`, `uustat(1C)`, `uux(1C)`, `Uutry(1M)`, `cron(1M)`, `uusched(1M)`, `attributes(5)`

Name uucleanup – uucp spool directory clean-up

Synopsis

```
/usr/lib/uucp/uucleanup [-Ctime] [-Dtime] [-mstring]
     [-otime] [-ssystem] [-Wtime] [-xdebug-level] [-Xtime]
```

Description `uucleanup` will scan the spool directories for old files and take appropriate action to remove them in a useful way:

- Inform the requester of send/receive requests for systems that can not be reached.
- Return undeliverable mail to the sender.
- Deliver `rnews` files addressed to the local system.
- Remove all other files.

In addition, there is a provision to warn users of requests that have been waiting for a given number of days (default 1 day). Note: `uucleanup` will process as if all option `times` were specified to the default values unless `time` is specifically set.

This program is typically started by the shell `uudemon.cleanup`, which should be started by `cron(1M)`.

Options

`-Ctime`	Remove any `C.` files greater or equal to `time` days old and send appropriate information to the requester (default 7 days).
`-Dtime`	Remove any `D.` files greater or equal to `time` days old, make an attempt to deliver mail messages, and execute `rnews` when appropriate (default 7 days).
`-m`*string*	Include *string* in the warning message generated by the `-W` option. The default line is "See your local administrator to locate the problem".
`-otime`	Delete other files whose age is more than `time` days (default 2 days).
`-s`*system*	Execute for *system* spool directory only.
`-Wtime`	Any `C.` files equal to `time` days old will cause a mail message to be sent to the requester warning about the delay in contacting the remote. The message includes the *JOBID*, and in the case of mail, the mail message. The administrator may include a message line telling whom to call to check the problem (`-m` option) (default 1 day).
`-x`*debug-level*	Produce debugging output on standard ouput. *debug-level* is a single digit between 0 and 9; higher numbers give more detailed debugging information. (This option may not be available on all systems.)
`-Xtime`	Any `X.` files greater or equal to `time` days old will be removed. The `D.` files are probably not present (if they were, the `X.` could get executed). But if there are `D.` files, they will be taken care of by D. processing (default 2 days).

Files

`/usr/lib/uucp`	directory with commands used by `uucleanup` internally
`/var/spool/uucp`	spool directory

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	service/network/uucp

See Also `uucp(1C)`, `uux(1C)`, `cron(1M)`, `attributes(5)`

Name uusched – uucp file transport program scheduler

Synopsis

```
/usr/lib/uucp/uusched [-u debug-level] [-x debug-level]
```

Description `uusched` is the `uucp(1C)` file transport scheduler. It is usually started by the daemon *uudemon.hour* that is started by `cron(1M)` from an entry in user `uucp`'s crontab file:

```
11,41 * * * * /etc/uucp/uucp/uudemon.hour
```

Options The options are for debugging purposes only. *debug-level* is a number between `0` and `9`. Higher numbers give more detailed debugging information:

The following options are supported:

`-u` *debug-level* Passes the `-u` *debug-level* option `uucico(1M)` as `-x` *debug-level*.

`-x` *debug-level* Outputs debugging messages from `uusched`.

Files

```
/etc/uucp/Devices

/etc/uucp/Permissions

/etc/uucp/Systems

/var/spool/locks/*

/var/spool/uucp/*

/var/spool/uucppublic/*
```

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	service/network/uucp

See Also `uucp(1C)`, `uustat(1C)`, `uux(1C)`, `cron(1M)`, `uucico(1M)`, `attributes(5)`

Name Uutry, uutry – attempt to contact remote system with debugging on

Synopsis

```
/usr/lib/uucp/Uutry [-r] [-c type] [-x debug-level] system-name
```

Description `Uutry` is a shell script that is used to invoke `uucico`(1M) to call a remote site. Debugging is initially turned on and is set to the default value of `5`. The debugging output is put in file `/tmp/`*system-name*.

Options The following options are supported:

`-r` This option overrides the retry time that is set in file `/var/uucp/.Status/`*system-name*.

`-c` *type* The first field in the `Devices` file is the "Type" field. The `-c` option forces `uucico` to use only entries in the "Type" field that match the user-specified `type`. The specified `type` is usually the name of a local area network.

`-x` *debug-level* *debug-level* is a number from `0` to `9`. Higher numbers give more detailed debugging information.

Files

`/etc/uucp/Devices`

`/etc/uucp/Limits`

`/etc/uucp/Permissions`

`/etc/uucp/Systems`

`/tmp/`*system-name*

`/var/spool/locks/*`

`/var/spool/uucp/*`

`/var/spool/uucppublic/*`

Attributes See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	service/network/uucp

See Also `uucp`(1C), `uux`(1C), `uucico`(1M), `attributes`(5)

Name uuxqt – execute remote command requests

Synopsis `/usr/lib/uucp/uuxqt [-s system] [-x debug-level]`

Description `uuxqt` is the program that executes remote job requests from remote systems generated by the use of the `uux` command. (`mail` uses `uux` for remote mail requests). `uuxqt` searches the spool directories looking for execution requests. For each request, `uuxqt` checks to see if all the required data files are available, accessible, and the requested commands are permitted for the requesting system. The `Permissions` file is used to validate file accessibility and command execution permission.

There are two environment variables that are set before the `uuxqt` command is executed:

- `UU_MACHINE` is the machine that sent the job (the previous one).
- `UU_USER` is the user that sent the job.

These can be used in writing commands that remote systems can execute to provide information, auditing, or restrictions.

Options The following options are supported:

`-s` *system* Specifies the remote *system* name.

`-x` *debug-level* *debug-level* is a number from `0` to `9`. Higher numbers give more detailed debugging information.

Files `/etc/uucp/Limits`

`/etc/uucp/Permissions`

`/var/spool/locks/*`

`/var/spool/uucp/*`

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	service/network/uucp

See Also `mail(1)`, `uucp(1C)`, `uustat(1C)`, `uux(1C)`, `uucico(1M)`, `attributes(5)`

Name vbiosd – vbios daemon

Synopsis

```
vbiosd
```

Description The `vbiosd` daemon is started at boot time to provide an execution environment for adapter-supplied VBIOS code that is used in graphics mode switching.

This mode switching occurs when exiting X, switching virtual terminals (see `vtdaemon(1M)` for more information), and during fast reboot.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Uncommitted

See Also `svcs(1)`, `svcadm(1M)`, `vtdaemon(1M)`, `attributes(5)`, `smf(5)`

Notes The `vbios` daemon is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/system/vbiosd:default
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. The service's status can be queried using the `svcs(1)` command.

Name vdiskadm – create and manage virtual disks

Synopsis

```
vdiskadm create -s size [-t type[:opt],[opt]
     [-c comment] vdname

vdiskadm destroy [-r] vdname|snapshot

vdiskadm snapshot vdname@snapname

vdiskadm rollback [-r] snapshot

vdiskadm clone [-c comment] vdname|snapshot clone_vdname

vdiskadm move vdname dir

vdiskadm rename vdname|snapshot vdname|snapshot

vdiskadm list [-fp]vdname

vdiskadm verify vdname

vdiskadm prop-get [-l] -p property vdname

vdiskadm prop-set -p property=value vdname

vdiskadm prop-add -p property=value vdname

vdiskadm prop-del -p property vdname

vdiskadm import [-fnpqm] [-x type] -d file|zvol|dsk
     [-t type[:opt]] vdname

vdiskadm export -x type[:opt] -d file|zvol|dsk vdname

vdiskadm convert [-t type[:opt]] vdname

vdiskadm translate [-i type[:opt]] -I input_file -x type[:opt]
     -d output_file

vdiskadm help [command]
```

Description The `vdiskadm` command manages virtual disks within `dom0`. In the SYNOPSIS above, *vdname* is the pathname of the virtual disk; it has a maximum length of `MAXPATHLEN` (1024 bytes).

`vdiskadm` is implemented as a set of subcommands, many with their own options and operands. These subcommands are described under "Subcommands," below.

The following subsections describe concepts related to virtual disks.

Snapshots A snapshot is a read-only copy of a virtual disk. Snapshots can be created extremely quickly and initially consume little space. As data within the active virtual disk changes, the snapshot consumes more data than would otherwise be shared with the active virtual disk.

Clones A clone is a writable copy of a virtual disk. The default type of clone is a merged (that is, coalesced) copy of the original virtual disk. An example of a merged clone occurs when a virtual disk is comprised of several snapshots; a subsequent clone operation results in a new

virtual disk containing no snapshots. A clone will be of the same type as the original virtual disk (that is, `vmdk:fixed`). When a merged clone is created there is no linkage back to the original virtual disk or to any of its snapshots. This lack of linkage allows the merged clone to be moved to another physical machine.

Numeric Values The values of numeric properties can be specified using human-readable suffixes, such as `k`, `KB`, `M`, `Gb`, and so forth, up to `Z` for zettabyte). The following are all valid (and equal) specifications:

```
1536M  1.5g  1.50GB
```

Types of Virtual Disks The following types and options of virtual disks are supported:

- `vmdk:fixed`
- `vmdk:sparse`
- `vdi:fixed`
- `vdi:sparse`
- `vhd:fixed`
- `vhd:sparse`
- `raw:fixed`

where `vmdk` is the native VMware format, `vdi` is the native VirtualBox format, `vhd` is the native Microsoft format, and `raw` describes a file that looks like a raw disk. A `raw` disk is always in fixed format so that option can be explicitly set or implicitly understood.

If the type is not specified, the default value is `vmdk`. If the option is not specified, the default value is `fixed` for type `raw` and `sparse` for types `vmdk`, `vdi`, and `vhd`.

Native and User-defined Properties Properties are divided into two types, native and user defined. Native properties either export internal statistics or control `vdiskadm` behavior. In addition, native properties are either editable or read-only. User–defined properties are arbitrary strings that have no effect on `vdiskadm` behavior. You can use them to annotate virtual disks in a way that is meaningful in your environment. User–defined property names must contain a colon (`:`) character, to distinguish them from native properties.

Properties are associated only with the virtual disk and not with individual snapshots.

Every virtual disk has a set of native properties that export statistics about the virtual disk, as well as control various behaviors.

The following are the native properties for a virtual disk:

`cdrom`
: Boolean property that is true if the virtual disk is a CDROM.

`removable`
: Boolean property that is true if the virtual disk is a removable media.

`readonly`
: Boolean property that is true if the virtual disk is read-only. This property is read-only.

`sectors`
Numeric property containing the number of disk sectors in the given virtual disk. This property is read-only.

`name`
String property that is the name of the virtual disk. This property is read-only.

`max-size`
Numeric property containing the maximum size of the virtual disk in bytes. This property is read-only.

`effective-size`
Numeric property containing the effective size of the virtual disk, in bytes. The effective size includes the size of the data file and all snapshots. The effective size can exceed the maximum size. This property is read-only.

`creation-time`
String property containing the date and time that the virtual disk was created. This property is read-only.

`creation-time-epoch`
Numeric property describing the creation-time property in seconds since the epoch (seconds since 00:00:00 UTC, Jan. 1, 1970). This property is read-only.

`modification-time`
String property containing the date and time of last modification to virtual disk. This property is read-only.

`modification-time-epoch`
Numeric property describing the modification-time property in seconds since the epoch (seconds since 00:00:00 UTC, Jan. 1, 1970). This property is read-only.

`description`
String property that contains the comment given when the virtual disk was created or cloned. This property is read-only.

`type`
String property that contains the type of virtual disk: `vmdk`, `vhd`, `vdi`, or `raw`. This property is read-only.

`sparse`
Boolean property that is true if the virtual disk is in `sparse` format. This property is read-only.

`owner`
String property that contains the user name of the owner of the virtual disk. This property is editable.

Sub-commands The `vdiskadm` subcommands and their arguments are described in the following subsections.

`vdiskadm create`

```
vdiskadm create -s size [-t type[:opt],[opt]]
     [-c comment] vdname
```

Creates a new virtual disk of the specified size and at the location specified by *vdname*. If *vdname* includes a path to the virtual disk, the directories that follow from that path will be created during creation of the virtual disk. This subcommand has the options listed below.

`-t` *type*[:*opt*],[*opt*]]
Specifies the type of virtual disk to be created. The default type is `vmdk`. For `vmdk` and `vdi` types the default option is `sparse`. For type `raw` the default option is `fixed`.

`-c` *comment*
Comment that can be attached to virtual disk.

`vdiskadm destroy`

```
vdiskadm destroy [-r] vdname|snapshot
```

Destroys the specified virtual disk or snapshot. By default, the destroy operation fails if the specified virtual disk contains snapshots. This subcommand has the option listed below.

`-r`
Recursively destroys the virtual disk, including all snapshots associated with the virtual disk.

`vdiskadm snapshot`

```
vdiskadm snapshot vdname@snapname
```

Creates a snapshot of the virtual disk with the specified *snapname*. This subcommand has no options.

`vdiskadm rollback`

```
vdiskadm rollback [-r] snapshot
```

Roll back the virtual disk to a previous snapshot. When a virtual disk is rolled back, all data that has changed since the snapshot is discarded, and the virtual disk reverts to the state at the time of the snapshot. By default, the command refuses to roll back to a snapshot other than the most recent one. In order to roll back further, all intermediate snapshots must be destroyed by specifying the `-r` option. This subcommand has the option listed below.

`-r`
Recursively destroy any snapshots more recent than the one specified.

`vdiskadm clone`

```
vdiskadm [-c comment] vdname|snapshot clone_vdname
```

Creates a clone of the specified snapshot or virtual disk. The clone is created with the type and option and the size of the virtual disk being cloned. If *clone_vdname* includes a path the directories that flow from that path will be created during creation of the cloned virtual disk. By default, a merged clone image is created. This subcommand has the option listed below.

`-c` *comment*
Comment that can be attached to cloned virtual disk.

`vdiskadm move` `vdiskadm move` *vdname* *dir*

Moves a specified virtual disk into the specified directory. The virtual disk maintains the same name. The new directory must exist. This subcommand has no options.

`vdiskadm rename` `vdiskadm rename` *vdname*|*snapshot* *vdname*|*snapshot*

Renames a virtual disk or snapshot. This subcommand has no options.

`vdiskadm list` `vdiskadm list [-fp]`*vdname*

Lists a specified virtual disk and its snapshots. This subcommand has the options listed below.

`-f`
Gives a list of all files associated with the virtual disk. This list includes the store file and the extents.

`-p`
Lists the files in an easily parsable format, prefixing the files with a label of `file:`, `snapshot:`, or `store`.

`vdiskadm verify` `vdiskadm verify` *vdname*

Returns an error if the virtual disk cannot be recognized or opened by Solaris. This subcommand has no options.

`vdiskadm prop-get` `vdiskadm prop-get [-l] -p` *property* *vdname*

Returns the value of the property for the specified virtual disk. A property value of `all` displays all native and user-defined properties for the virtual disk. This subcommand has the options listed below.

`-l`
Gives additional property information, such as the writeable status of property.

`-p` *property*
Specifies the property being queried and displays the value of the property. For the property `all`, the name of the property, a colon, and a space are displayed before the value of the property.

`vdiskadm prop-set` `vdiskadm prop-set -p` *property*=*value* *vdname*

Sets the value of the specified property for the specified virtual disk. *property* can be a native or a user-defined property, but must be writable. Can be used to change the value of a property added with the `prop-add` subcommand. This subcommand has the option listed below.

`-p` *property*=*value*
Specifies the property being set.

`vdiskadm prop-add` `vdiskadm -p` *property=value vdname*

Adds the user-defined property with the specified value to the specified virtual disk. Returns an error if the property already exists. The user-defined property name must contain a colon character (`:`). This subcommand has the option listed below.

`-p` *property=value*
Specifies the property being added.

`vdiskadm prop-del` `vdiskadm prop-del -p` *property vdname*

Deletes a user–defined property from the specified virtual disk. This subcommand has the option listed below.

`-p` *property*
Specifies the property being deleted.

`vdiskadm import` `vdiskadm import [-fnpqm] [-x` *type*`] -d file|zvol|dsk \`
`[-t` *type*`[:`*opt*`]] `*vdname*

Creates a new virtual disk using data from a file or block device. The file may be in `vdi`, `vhd`, `vmdk`, or `raw` format. A block device is always assumed to be in raw format. This subcommand has the following options.

`-f`
Returns a list of files that will be used in the import process.

`-n`
Show output from import without actually running the import.

`-p`
Displays files in an easily parsable format.

`-q`
Run in quiet mode giving no output.

`-m`
Move the imported file to virtual disk without copying the data.

`-x` *type*
Specifies the type of virtual disk data being imported. If `vdiskadm` is unable to detect the imported file type, `-x` must be specified.

`-d file|zvol|dsk`
File or block device containing data to be imported.

`-t` *type*[:*opt*]
Specifies the type of virtual disk to be created on import. The default type is `vmdk`. For `vmdk`, `vdi`, and `vhd` types the default *opt* is `sparse`. For type `raw` the default *opt* is `fixed`.

`vdiskadm export` `vdiskadm export -x` *type*`[:`*opt*`] -d file|zvol|dsk` *vdname*

Exports data from a virtual disk to a file or block device. This subcommand has the following options.

`-x` *type*[:*opt*]
Specifies the type of virtual disk data being exported.

`-d file|zvol|dsk`
File or block device receiving data being exported.

`vdiskadm convert` `vdiskadm convert [-t` *type*`[:`*opt*`]]` *vdname*

Converts a virtual disk into a different type virtual disk. This subcommand has the following option.

`-t` *type*[:*opt*]
Specifies the type of virtual disk to be created upon conversion. The default type is `vmdk`. For `vmdk`, `vdi`, and `vhd` types the default *opt* is `sparse`. For type `raw` the default *opt* is `fixed`.

`vdiskadm translate` `vdiskadm translate [-i` *type*`[:`*opt*`]] -I` *input_file* `\`
`-x` *type*`[:`*opt*`] -d` *output_file*

Translate data from one virtual disk data type to another without creating a virtual disk. This subcommand has the following options.

`-i` *type*[:*opt*]
Specifies the input type of virtual disk data being translated. If `vdiskadm` is unable to detect the input file type, `-i` must be specified.

`-I` *input_file*
File or block device containing data being translated.

`-x` *type*[:*opt*]
Specifies the output type of virtual disk data being translated. For `vmdk`, `vdi`, and `vhd` types the default *opt* is `sparse`. For type `raw` the default *opt* is `fixed`.

`-d` *output_file*
File or block device receiving data being translated. *output_file* must not exist. The file will be created during translation.

`vdiskadm help` `vdiskadm help [`*command*`]`

Displays a general or command-specific help message. This subcommand has only the command name optional argument.

Examples

EXAMPLE 1 Creating a vmdk Sparse File

The following command creates a virtual disk named `disk1` of size 8 GB in the directory `/guests/disks`.

```
# vdiskadm create -s 8g -t vmdk:sparse /guests/disks/disk1
```

EXAMPLE 2 Creating a Snapshot

The following command creates a snapshot of the virtual disk located at `/guests/disks/disk1`. The snapshot is named `install`.

```
# vdiskadm snapshot /guests/disks/disk1@install
```

EXAMPLE 3 Creating and Destroying Snapshots

The following commands create two snapshots, named `install` and `bfu`, of the virtual disk located at `/guests/disks/disk1`. The third command destroys the newly created snapshot `install`.

```
# vdiskadm snapshot /guests/disks/disk1@install
# vdiskadm snapshot /guests/disks/disk1@bfu
# vdiskadm destroy /guests/disks/disk1@install
```

EXAMPLE 4 Rolling Back a Virtual Disk

The following command reverts the contents of the virtual disk to the snapshot named `install`, deleting all intermediate snapshots.

```
# vdiskadm rollback -r /guests/disks/disk1@install
```

EXAMPLE 5 Listing a Virtual Disk and Snapshots

The following command lists all of the images associated with the virtual disk `/guests/disks/disk1`.

```
# vdiskadm list /guests/disks/disk1
disk1@install
disk1@bfu
disk1
```

EXAMPLE 6 Creating a Clone

The following command creates a new virtual disk that is a coalesced copy of the virtual disk `/guests/disks/disk1`. The clone is created in the same format (that is, `vmdk:sparse`) as the original virtual disk.

```
# vdiskadm clone /guests/disks/disk1 /guests/clone/clone_disk1
```

EXAMPLE 7 Adding a User-defined Property

The following command adds a user-defined property to the virtual disk and assigns it the specified value. This property name was chosen to represent the source and requirements of this virtual disk data using the required colon to delineate the fields.

```
# vdiskadm prop-add -p com.sun:required-nic=2 /guests/disks/disk1
```

EXAMPLE 8 Importing Existing vmdk Format File

The following command takes an existing `vmdk` format file and imports it to a virtual disk.

```
# vdiskadm import -d /downloads/appliance.vmdk /guests/import/disk1
```

EXAMPLE 9 Importing vmdk File and Converting to vhd

The following command takes an existing `vmdk` format file and, upon import, converts it to a `vhd`-type virtual disk.

```
# vdiskadm import -d /downloads/appliance.vmdk -t vhd \
/guests/import/disk1
```

EXAMPLE 10 Importing Data from zvol

The following command imports virtual disk data from a `zvol` and, upon import, converts it to a `vmdk:fixed` type virtual disk.

```
# vdiskadm import -d /dev/zvol/dsk/pool1/disk1 -t vmdk:fixed \
/guests/import/disk1
```

EXAMPLE 11 Exporting Data to Block Device

The following command takes an existing virtual disk and, upon export, converts it to a disk slice, of `raw` type.

```
# vdiskadm export -d /dev/dsk/c0t1d0s3 -x raw /guests/disks/disk1
```

EXAMPLE 12 Converting Virtual Disk Type

The following command takes an existing virtual disk and converts it (in place) to a different format type.

```
# vdiskadm convert -t vdi:fixed /guests/disks/disk1
```

EXAMPLE 13 Translating Data from One Type to Another

The following command translates data from a virtual disk format file to raw data written to a `zvol` without creating a virtual disk.

```
# vdiskadm translate -I /downloads/appliance.vmdk -x raw \
-d /dev/zvol/dsk/pool1/disk1
```

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/storage/vdiskadm
Interface Stability	Uncommitted

See Also attributes(5)

Name vdpd – VSI Discovery and Configuration Protocol daemon

Synopsis

```
/usr/lib/vdpd

svc:/network/evb/vdp:default
```

Description vdpd is a system daemon that exchanges VSI (Virtual Station Interface or VNIC) information using VDP and ECP protocols, defined in the IEEE 802.1qbg specification. The vdpd daemon is controlled through the service management facility (SMF, see smf(5)) service instance:

```
svc:/network/evb/vdp:default
```

This means that svcadm(1M) must be used to start, stop, restart, and refresh the vdpd daemon. The daemon is enabled by default. To enable VDP exchanges for any VNIC created on the system, this daemon should be running.

The vdpd daemon has no options. Note that it is a Project Private interface.

Examples **EXAMPLE 1** Enabling the vdpd Daemon

The following svcadm command enables the vdpd daemon.

```
# svcadm enable network/vdpd
# svcs network/vdpd

STATE STIME FMRI
online 1:32:31 svc:/network/evb/vdp:default
```

EXAMPLE 2 Disabling the vdpd Daemon

The following svcadm command disables the vdpd daemon.

```
# svcadm disable network/vdpd
# svcs network/vdpd

STATE STIME FMRI
disabled 1:33:41 svc:/network/evb/vdp:default
```

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	service/network/evb
Interface Stability	Project Private

See Also dladm(1M), lldpadm(1M), svcadm(1M), attributes(5), smf(5)

IEEE Std 802.1 Qbg/D1.6: *Draft Standard for Local and Metropolitan Area*

Networks-Virtual Bridged Local Area Networks; Amendment XX : *Edge Virtual Bridging*.

Name virt-convert – convert virtual machines between formats

Synopsis

```
virt-convert [option]... input.vmx | input.ovf | input-dir
    [output.xml | output-dir]
```

Description The `virt-convert` program is a command line tool for converting virtual machines (VMs) from one format to another. It accepts either a VM definition file (such as VMware `vmx` format) or a directory containing a VM. By default, a new VM definition file, and converted disk images, will be placed in a new output directory.

If an output directory is specified, it will be created if necessary, and the output VM definition placed within the new directory, along with any disk images, as needed.

If an output VM definition file is specified, it will be created alongside any disks in the same directory.

Options Any of the following options can be omitted, in which case `virt-convert` will use defaults when required. An input VM definition or containing directory must be provided. By default, an output directory is generated based upon the name of the VM. The default input format is VMware `vmx`, and the default output format is a `libvirt` "image" XML definition.

`-a`, `--arch`=*arch*
Architecture of the virtual machine (`i686`, `x86_64`, `ppc`). Defaults to that of the host machine.

`-D`, `--disk-format`=*format*
Output disk format, or `none` if no conversion should be performed. *format* is one of:

`none`
No disks are converted or copied.

`vmdk`
VMWare VMDK format

`raw`
raw file

`vdisk`
`vdisk` format (see `vdiskadm(1M)`)

`-d`, `--debug`
Display debugging information.

`-h`, `--help`
Display the help message and exit.

`-i`, `--input-format` *format*
Input format. Currently, `vmx` and `ovf` are supported.

`--noacpi`
Override the OS type and variant to disable the ACPI setting for fully virtualized guest.

`--noapic`
Override the OS type and variant to disable the APIC setting for fully virtualized guest.

`-o`, `--output-format` *format*
Output format. Currently, the supported output formats are `virt-instance` and `virt-image`. `virt-instance` is the recommended format for Solaris.

`--os-type=`*os_type*
Optimize the guest configuration for a type of operating system. This will attempt to pick the most suitable ACPI and APIC settings, optimally supported mouse drivers, and generally accommodate other operating system quirks.

`--os-variant=`*os_variant*
Further optimize the guest configuration for a specific operating system variant. This parameter is optional.

`-p`, `--paravirt`
Create a paravirtualized guest image. Convert machine to a paravirtualized Xen-based image.

`-q`, `--quiet`
Avoid verbose output.

`-v`, `--hvm`
Create a fully virtualized guest image. Convert machine to a `hvm`/`qemu`–based image (this is the default if `--paravirt` is not specified).

Examples

EXAMPLE 1 Converting a VMware VMX appliance

The following sequence of commands converts a VMware VMX appliance and imports it into Solaris xVM.

```
# virt-convert -o virt-instance /guests/vmx-appliance/ \
/guests/xvm-appliance/
# virsh define --relative-path=/guests/xvm-appliance/ \
\guests/xvm-appliance/appliance.xml
```

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/xvm/header-xvm
Interface Stability	Volatile

See Also

`vdiskadm(1M)`, `attributes(5)`

Caveats Not all conversions will result in a working guest installation. If the source OS image is configured to use SCSI disks, the use of IDE disks may cause the OS boot to fail. Some images may be configured to use the VMware drivers such as `vlance`. In the `vlance` case and in general, device emulation support may not be sufficient for all OS installations.

Authors Written by Joey Boggs and John Levon.

See the `AUTHORS` file in the source distribution for the complete list of credits.

Name virtinfo – virtualization domain information

Synopsis

```
virtinfo [ -acdpstu ]
```

Description The `virtinfo` utility provides virtualization information returned by `libv12n(3LIB)` about the current domain. When options are specified, symbols that represent one or more virtualization domain characteristics are written to standard output. If no options are specified, `virtinfo` shows only the virtualization domain type.

Options The `virtinfo` command supports the following options:

`-a` Shows all information that is currently available from the system.

`-c` Shows the network node name of the control domain. This name is identical to the string shown by running the `uname -n` command on the control domain.

`-d` Shows the domain name for this virtual domain.

`-p` Shows information in a parseable format.

The parseable output for Logical Domains (LDoms) has the following format:

```
VERSION 1.0
DOMAINROLE|impl=LDoms|control={true|false}|
    io={true|false}|service={true|false}|
    root={true|false}
DOMAINNAME|name=domain-name
DOMAINUUID|uuid=uuid
DOMAINCONTROL|name=control-domain-nodename
DOMAINCHASSIS|serialno=serial-no
```

`-s` Shows the platform serial number.

`-t` Shows the domain type.

The domain type for Logical Domains is a string of one or more of the following blank-separated values:

`LDoms`	LDoms implementation
`control`	Control domain
`guest`	Guest domain
`I/O`	I/O domain
`service`	Service domain
`root`	Root I/O domain

Note that the `control` and `guest` values are mutually exclusive.

`-u` Shows the universally unique identifier (UUID) of the domain. See the `libuuid(3LIB)` man page.

Exit Status The following exit values are returned:

0 Successful completion.

>0 An error occurred.

Examples **EXAMPLE 1** Viewing Default Information

The following example shows the default information about the domain:

```
$ virtinfo
Domain role: LDoms control I/O service root

The current domain is the control domain, which is also an
I/O domain, the service domain and a root I/O domain.
```

EXAMPLE 2 Viewing All Information

The following example shows all available information about the domain:

```
$ virtinfo -a
Domain role: LDoms control I/O service root
Domain name: primary
Domain UUID: 8e0d6ec5-cd55-e57f-ae9f-b4cc050999a4
Control domain: san-t2k-6
Chassis serial#: 0704RB0280
```

EXAMPLE 3 Viewing All Information In a Parsable Format

The following example shows all available information about the domain in a parsable format:

```
$ virtinfo -ap
VERSION 1.0
DOMAINROLE|impl=LDoms|control=true|io=true|service=true|root=true
DOMAINNAME|name=primary
DOMAINUUID|uuid=8e0d6ec5-cd55-e57f-ae9f-b4cc050999a4
DOMAINCONTROL|name=san-t2k-6
DOMAINCHASSIS|serialno=0704RB0280
```

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os
Interface Stability	Uncommitted

See Also libuuid(3LIB), libv12n(3LIB)

Name vmstat – report virtual memory statistics

Synopsis

```
vmstat [-ipqsS] [-T u | d] [disks] [interval [count]]
```

Description `vmstat` reports virtual memory statistics regarding kernel thread, virtual memory, disk, trap, and CPU activity.

On MP (multi-processor) systems, `vmstat` averages the number of CPUs into the output. For per-processor statistics, see `mpstat(1M)`.

`vmstat` only supports statistics for certain devices. For more general system statistics, use `sar(1)`, `iostat(1M)`, or `sar(1M)`.

Without options, `vmstat` displays a one-line summary of the virtual memory activity since the system was booted.

During execution of the kernel status command, the `state` of the system can change. If relevant, a state change message is included in the `vmstat` output, in one of the following forms:

```
<<device added: sd0>>
<<device removed: sd0>>
<<processors added: 1, 3>>
<<processors removed: 1, 3>>
```

See *Oracle Solaris Administration: Common Tasks* for device naming conventions for disks.

Options The following options are supported:

`-i` Report the number of interrupts per device. *count* and *interval* does not apply to the `-i` option.

`-p` Report paging activity in details. This option will display the following, respectively:

- epi Executable page-ins.
- epo Executable page-outs.
- epf Executable page-frees.
- api Anonymous page-ins.
- apo Anonymous page-outs.
- apf Anonymous page-frees.
- fpi File system page-ins.
- fpo File system page-outs.
- fpf File system page-frees.

When executed in a zone and if the pools facility is active, all of the above only report actitivity on the processors in the processor set of the zone's pool.

`-q` Suppress messages related to state changes.

`-s` Display the total number of various system events since boot. *count* and *interval* does not apply to the `-s` option.

`-S` Report on swapping rather than paging activity. This option will change two fields in `vmstat`'s "paging" display: rather than the "re" and "mf" fields, `vmstat` will report "si" (swap-ins) and "so" (swap-outs).

`-T u | d` Specify `u` for a printed representation of the internal representation of time. See `time(2)`. Specify `d` for standard date format. See `date(1)`.

Operands The following operands are supported:

count Specifies the number of times that the statistics are repeated. *count* does not apply to the `-i` and `-s` options.

disks Specifies which disks are to be given priority in the output (only four disks fit on a line). Common disk names are `id`, `sd`, `xd`, or `xy,` followed by a number (for example, `sd2`, `xd0`, and so forth).

interval Specifies the last number of seconds over which `vmstat` summarizes activity. This number of seconds repeats forever. *interval* does not apply to the `-i` and `-s` options.

Examples **EXAMPLE 1** Using `vmstat`

The following command displays a summary of what the system is doing every five seconds.

```
example% vmstat 5


kthr   memory          page             disk      faults        cpu
r b w swap  free re mf pi p fr de sr s0 s1 s2 s3  in  sy  cs us sy id
0 0 0 11456 4120 1  41 19 1  3  0  2  0  4  0  0  48 112 130  4 14 82
0 0 1 10132 4280 0   4 44 0  0  0  0  0 23  0  0 211 230 144  3 35 62
0 0 1 10132 4616 0   0 20 0  0  0  0  0 19  0  0 150 172 146  3 33 64
0 0 1 10132 5292 0   0  9 0  0  0  0  0 21  0  0 165 105 130  1 21 78
1 1 1 10132 5496 0   0  5 0  0  0  0  0 23  0  0 183  92 134  1 20 79
1 0 1 10132 5564 0   0 25 0  0  0  0  0 18  0  0 131 231 116  4 34 62
1 0 1 10124 5412 0   0 37 0  0  0  0  0 22  0  0 166 179 118  1 33 67
1 0 1 10124 5236 0   0 24 0  0  0  0  0 14  0  0 109 243 113  4 56 39
^C

example%
```

The fields of `vmstat`'s display are

EXAMPLE 1 Using `vmstat` *(Continued)*

`kthr` Report the number of kernel threads in each of the three following states:

- `r` the number of kernel threads in run queue
- `b` the number of blocked kernel threads that are waiting for resources I/O, paging, and so forth
- `w` the number of swapped out lightweight processes (LWPs) that are waiting for processing resources to finish.

`memory` Report on usage of virtual and real memory.

- `swap` available swap space (Kbytes)
- `free` size of the free list (Kbytes)

`page` Report information about page faults and paging activity. The information on each of the following activities is given in units per second.

- `re` page reclaims — but see the `-S` option for how this field is modified.
- `mf` minor faults — but see the `-S` option for how this field is modified.
- `pi` kilobytes paged in
- `po` kilobytes paged out
- `fr` kilobytes freed
- `de` anticipated short-term memory shortfall (Kbytes)
- `sr` pages scanned by clock algorithm

When executed in a `zone` and if the pools facility is active, all of the above (except for "de") only report activity on the processors in the processor set of the `zone`'s pool.

`disk` Report the number of disk operations per second. There are slots for up to four disks, labeled with a single letter and number. The letter indicates the type of disk (s = SCSI, i = IPI, and so forth); the number is the logical unit number.

`faults` Report the trap/interrupt rates (per second).

- `in` interrupts
- `sy` system calls
- `cs` CPU context switches

When executed in a `zone` and if the pools facility is active, all of the above only report actitivity on the processors in the processor set of the `zone`'s pool.

EXAMPLE 1 Using `vmstat` *(Continued)*

`cpu` Give a breakdown of percentage usage of CPU time. On MP systems, this is an average across all processors.

`us` user time

`sy` system time

`id` idle time

When executed in a `zone` and if the pools facility is active, all of the above only report actitivity on the processors in the processor set of the `zone`'s pool.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os
Interface Stability	See below.

Invocation is evolving. Human readable output is unstable.

See Also `date(1)`, `sar(1)`, `iostat(1M)`, `mpstat(1M)`, `sar(1M)`, `time(2)`, `attributes(5)`

Notes The sum of CPU utilization might vary slightly from 100 because of rounding errors in the production of a percentage figure.

Name vntsd – virtual network terminal server daemon for Logical Domains

Synopsis `/usr/lib/ldoms/vntsd`

Description The `vntsd` daemon is a server that supports connections to the Logical Domains (LDoms) console by using `telnet(1)`. When a `telnet` session starts, `vntsd` sends `telnet` options to the client indicating a willingness to remotely echo characters and to suppress go ahead.

Consoles are organized into groups by the LDoms Manager. Each console group is assigned a unique group name and TCP port number. `vntsd` uses the group's port number to export access to the consoles within that group. To establish a connection with a console or console group, a user starts a `telnet(1)` session with the corresponding group's port number. Depending on the number of consoles within that group, `vntsd` does one of two things:

- If there is only one console in the group, `vntsd` connects a session to that LDoms console.
- If there are multiple consoles in the group, `vntsd` prompts the user to select the console to which they would like to connect, as shown in "Multiple-Console Options," below.

For each console, `vntsd` provides write access only to the first user connecting to the console. Subsequent users connecting to the console are allowed only to read from the console and wait for write access. When the first user disconnects, write privileges are transferred to the next user waiting in the queue. If a user who does not have write privileges attempts to write to a console, the `vntsd` displays the following message:

```
You do not have write access
```

A user who has no write access can acquire write access forcibly by using the `~w` special console command, described in "Special Console Commands," below.

`vntsd` can be invoked only with superuser privileges or by someone fitting the appropriate security profile.

Options The options for `vntsd` are divided into multiple-console options and console commands.

Multiple-Console Options

The options listed below are supported when there are multiple LDoms consoles in a group. The syntax for the use of these options is:

<hostname>`-vnts-`*<group-name>*`: `*<option>*

For example:

```
myhost-vnts-salesgroup: h
```

The `h` option invokes help, as described below.

`h`

Display the following help text:

```
h -- this help
l -- list of consoles
q -- quit
c{id}, n{name} -- connect to console of domain {id} or domain name
```

`l`
List all consoles in the group. For example:

```
DOMAIN ID       DOMAIN NAME     DOMAIN STATE
0               ldg1            online
1               ldg2            connected
...             ...             ...
```

The two domain states and their meanings are:

`online`
No one is connected to the console.

`connected`
At least one user is already connected to the console.

`q`
Disconnect from `vntsd`.

`c`*{id}*`, n`*{name}*
Connect to specified console. Upon connection, the following message is displayed:

```
Connecting to console <domain-name> in group <group-name>
Press ~? for control options ....
```

Special Console Commands

A tilde (~) appearing as the first character of a line is an escape signal that directs `vntsd` to perform a special console command. The tilde-tilde (~~) sequence outputs a tilde. In conjunction with the initial tilde, `vntsd` accepts the following special console commands:

`~.`
Disconnect from the console or console group.

`~w`
Force write access to the console.

`~p`
Disconnect from this console, and connect to the console that precedes this console in the list of consoles.

`~n`
Disconnect from this console, and connect to the console that follows this console in the list of consoles.

`~#`
Send break.

`~^B`
Send alternate break.

`~?`
Display `vntsd` help, as follows:

```
~# - Send break
~^B - Send alternate break
~. - Exit from this console
~w - Force write access
~n - Console next
~p - Console previous
~? - Help
```

Files `/usr/lib/ldoms/vntsd`
Binary executable `vntsd` file.

`/usr/lib/ldoms/vntsd.xml`
Service management facility (`smf(5)`) manifest file for `vntsd`.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/ldoms
Interface Stability	Committed

See Also `telnet(1)`, `svccfg(1M)`, `usermod(1M)`, `auth_attr(4)`, `attributes(5)`, `smf(5)`

Notes The `vntsd` is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/ldoms/vntsd
```

You can change the following properties using the `svccfg(1M)` command:

`vntsd/vcc_device`
Set an instance of the virtual console concentrator (`vcc`) driver to which `vntsd` is connected.

`vntsd/listen_addr`
Set the IP address to which `vntsd` listens, using the following syntax:

`vntsd/listen_addr:"`*xxx.xxx.xxx.xxx*`"`

...where *xxx.xxx.xxx.xxx* is a valid IP address. The default value of this property is to listen on IP address 127.0.0.1. Users can connect to a guest console over a network if the value is set to the IP address of the control domain.

Note – Enabling network access to a console has security implications. Any user can connect to a console and for this reason it is disabled by default.

`vntsd/timeout_minutes`
Set timeout in minutes. `vntsd` will timeout (close) `telnet` connection if there is no activity (input or output) on the console. The default value is 0, which disables timeout.

`vntsd/authorization`

Enable the authorization checking of users and roles for the domain console or consoles that are being accessed. The default value of this property is `false` to maintain backward compatibility. To enable authorization checking, use the `svccfg(1M)` command to set the property value to `true`. While this option is enabled, `vntsd` listens and accepts connections on `localhost`. If the `listen_addr` property specifies an alternate IP address when this option is enabled, `vntsd` ignores the alternate IP address and continues to listen on `localhost`. Connections that are initiated from other hosts will also fail. Authorizations are available to access all consoles or console groups, or to access specific consoles or console groups. When the `vntsd` service is enabled, the following authorization is added to the authorization description database, `auth_attr(4)`:

```
solaris.vntsd.consoles:::Access All LDoms Guest Consoles::
```

Add any fine-grained authorizations based on the name of the console group. For example, if the name of the console group to be authorized is `ldg1`, add the following entry to the `auth_attr(4)` file:

```
solaris.vntsd.console-ldg1:::Access Specific LDoms Guest Console::
```

By default, the authorization to access all consoles is assigned to the root user or role. The superuser, or user with appropriate privileges, can use the `usermod(1M)` command to assign the required authorization or authorizations to other users or roles.

The following example gives user `user1` the authorization to access all domain consoles:

```
# usermod -A "solaris.vntsd.consoles" user1
```

The following example gives user `user1` the authorization to access the console group named `ldg1`:

```
# usermod -A "solaris.vntsd.console-ldg1" user1
```

Name volcopy – make an image copy of file system

Synopsis

```
volcopy [-F FSType] [-V] [generic_options]
     [-o FSType-specific_options] operands
```

Description `volcopy` makes a literal copy of the file system. This command may not be supported for all *FSTypes*.

Options The following options are supported:

`-F` *FSType* — Specify the *FSType* on which to operate. The *FSType* should either be specified here or be determinable from `/etc/vfstab` by matching the *operands* with an entry in the table. Otherwise, the default file system type specified in `/etc/default/fs` will be used.

`-V` — Echo the complete command line, but do not execute the command. The command line is generated by using the options and arguments provided by the user and adding to them information derived from `/etc/vfstab`. This option should be used to verify and validate the command line.

generic_options — Options that are commonly supported by most *FSType*-specific command modules. The following options are available:

- `-a` Require the operator to respond "yes" or "no" instead of simply waiting ten seconds before the copy is made.
- `-s` (Default) Invoke the DEL if wrong verification sequence.

`-o` *FSType-specific_options* — Specify *FSType*-specific options in a comma separated (without spaces) list of suboptions and keyword-attribute pairs for interpretation by the *FSType*-specific module of the command.

Operands The following operands are supported:

operands — generally include the device and volume names and are file system specific. A detailed description of the *operands* can be found on the *FSType*-specific man pages of `volcopy`.

Exit Status The following exit values are returned:

- `0` Successful file system copy
- `1` An error has occurred.

Files

`/etc/vfstab` list of default parameters for each file system

`/etc/default/fs` default local file system type. Default values can be set for the following flags in `/etc/default/fs`. For example: `LOCAL=ufs`.

`LOCAL:` The default partition for a command if no *FSType* is specified.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also `labelit(1M)`, `vfstab(4)`, `attributes(5)` Manual pages for the *FSType*-specific modules of `volcopy`.

Name volcopy_ufs – make an image copy of a ufs file system

Synopsis `volcopy [-F ufs] [`*generic_options*`] `*fsname srcdevice volname1 destdevice volname2*

Description `volcopy` makes a literal copy of the `ufs` file system using a blocksize matched to the device.

Options The following option is supported:

generic_options options supported by the generic `volcopy` command. See `volcopy(1M)`.

Operands The following operands are supported:

fsname represents the mount point (for example, `root`, `u1`, etc.) of the file system being copied.

srcdevice or *destdevice* the disk partition specified using the raw device (for example, `/dev/rdsk/cld0s8`, `/dev/rdsk/cld1s8`, etc.).

srcdevice and *volname1* the device and physical volume from which the copy of the file system is being extracted.

destdevice and *volname2* the target device and physical volume.

fsname and *volname* are limited to six or fewer characters and recorded in the superblock. *volname* may be '–' to use the existing volume name.

Exit Status The following exit values are returned:

`0` Successful file system copy.

non-zero An error has occurred.

Files `/var/adm/filesave.log` a record of file systems/volumes copied

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also `cpio(1)`, `dd(1M)`, `labelit(1M)`, `volcopy(1M)`, `attributes(5)`, `ufs(7FS)`

Notes `volcopy` does not support copying to tape devices. Use `dd(1M)` for copying to and from tape devices.

Name vrrpadm – VRRP administration tool

Synopsis

```
vrrpadm create-router -V vrid -l link -A inet | inet6
        [-p priority] [-i adv_interval] [-o flags] router_name

vrrpadm delete-router router_name

vrrpadm disable-router router_name

vrrpadm enable-router router_name

vrrpadm modify-router [-p priority] [-i adv_interval]
        [-o flags] [router_name]

vrrpadm show-router [-P | -x] [-p] [-o field[,...]] [router_name]
```

Description The `vrrpadm` command is used to administer the VRRP (Virtual Router Redundancy Protocol) service in a system.

VRRP specifies an election protocol that dynamically assigns responsibility for a virtual router to one of the VRRP routers within a LAN. At a given moment, only one VRRP router controls the IPv4 or IPv6 virtual address(es) associated with a virtual router (known as the *master*), and forwards packets sent to these IP addresses. The election process provides dynamic failover of the forwarding responsibility should the master become unavailable.

Each `vrrpadm` subcommand operates on a VRRP router, which is identified by a name given by the administrator. VRRP routers with the same VRID and address family within a LAN comprise a virtual router, which protects a set of virtual IP addresses.

A system can have multiple VRRP routers; each belongs to a different virtual router.

Sub-commands The following subcommands are supported. Note that all subcommands but `show-router` require the `solaris.network.vrrp` authorization. The `show-router` subcommand does not require special authorizations.

`vrrpadm create-router -V` *vrid* `-l` *link* `-A` *inet* | *inet6* [`-p` *priority*] [`-i` *adv_interval*] [`-o` *flags*] *router_name*

Create a VRRP router with a specified configuration.

`-A` *inet* | *inet6*, `--address_family=`*inet* | *inet6*
Address family. Either IPv4 or IPv6.

`-i` *adv_interval*, `--adv_interval=`*adv_interval*
The advertisement interval in milliseconds. Default is `1000` (one second). The valid interval range is 10-40950.

`-l` *link*, `--link=`*link*
The data link on which the VRRP router is configured. This determines the LAN this VRRP router is running in. The data-link can be a physical link, a VLAN, or an aggregation.

`-o` *flags*, `--flags=`*flags*
The preempt and accept modes, delimited by a comma. Values can be:

- `preempt`
- `un_preempt`
- `accept`
- `noaccept`

By default both modes are set to true.

The `preempt` mode controls whether an enabled higher priority backup router preempts a lower priority master router. If `preempt` mode is true, then the preemption is allowed; otherwise, preemption is prohibited. Note that the `preempt` mode must be true if the VRRP router is the owner of the virtual IP addresses.

The `accept` mode controls the local packet acceptance of the virtual IP addresses. If `accept` mode is true, the master must accept packets sent to the virtual IP addresses. If `accept` mode is false, the master does not accept those packets, although it does respond to ARP requests or ND Solicitations and Advertisement for those non-accepted virtual IP addresses. It also must forward packets for the router specified in this subcommand. Note that `accept` mode must be true if the VRRP router is the owner of the virtual IP addresses. An example of syntax for this option:

```
-o preempt,no_accept
```

`-p` *priority*, `--priority=`*priority*
The priority of the specified VRRP router used in master selection. The higher the value, the greater the possibility the router is selected as the master.

The default value is 255, which indicates the specified VRRP router is the IP Address Owner and owns all the virtual IP addresses. An IP Address Owner will respond to the packets addressed to one of the virtual IP addresses for ICMP pings, TCP connections, and so forth.

The range 1-254 is available for VRRP routers backing up a virtual router. Master selection is weighted toward the VRRP router with the higher priority.

`-V` *vrid*, `--VRID=`*vrid*
The virtual router identifier (VRID). Together with the address family, it identifies a virtual router within a LAN.

router_name
The name of a VRRP router. This name is used to identify a VRRP router in other `vrrpadm` subcommands.

The maximum length of a valid router name is 31 characters. Legal characters are alphanumeric (a-z, A-Z, 0-9) and the underscore ('_').

`vrrpadm delete-router` *router_name*
Delete the VRRP router identified by *router_name*.

`vrrpadm disable-router` *router_name*
Disable the virtual router identified by *router_name*. Once the router is disabled, it will stop participating in the master selection process in the virtual router.

`vrrpadm enable-router` *router_name*
Re-enable the virtual router identified by *router_name* that was disabled. The router will resume participating in the master selection process in the virtual router.

`vrrpadm modify-router` [`-p` *priority*] [`-i` *adv_interval*] [`-o` *flags*] [*router_name*]
Modify the configuration of the VRRP router identified by *router_name*. Only the priority, the advertisement interval, the preempt mode, and the accept mode can be modified.

`-p` *priority*, `--priority=`*priority*	The new priority of this VRRP router.
`-i` *adv_interval*, `--adv_interval=`*adv_interval*	The new advertisement interval.
`-o` *flags*, `--flags=`*flags*	The new preempt and accept modes. Either one or both can be specified. If both are specified, they are delimited by a comma. For example: `-o preempt,no_accept`

`vrrpadm show-router` [`-P` | `-x`] [`-p`] [`-o` *field*[,...]] [*router_name*]
Display the information for the VRRP router identified by *router_name*. If no *router_name* is specified, display information for all the VRRP routers on the system.

By default (with no options), the following fields are displayed:

`NAME`
The name of the VRRP router.

`VRID`
The VRID of the VRRP router.

`LINK`
The data link on which the VRRP router is created.

`AF`
he address family of the VRRP router, either `IPv4` or `IPv6`.

`PRIO`
The priority of this VRRP router used in master selection.

`ADV_INTV`
The advertisement interval, in milliseconds.

`STATE`
The current state of the VRRP router, `INIT` (Initialize), `BACK` (Backup), or `MAST` (Master).

`MODE`
A set of flags associated with the VRRP router. Possible values are:

e The router has been enabled.

p Preempt mode is true.

a Accept mode is true.

o Virtual address owner.

`VNIC`
The VRRP VNIC created for this VRRP router.

Note that the name of the VNIC can change over time unless the router is enabled.

The `show-router` subcommand has the following options:

`-x`, `--extended`
Display additional information of the given VRRP router:

`PRIMARY_IP`
The primary IP address selected by the VRRP router.

`VIRTUAL_IPS`
The virtual IP addresses configured on the VRRP router.

`PRV_STAT`
The previous state of the VRRP router.

`STAT_LAST`
Time since the last state transition.

`-P`, `--peer`
Display information for the backup VRRP router. This option is meaningful only when the VRRP router is in the backup state.

The following fields are displayed:

`NAME`
The name of the VRRP router.

`PEER`
The primary IP address of the peer VRRP router.

`P_PRIO`
The priority of the peer VRRP router contained in the advertisement received from the peer.

`P_INTV`
The advertisement interval (in milliseconds) contained in advertisements received from the peer.

`P_ADV_LAST`
Time since last received advertisement from the peer.

`MASTER_DOWN_INT`
Time interval (in milliseconds) after which to declare Master down.

`-p`, `--parseable`
Display the VRRP router information in the machine parseable format.

`-o` *field*[,...], `--output=`*field*
A case-insensitive, comma-separated list of output fields to display. The field name must be one of the fields listed above, or the special value `all` to display all fields. By default (without `-o`), `vrrpadm show` displays all fields.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/network/routing/vrrp
Interface Stability	Committed

See Also `dladm(1M)`, `vrrpd(1M)`, `attributes(5)`

Name vrrpd – VRRP daemon

Synopsis

```
/usr/lib/inet/vrrpd

svc:/network/vrrp:default
```

Description `vrrpd` is the system daemon program that handles the administrative events for the VRRP routers. It is controlled through the service management facility (SMF) service instance:

```
svc:/network/vrrp:default
```

The daemon should not be invoked directly. It does not constitute an administrative nor a programming interface. The administrative interface for managing VRRP routers is `vrrpadm(1M)`.

Options The daemon has no options.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/network/routing/vrrp
Interface Stability	Private

See Also `vrrpadm(1M)`, `attributes(5)`

Name vscanadm – vscan service configuration utility

Synopsis

```
vscanadm set -p property=value [-p property=value]...

vscanadm get [-p property]...

vscanadm import -p property filename

vscanadm export -p property filename

vscanadm validate -p property filename

vscanadm add-engine [-p property=value]... engine_id

vscanadm remove-engine engine_id

vscanadm set-engine -pproperty=value [-p property=value]... engine_id

vscanadm get-engine [-p property=value]... [engine_id]

vscanadm show

vscanadm stats [-z]
```

Description The `vscanadm` command sets and displays properties of the `vscan` service, `zpool(1M)`, and provides scan statistics.

File system exemption from virus scanning may be configured per file system using the appropriate file system administrative command, for example `zfs(1M)`.

Scan engines are third-party applications on external hosts that perform the actual virus scanning operation on files. Multiple scan engines can be configured for use by the `vscan` service. A minimum of two scan engines is recommended. File scan requests are distributed among the configured scan engines to achieve load balancing. A scan engine is identified by its *engine_id*. The *engine_id* is a user defined string of up to 64 bytes.

The `vscan` service properties are divided into two categories: scan engine properties, which are specific to a scan engine definition, and general properties, which apply to the service and are not scan engine-specific.

Subcommands `vscanadm` recognizes the following subcommands:

`vscanadm set -p` *property=value* [`-p` *property=value*]...
Sets the values of `vscan` service general properties.

`-p` *property=value* Specifies a property value

`vscanadm get` [`-p` *property*]...
Displays the values of `vscan` service general properties. If no properties are specified, all `vscan` service general properties are displayed.

`-p` *property* Specifies a property value

The following properties are available for the `vscanadm set` and `vscanadm get` subcommands:

`max-size` The maximum size of files that should be virus scanned. Files exceeding *max-size* are not scanned. The *max-size-action* property determines whether access should be allowed or denied to files that exceed *max-size*.

The value of *max-size* is a string with a numeric (decimal) component and an optional letter component that specifies a unit size, in the format "N[.N][KMGTP][B]".

Following the numeric component, the optional unit can be specified as either one or two characters. For example, either "K" or "KB" can be used to specify kilobytes. Unit specifiers are not case-sensitive, and must follow the numeric value immediately with no intervening whitespace.

With either no unit specifier, or a unit specifier of only "B", the numeric value is assumed to be in bytes. The default value is 1GB.

Note that while the `vscan` service defines a maximum file size for scanning, scan engines also typically define their own maximum file size setting. It is recommended that *max-size* be set to a value less than or equal to the maximum file size for the scan engine(s).

`max-size-action` Specifies whether access will be allowed or denied to files larger than *max-size*. Files larger than *max-size* are not virus scanned. Valid values are:

allow allow access to files larger than *max-size* (no virus scan). This is the default value.

deny deny access to files larger than *max-size* (no virus scan)

`vscanadm import -p` *property filename*
Imports the property value from the specified file. The file must contain a single line specifying the value of a single property.

`vscanadm export -p` *property filename*
Exports the property value to the specified file. The file must contain a single line specifying the value of a single property.

`vscanadm validate -p` *property filename*
Validates the property value in the specified file. The file must contain a single line specifying the value of a single property.

The following properties are available for the `vscanadm import`, `vscanadm export`, and `vscanadm validate` subcommands:

`types` A comma-separated list of file type extension matching rules. This list defines which types of files are scanned and which should be excluded during virus

scanning. Each rule comprises the rule indicator [+|-], followed by a file type *expresssion* against which a file's type extension is compared. The file type *expresssion* is case insensitive and may include the "*" and "?" wildcards. There should be no whitespace between the rule indicator and the file type *expresssion*. If a comma is included within the file type expression, it must be escaped using a "\" (backslash). A file type extension does not include its preceding dot.

The rule indicator is a single character and can be one of:

```
+ include file type in virus scanning
- exclude file type from virus scanning
```

When a file is being evaluated as a candidate for virus scanning, its file type will be compared with the rules defined in types. The first rule matched will be applied. If no match is found, the file will be virus scanned. The total length of the types string can not exceed 4096 bytes. The default content of the types list is "+*".

`vscanadm add-engine` [`-p` *property=value*]... *engine_id*

Adds a new scan engine identified by *engine_id*. The default values are used for any scan engine properties that are not specified. The hostname defaults to the *engine_id*.

`-p` *property=value* Specifies a property value

`vscanadm remove-engine` *engine_id*

Remove scan engine identified by *engine_id*, removing all of its configuration property values.

`vscanadm set-engine -p`*property=value* [`-p` *property=value*]... *engine_id*

Creates or updates the configuration property values for the scan engine identified by *engine_id*.

`-p` *property=value* Specifies a property value

`vscanadm get-engine` [`-p` *property=value*]... [*engine_id*]

Displays the values of the specified scan engine properties for the scan engine identified by *engine_id*. If no *engine_id* is specified, this subcommand displays the specified scan engine property values for all configured scan engines. If no properties are specified, this subcommand displays all `vscan` service scan engine properties.

`-p` *property=value* Specifies a property value

The following properties are available for the `vscanadm add-engine`, `vscanadm remove-engine`, `vscanadm set-engine`, and `vscanadm get-engine` subcommands:

`enable` Specifies whether the scan engine is enabled or disabled. Valid values are "on" (enabled) and "off" (disabled). The default is "on" (enabled). A scan engine cannot be enabled if its host property is invalid.

`host` Hostname or IPv4 format IP address of the scan engine.

`port` ICAP port number of the scan engine. The numeric value ranges from 0 to 65535. The default ICAP port is 1344.

`max-connection` The maximum number of concurrent connections that may be established with a scan engine. The numeric value ranges from 1 to 512. This property defaults to 8.

`vscanadm show` Displays the values of all `vscan` service general properties and scan engine properties.

`vscanadm stats [-z]` Displays or resets the following `vscan` service statistics:

- number of files scanned
- number of infected files
- number of failed scan requests
- scan errors (including a per scan engine error count)

`-z` Resets `vscan` service statistics counters to zero

Examples

EXAMPLE 1 Setting the Maximum Size Limit

To set the maximum size limit for files to be virus scanned to 128 megabytes, enter

```
# vscanadm set -p max-size=128M
```

EXAMPLE 2 Allowing Access to Files

To allow access to files exceeding the maximum file size, enter

```
# vscanadm set -p max-size-action=allow
```

EXAMPLE 3 Setting File Types

To set the types so that only files of type “odt”, “exe” and “jpg” are virus scanned, enter

```
# vscanadm set -p types=+odt,+exe,+jpg,-*
```

To set the types so that all file types except “doc” are virus scanned, enter

```
# vscanadm set -p types=-doc,+*
```

EXAMPLE 4 Displaying the File Types List

To display the file types list, enter

```
# vscanadm get -p types
```

EXAMPLE 5 Adding the Scan Engine

To add the scan engine “`my_eng`” using the default values, enter

```
# vscanadm add-engine my_eng
```

EXAMPLE 6 Disabling the Scan Engine

To disable the scan engine "my_eng", enter

```
# vscanadm set-engine -p enable=off my_eng
```

EXAMPLE 7 Displaying Scan Engine Properties

To display the properties of the scan engine "my_eng", enter

```
# vscanadm get-engine my_eng
```

EXAMPLE 8 Removing Scan Engine

To remove the scan engine "my_eng", enter

```
# vscanadm remove-engine my_eng
```

EXAMPLE 9 Displaying Vscan Service General and Scan Engine Properties

To Display all vscan service general properties and scan engine properties, enter

```
# vscanadm show
```

Exit Status

The following exit values are returned:

`0` Successful completion.

`non-zero` An error occurred.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	service/storage/virus-scan
Interface Stability	Uncommitted
Utility output format	Not-An-Interface

See Also

`vscand(1M)`, `zfs(1M)`, `attributes(5)`, `smf(5)`

Notes

All users are permitted to use `vscanadm` to view `vscan` properties and statistics. To set property values or reset statistics, the following authorizations are required:

`solaris.smf.value.vscan`
change the property values or reset statistics

`solaris.manage.vscan`
refresh the service to apply property value changes

To add or remove properties (`add-engine`, `remove-engine`) the following authorizations are required:

`solaris.smf.modify.application`	add or remove property group
`solaris.manage.vscan`	refresh the service to apply property value changes

All of these authorizations are included in the "VSCAN Management" profile.

Name vscand – vscan service daemon

Synopsis

```
/usr/lib/vscan/vscand
```

Description `vscand` is the daemon that handles virus scan requests from file systems on file open and close operations. A file system may support enabling and disabling of virus scanning on a per dataset basis, using that file system's administrative command, for example `zfs(1M)`.

If the file state or scan policy (see `vscanadm(1M)` requires that a file be scanned, `vscand` communicates with external third-party virus scanners (scan engines) using the Internet Content Adaptation Protocol (ICAP, RFC 3507) to have the file scanned.

A file is submitted to a scan engine if it has been modified since it was last scanned, or if it has not been scanned with the latest scan engine configuration (Virus definitions). The file's modified attribute and `scanstamp` attribute are used to store this information. Once the file is scanned, the modified attribute is cleared and the `scanstamp` attribute is updated.

If the file is found to contain a virus, the virus is logged in `syslogd(1M)`, an audit record is written, and the file is quarantined (by setting its quarantine attribute). Once a file is quarantined, attempts to read, execute or rename the file will be denied by the file system. The `syslogd(1M)` entry and the audit record specify the name of the infected file and the violations detected in the file. Each violation is specified as “ID - threat description”, where ID and threat description are defined in the X-Infection-Found-Header in ICAP RFC 3507; Extensions.

By default, `vscand` connects to scan engines on port 1344. The port and other service configuration parameters can be configured using `vscanadm(1M)`.

The `vscan` service is disabled by default, and can be enabled using `svcadm(1M)`.

Exit Status The following exit values are returned:

`0` Daemon started successfully.

`non-zero` Daemon failed to start.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	service/storage/virus-scan
Interface Stability	Uncommitted

See Also `ps(1)`, `svcs(1)`, `logadm(1M)`, `svcadm(1M)`, `syslogd(1M)`, `vscanadm(1M)`, `zfs(1M)`, `attributes(5)`, `smf(5)`

Notes If a file is accessed using a protocol which does not invoke the file system open and close operations, for example NFSv3, virus scanning is not initiated on the file.

File content is transferred to the scan engines as `cleartext` data.

Administrative actions for the `vscan` service, such as enabling, disabling, or requesting a restart, can be performed using `svcadm(1M)`. The `vscan` service status can be queried using the `svcs(1)` command.

The `vscan` service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/system/filesystem/vscan
```

Name vtdaemon – switch function between text virtual consoles

Synopsis

```
/usr/lib/vtdaemon
```

Description `vtdaemon` provides a secure switch function, by means of hotkeys, between different text virtual consoles. It runs with full privileges. See `vt(7I)` for specifications of hotkey sequences.

The service is managed using the action authorization `solaris.smf.manage.vt` which is included in the Device Security Rights Profile. The service is local only and has no inbound network ports. It is disabled in non-global zones or when the virtual console functionality is not available.

`vtdaemon` uses PAM for authentication.

Note that `vtdaemon` only locks and unlocks the text console. For graphical X sessions, it invokes the corresponding lock program (for example, `xscreensaver`), which deals with authentication and audit.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os
Interface Stability	Private

See Also `svcs(1)`, `svcadm(1M)`, `attributes(5)`, `smf(5)`, `vt(7I)`

Notes The `vtdaemon` service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/system/vtdaemon:default
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. The service's status can be queried using the `svcs(1)` command.

The `vtdaemon` service has the following SMF configuration properties:

`hotkeys` (boolean)
: Allows authorized users to dynamically enable or disable the VT switch by means of hotkeys. Its default value is `TRUE` (enabled).

`secure` (boolean)
: Allows authorized users to dynamically enable or disable the security of hotkeys. If disabled, the user can freely switch to any session without authentication. Its default value is TRUE (enabled).

Name wall – write to all users

Synopsis `/usr/sbin/wall [-a] [-g` *grpname*`] [`*filename*`]`

Description `wall` reads its standard input until an end-of-file. It then sends this message to all currently logged-in users preceded by:

```
Broadcast Message from . . .
```

If *filename* is given, then the message is read in from that file. Normally, pseudo-terminals that do not correspond to rlogin sessions are ignored. Thus, when using a window system, the message appears only on the console window. However, `-a` will send the message even to such pseudo-terminals.

It is used to warn all users, typically prior to shutting down the system.

The sender must be superuser to override any protections the users may have invoked See `mesg(1)`.

`wall` runs `setgid()` to the group ID `tty`, in order to have write permissions on other user's terminals. See `setuid(2)`.

`wall` will detect non-printable characters before sending them to the user's terminal. Control characters will appear as a " ^ " followed by the appropriate ASCII character; characters with the high-order bit set will appear in "meta" notation. For example, '`\003`' is displayed as '`^C`' and '`\372`' as '`M–z`'.

Options The following options are supported:

`-a` Broadcast message to the console and pseudo-terminals.

`-g` *grpname* Broadcast to the users in a specified group only, per the group database (see `group(4)`).

Environment Variables If the `LC_*` variables (`LC_CTYPE`, `LC_TIME`, `LC_COLLATE`, `LC_NUMERIC`, and `LC_MONETARY`) are not set in the environment, the operational behavior of `wall` for each corresponding locale category is determined by the value of the `LANG` environment variable. See `environ(5)`. If `LC_ALL` is set, its contents are used to override both the `LANG` and the other `LC_*` variables. If none of the above variables are set in the environment, the "C" (U.S. style) locale determines how `wall` behaves.

Files `/dev/tty*`

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also mesg(1), write(1), setuid(2), attributes(5), environ(5)

Notes wall displays "Cannot send to ..." when the open on a user's tty file fails.

Name wanboot_keygen – create and display client and server keys for WAN booting

Synopsis

```
/usr/lib/inet/wanboot/keygen -c -o net=a.b.c.d ,cid=client_ID,type=3des

/usr/lib/inet/wanboot/keygen -c -o net=a.b.c.d ,cid=client_ID,type=aes

/usr/lib/inet/wanboot/keygen -m

/usr/lib/inet/wanboot/keygen -c -o net=a.b.c.d ,cid=client_ID,type=sha1

/usr/lib/inet/wanboot/keygen -d -m

/usr/lib/inet/wanboot/keygen -c -o net=a.b.c.d ,cid=client_ID,type=keytype
```

Description The keygen utility has three purposes:

- Using the -c flag, to generate and store per-client 3DES/AES encryption keys, avoiding any DES weak keys.
- Using the -m flag, to generate and store a "master" HMAC SHA-1 key for WAN install, and to derive from the master key per-client HMAC SHA-1 hashing keys, in a manner described in RFC 3118, Appendix A.
- Using the -d flag along with either the -c or -m flag to indicate the key repository, to display a key of type specified by *keytype*, which must be one of 3des, aes, or sha1.

The net and cid arguments are used to identify a specific client. Both arguments are optional. If the cid option is not provided, the key being created or displayed will have a per-network scope. If the net option is not provided, then the key will have a global scope. Default net and code values are used to derive an HMAC SHA-1 key if the values are not provided by the user.

Options The following options are supported:

-c Generate and store per-client 3DES/AES encryption keys, avoiding any DES weak keys. Also generates and stores per-client HMAC SHA-1 keys. Used in conjunction with -o.

-d Display a key of type specified by *keytype*, which must be one of 3des, aes, or sha1. Use -d with -m or with -c and -o.

-m Generate and store a "master" HMAC SHA-1 key for WAN install.

-o Specifies the WANboot client and/or keytype.

Examples

EXAMPLE 1 Generate a Master HMAC SHA-1 Key

```
# keygen -m
```

EXAMPLE 2 Generate and Then Display a Client-Specific Master HMAC SHA-1 Key

```
# keygen -c -o net=172.16.174.0,cid=010003BA0E6A36,type=sha1
# keygen -d -c -o net=172.16.174.0,cid=010003BA0E6A36,type=sha1
```

EXAMPLE 3 Generate and Display a 3DES Key with a Per-Network Scope

```
# keygen -c -o net=172.16.174.0,type=3des
# keygen -d -o net=172.16.174.0,type=3des
```

Exit Status

`0` Successful operation.

`>0` An error occurred.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/boot/wanboot
Interface Stability	Obsolete

See Also

`attributes(5)`

Name wanboot_keymgmt – insert and extract keys

Synopsis

```
/usr/lib/inet/wanboot/keymgmt -i -k key_file -s keystore -o type=keytype
/usr/lib/inet/wanboot/keymgmt -x -f outfile -s keystore -o type=keytype
```

Description The `keymgmt` utility has two purposes:

- To take a raw key, stored in *key_file*, and insert it in the repository specified by *keystore*.
- To extract a key of a specified type from the repository specified by *keystore*, depositing it in *outfile*.

outfile will be created if it does not already exist. The type of key being added or extracted is specified by *keytype* and may have one of four values: `3des`, `aes`, `rsa`, or `sha1` (the last used by HMAC SHA-1). When extracting a key, the first key with an OID matching the supplied type is used.

Arguments The following arguments are supported:

`-i`	Used in conjunction with `-k` to insert a raw key in *keystore*.
`-f` *outfile*	Used to specify a file to receive an extracted key.
`-k` *key_file*	Used in conjunction with `-i` to specify the file in which a raw key is stored. This key will be inserted in *keystore*.
`-o` type=*keytype*	Specifies the type of key being inserted or extracted. Must be one of `3des`, `aes`, `rsa`, or `sha1`.
`-s` *keystore*	Specifies a repository in which a key will be inserted or from which a key will be extracted.
`-x`	Used in conjunction with `-f` to extract a key of a specified type and deposit it in *outfile*.

Exit Status

`0`	Successful operation.
`>0`	An error occurred.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/boot/wanboot
Interface Stability	Obsolete

See Also `attributes(5)`

ITU-T Recommendation X.208

Name wanboot_p12split – split a PKCS #12 file into separate certificate and key files

Synopsis

```
/usr/lib/inet/wanboot/p12split -i p12file -c out_cert -k out_key
     [-t out_trust -l id -v]
```

Description The `p12split` utility extracts a certificate and private key from the repository specified by *p12file*, depositing the certificate in *out_cert* and the key in *out_key*. If supplied, the `-l` option specifies the value for the `LocalKeyId` that will be used in the new certificate and key files. `p12split` can optionally extract a trust certificate into the *out_trust* file if the `-t` option is specified. Use the `-v` option to get a verbose description of the split displayed to standard output.

Options The following arguments and options are supported:

`-c` *out_cert* Specifies a repository that receives a extracted certificate.

`-i` *p12file* Specifies a repository from which a certificate and private key is extracted.

`-k` *out_key* Specifies a repository that receives a extracted private key.

`-l` *id* Specifies the value for the `LocalKeyId` that will be used in the new certificate and key files.

`-t` *out_trust* Specifies a file for receiving an extracted trust certificate.

`-v` Displays a verbose description of the split to stdout.

Exit Status

`0` Successful operation.

`>0` An error occurred.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/boot/wanboot
Interface Stability	Uncommitted

See Also `attributes(5)`

Name wanbootutil – manage keys and certificates for WAN booting

Synopsis

```
wanbootutil [keygen] [option_specific_arguments]

wanbootutil [keymgmt] [option_specific_arguments]

wanbootutil [p12split] [option_specific_arguments]
```

Description The `wanbootutil` command creates and manages WANboot encyrption and hashing keys and manipulates PKCS #12 files for use by WAN boot.

`wanbootutil` has three subcommands, each covered in a separate man page:

`wanboot_keygen(1M)`	Generates encryption and hashing keys.
`wanboot_keymgmt(1M)`	Inserts and extracts keys from WAN boot key repositories.
`wanboot_p12split(1M)`	Splits a PKCS #12 file into separate certificate and key files for use by WAN boot.

Options The options are supported for `wanbootutil` are the use of `keygen`, `keymgmt`, or `p12split`. The options for these subcommands are described in their respective man pages.

Examples

EXAMPLE 1 Generate a 3DES Client Key

```
# wanbootutil keygen -c -o net=172.16.174.0,cid=010003BA0E6A36,type=3des
```

EXAMPLE 2 Insert an RSA Private Client Key

```
wanbootutil keymgmt -i -k keyfile \
  -s /etc/netboot/172.16.174.0/010003BA0E6A36/keystore -o type=rsa
```

EXAMPLE 3 Split a PKCS #12 File into Certificate and Key Components

```
# wanbootutil p12split -i p12file -c out_cert -k out_key
```

Exit Status

`0`	Successful operation.
`non-zero`	An error occurred. Writes an appropriate error message to standard error.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/boot/wanboot
Interface Stability	Obsolete

See Also `wanboot_keygen(1M)`, `wanboot_keymgmt(1M)`, `wanboot_p12split(1M)`, `attributes(5)`

Name wbemadmin – start Sun WBEM User Manager

Synopsis

```
/usr/sadm/bin/wbemadmin
```

Description The `wbemadmin` utility starts Sun WBEM User Manager, a graphical user interface that enables you to add and delete authorized WBEM users and to set their access privileges. Use this application to manage access to groups of managed resources, such as disks and installed software, in the Solaris operating environment.

The `wbemadmin` utility allows you to perform the following tasks:

Manage user access rights	Use the `wbemadmin` utility to add, delete, or modify an individual user's access rights to a namespace on a WBEM-enabled system.
Manage namespace access rights	Use the `wbemadmin` utility to add, delete, or modify access rights for all users to a namespace.

The Sun WBEM User Manager displays a Login dialog box. You must log in as root or a user with write access to the `root\security` namespace to grant access rights to users. By default, Solaris users have guest privileges, which grants them read access to the default namespaces.

Managed resources are described using a standard information model called Common Information Model (CIM). A CIM object is a computer representation, or model, of a managed resource, such as a printer, disk drive, or CPU. CIM objects can be shared by any WBEM-enabled system, device, or application. CIM objects are grouped into meaningful collections called schema. One or more schemas can be stored in directory-like structures called namespaces.

All programming operations are performed within a namespace. Two namespaces are created by default during installation:

- `root\cimv2` — Contains the default CIM classes that represent objects on your system.
- `root\security` — Contains the security classes used by the CIM Object Manager to represent access rights for users and namespaces.

When a WBEM client application connects to the CIM Object Manager in a particular namespace, all subsequent operations occur within that namespace. When you connect to a namespace, you can access the classes and instances in that namespace (if they exist) and in any namespaces contained in that namespace.

When a WBEM client application accesses CIM data, the WBEM system validates the user's login information on the current host. By default, a validated WBEM user is granted read access to the Common Information Model (CIM) Schema. The CIM Schema describes managed objects on your system in a standard format that all WBEM-enabled systems and applications can interpret.

You can set access privileges on individual namespaces or for a user-namespace combination. When you add a user and select a namespace, by default the user is granted read access to CIM objects in the selected namespace. An effective way to combine user and namespace access rights is to first restrict access to a namespace. Then grant individual users read, read and write, or write access to that namespace.

You cannot set access rights on individual managed objects. However you can set access rights for all managed objects in a namespace as well as on a per-user basis.

If you log in to the root account, you can set the following types of access to CIM objects:

- Read Only — Allows read-only access to CIM Schema objects. Users with this privilege can retrieve instances and classes, but cannot create, delete, or modify CIM objects.
- Read/Write — Allows full read, write, and delete access to all CIM classes and instances.
- Write — Allows write and delete, but not read access to all CIM classes and instances.
- None — Allows no access to CIM classes and instances.

Context help is displayed in the left side of the `wbemadmin` dialog boxes. When you click on a field, the help content changes to describe the selected field. No context help is available on the main User Manager window.

The `wbemadmin` security administration tool updates the following Java classes in the `root\security` namespace:

- `Solaris_UserAcl` — Updated when access rights are granted or changed for a user.
- `Solaris_namespaceAcl` — Updated when access rights are granted or changed for a namespace.

Usage The `wbemadmin` utility is not the tool for a distributed environment. It is used for local administration on the machine on which the CIM Object Manager is running.

Exit Status The `wbemadmin` utility terminates with exit status `0`.

Warning The `root\security` namespace stores access privileges. If you grant other users access to the `root\security` namespace, those users can grant themselves or other users rights to all other namespaces.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWwbco

See Also mofcomp(1M), wbemlogviewer(1M), init.wbem(1M), attributes(5)

Name wbemconfig – convert a JavaSpaces datastore to the newer Reliable Log datastore format

Synopsis

```
/usr/sadm/lib/wbem/wbemconfig convert
```

Description A Reliable Log directory is created that contains the converted data. This directory is named `/var/sadm/wbem/logr`.

The `convert` argument is the only supported option of this command. You should only run this command after stopping WBEM (CIM Object Manager) with the `init.wbem stop` command. Otherwise your data may be corrupted.

This command successfully converts any proprietary custom MOFs you have created in the datastore, but not any CIM or Solaris MOFs you have modified. These will be destroyed. To recompile any modified CIM or Solaris MOFs into the new datastore, run the `mofcomp` command on the MOF files containing the class definitions.

Because the `wbemconfig convert` command invokes the JVM (Java Virtual Machine) to perform conversion of the JavaSpaces datastore, you must be running the same version of the JVM as when the original JavaSpaces storage was created. After the `wbemconfig convert` command is completed, you can change to any version of the JVM you want.

To see what version of the JVM you are running, issue the `java -version` command.

Options The following options are supported:

convert Convert a JavaSpaces datastore to the newer Reliable Log datastore format.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWwbco

See Also `init.wbem(1M)`, `wbemadmin(1M)`, `wbemlogviewer(1M)`, `mofcomp(1M)`, `attributes(5)`

Name wbemlogviewer – start WBEM Log Viewer

Synopsis

```
/usr/sadm/bin/wbemlogviewer
```

Description The `wbemlogviewer` utility starts the WBEM Log Viewer graphical user interface, which enables administrators to view and maintain log records created by WBEM clients and providers. The WBEM Log Viewer displays a Login dialog box. You must log in as root or a user with write access to the `root\cimv2` namespace to view and maintain log files. Namespaces are described in `wbemadmin(1M)`.

Log events can have three severity levels.

- Errors
- Warnings
- Informational

The WBEM log file is created in the `/var/sadm/wbem/log` directory, with the name `wbem_log`. The first time the log file is backed up, it is renamed `wbem_log.1`, and a new `wbem_log` file is created. Each succeeding time the `wbem_log` file is backed up, the file extension number of each backup log file is increased by 1, and the oldest backup log file is removed *if* the limit, which in turn is specified in the log service settings, on the number of logfiles is exceeded. Older backup files have higher file extension numbers than more recent backup files.

The log file is renamed with a `.1` file extension and saved when one of the following two conditions occur:

- The current file reaches the specified file size limit.
- A WBEM client application uses the `clearLog()` method in the `Solaris_LogService` class to clear the current log file.
- A WBEM client application uses the `clearLog()` method in the Solaris_LogService class to clear the current log file.
- A user chooses Action->Back Up Now in the Log Viewer application.

Help is displayed in the left panel of each dialog box. Context help is not displayed in the main Log Viewer window.

Usage The WBEM Log Viewer is not the tool for a distributed environment. It is used for local administration.

The WBEM Log Viewer allows you to perform the following tasks:

View the logs

Set properties of log files	Click Action->Log File Settings to specify log file parameters and the log file directory.
Back up a log file	Click Action->Back Up Now to back up and close the current log file and start a new log file.

Open historical log files	Click Action->Open Log File to open a backed-up log file.
Delete an old log file	Open the file and then click Action->Delete Log File. You can only delete backed-up log files.
View log record details	Double-click a log entry or click View->Log Entry Details to display the details of a log record.
Sort the logs	Click View->Sort By to sort displayed entries. You can also click any column heading to sort the list. By default, the log entries are displayed in reverse chronological order (new logs first).

Exit Status The `wbemlogviewer` utility terminates with exit status `0`.

Files `/var/sadm/wbem/log/wbem_log` WBEM log file

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWwbco

See Also `wbemadmin(1M)`, `init.wbem(1M)`, `mofcomp(1M)`, `attributes(5)`

Name wcadmin – manage the configuration of the Sun Java Web Console

Synopsis

```
/usr/sbin/wcadmin subcommand options
/usr/sbin/wcadmin [-h| --help]
/usr/sbin/wcadmin [-V | --version]
```

Description `wcadmin` is a Command Line Interface (CLI) — based tool for managing the configuration of the Sun Java Web Console.

Subcommands The following subcommands are supported:

add — The `wcadmin add` subcommand adds a new shared jar file, a new JAAS login module, or a new shared service property to the console configuration. An optional identifier may be specified; if omitted, the identifier is derived from the resource name. The resource is added when the console is next started.

The format of the `add` subcommand is:

add `-l -a` *appname* [`-n` *id*] *jarpath*

add `-m -a` *appname* [`-n` *id*] `-b` *behavior* `-s` *service* [`-o` *name=value*] *classname*

add `-p -a` *appname name=value* [... *name=value*]

deploy — The `wcadmin deploy` subcommand deploys the specified web application into the console's web server instance. Applications are deployed directly from their installation directories.

The format of the `deploy` subcommand is:

deploy [`-D`] `-a` *appname* `-x` *context app_path*

disable — The `wcadmin disable` subcommand disables access to the specified web application in the console's web server instance.

The format of the `disable` subcommand is:

disable `-x` *context*

enable — The `wcadmin enable` subcommand enables access to the specified web application in the console's web server instance.

The format of the `enable` subcommand is:

enable `-x` *context*

list — The `wcadmin list` subcommand lists the resources currently configured for the console; including deployed web applications, shared jar files, login modules, and shared service properties. If no option is specified, all resources are listed.

The format of the `list` subcommand is:

list [`-a`] [`-d`] [`-l`] [`-m`] [`-p`]

password — The `wcadmin password` subcommand manages the administrator and security keystore passwords for the console. Keystore passwords should not be changed while the console is running.

The format of the `password` subcommand is:

password [`-a`] [`-k`] [`-t`]

password `-f` *password_file*

reload — The `wcadmin reload` subcommand unloads the specified web application from the console's web server instance and reloads the application from its original installation directory.

The format of the `reload` subcommand is:

reload `-x` *context*

remove — The `wcadmin remove` subcommand removes a shared jar file, a login module, or a shared service property from the console configuration. The resource may be specified by its identifier or by its full jarpath or classname. The resource is removed when the console is next started.

The format of the `remove` subcommand is:

remove `-l -a` *appname* [`-n` *id*] *jarpath*

remove `-m -a` *appname* [`-n` *id*] *classname*

remove `-p -a` *appname property* [...]

undeploy — The `wcadmin undeploy` subcommand undeploys the specified web application from the console's web server instance.

The format of the `undeploy` subcommand is:

undeploy [`-D`] `-a` *appname* `-x` *context*

Options The following options are supported:

`-h` | `--help` | `-?` — Display runtime help.

`-V` | `--version` — Display console version information.

`-D` | `--defer` — When used with the deploy and undeploy subcommands, defers the deployment or undeployment until the next console restart.

The operation is deferred by simply adding or removing the corresponding resource registration notification file. If defer is not specified, a runtime deployment or undeployment is performed, so that the application becomes available or unavailable in the running console. If the console instance is not currently running, the operation is automatically deferred.

`-a|--adminpassword` Specify that the administrator password should be changed, when used with password subcommand. You are prompted for a new password, which must be 8 to 32 characters.

`-a|--application` Specify the application name, when used with subcommands other than password subcommand.

`-d|--detail` Specify that configuration details of each resource should be displayed.

`-f` Specify the fully qualified path name to a file containing one or more password property values. See the description of the password_file argument.

`-k|--keypassword` Specify that the keystore password should be changed. You are prompted for a new password, which must be 8 to 32 characters.

`-l|--library` Specify that the resource is a shared jar file.

`-m|--module` Specify that the resource is a JAAS login module.

`-n|--name` Specify the short-hand identifier name for the resource. If omitted, the identifier name is derived from the full resource name.

`-o|--option` Specify the name and value of a login module option property, separated by the equals character.

`-p|--property` Specify that the resource is one or more shared service properties.

`-s|--service` Specify the name of the JAAS login service definition. If omitted, the default console login service definition is assumed.

`-t|--trustpassword` Specify that the truststore password should be changed. You are prompted for a new password, which must be 8 to 32 characters.

`-x|--context` Specify the web application context path name under which the application is deployed.

Arguments

app_path The fully qualified file system path to the web application installation directory.

appname The application name. The name must be unique among all web applications registered with the console. It is also used as the name of the subdirectory under the console's pre-registration directory which contains

all the resource registration notification files for that application. Typically, the application package name, plugin identifier, or context path name is specified for the application name.

behavior The JAAS login module control flag behavior. Must be one of "optional", "required", "requisite", or "sufficient".

classname The fully qualified Java package class name of the JAAS login module. The specified class must be included in a shared jar file added to the console.

context The web application context path name under which the application is deployed. With the .reg suffix, the context forms the file name of the registration notification file for that application.

id The short-hand identifier name for a jar file or login module resource to be added or removed. The identifier name must be unique among the resources shared for a given application name. With the .reg suffix, it forms the file name of the registration notfication file for that resource.

jarpath The fully qualified file system path to the jar file resource. When the resource is added to the console, its path is included in the classpath of the console's shared class loader.

option The JAAS login module option property name.

property The shared service property name.

password_file The fully qualified path to a password text file that contains the new administrator, keystore, and truststore passwords in property file format. The administrator password is specified using the "adminpassword" property. The keystore password is specified using the "keypassword" property. The truststore password is specified using the "trustpasswd" property. At least one password property must be contained in the password file.

value The login module option or shared service property value. If the value contains white space, it must be quoted.

Examples The following command adds a jar file to be shared in the console:

```
wcadmin add -l -a myapp_1.0 -n wbem /usr/sadm/lib/wbem.jar
```

The following command deploys a new web application:

```
wcadmin deploy -a myapp_1.0 -x myapp /opt/SUNWmyapp/myapp
```

The following command reloads an existing web application:

```
wcadmin reload -x myapp
```

The following command undeploys a web application at the next server restart:

```
wcadmin undeploy -D -a myapp_1.0 -x myapp
```

The following command lists all the deployed web applications in the console. If the status field is "running", the web application is available. If the status field is "stopped", the web application is disabled and is not available. If all web applications are "stopped", this typically indicates the console web server instance is not running.

```
wcadmin list -a
```

The following command removes a shared jar file:

```
wcadmin remove -l -a myapp_1.0 -n wbem
```

The following command changes passwords that are specified in a file:

```
wcadmin password -f /home/mydir/console-passwords
```

Exit Status The following exit values are returned:

`0` Subcommand succeeded without error

`1` Usage error: missing or malformed arguments

`2` Fatal error: subcommand failed with one or more errors

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWmcon
Interface Stability	Committed

See Also `attributes(5)`, `environ(5)`

Name whodo – who is doing what

Synopsis

```
/usr/sbin/whodo [-h] [-l] [user]
```

Description The `whodo` command produces formatted and dated output from information in the `/var/adm/utmpx` and `/proc/pid` files.

The display is headed by the date, time, and machine name. For each user logged in, device name, user-ID and login time is shown, followed by a list of active processes associated with the user-ID. The list includes the device name, process-ID, CPU minutes and seconds used, and process name.

If *user* is specified, output is restricted to all sessions pertaining to that user.

Options The following options are supported:

`-h` Suppress the heading.

`-l` Produce a long form of output. The fields displayed are: the user's login name, the name of the tty the user is on, the time of day the user logged in (in *hours*:*minutes*), the idle time — that is, the time since the user last typed anything (in *hours*:*minutes*), the CPU time used by all processes and their children on that terminal (in *minutes*:*seconds*), the CPU time used by the currently active processes (in *minutes*:*seconds*), and the name and arguments of the current process.

Examples **EXAMPLE 1** Using the whodo Command

The command:

```
example% whodo
```

produces a display like this:

```
Tue Mar 12 15:48:03 1985
bailey
tty09    mcn       8:51
    tty09   28158    0:29 sh

tty52    bdr      15:23
    tty52   21688    0:05 sh
    tty52   22788    0:01 whodo
    tty52   22017    0:03 vi
    tty52   22549    0:01 sh

xt162    lee      10:20
    tty08    6748    0:01 layers
    xt162    6751    0:01 sh
    xt163    6761    0:05 sh
    tty08    6536    0:05 sh
```

Environment Variables

If any of the LC_* variables (LC_CTYPE, LC_MESSAGES, LC_TIME, LC_COLLATE, LC_NUMERIC, and LC_MONETARY) (see environ(5)) are not set in the environment, the operational behavior of tar(1) for each corresponding locale category is determined by the value of the LANG environment variable. If LC_ALL is set, its contents are used to override both the LANG and the other LC_* variables. If none of the above variables is set in the environment, the "C" (U.S. style) locale determines how whodo behaves.

LC_CTYPE — Determines how whodo handles characters. When LC_CTYPE is set to a valid value, whodo can display and handle text and filenames containing valid characters for that locale. The whodo command can display and handle Extended Unix code (EUC) characters where any individual character can be 1, 2, or 3 bytes wide. whodo can also handle EUC characters of 1, 2, or more column widths. In the "C" locale, only characters from ISO 8859-1 are valid.

LC_MESSAGES — Determines how diagnostic and informative messages are presented. This includes the language and style of the messages, and the correct form of affirmative and negative responses. In the "C" locale, the messages are presented in the default form found in the program itself (in most cases, U.S. English).

LC_TIME — Determines how whodo handles date and time formats. In the "C" locale, date and time handling follow the U.S. rules.

Exit Status

The following exit values are returned:

0 — Successful completion.

non-zero — An error occurred.

Files

/etc/passwd — System password file

/var/adm/utmpx — User access and administration information

/proc/pid — Contains PID

Attributes

See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also

ps(1), who(1), attributes(5), environ(5)

Name wpad – WPA and WPA2 protocol daemon

Synopsis `/usr/lib/inet/wpad [-i` *interface*`] [-k` *pre_shared_key_name*`]`

Description The `wpad` daemon provides common client functionality for the WiFi Protected Access (WPA) versions 1 and 2, as defined by IEEE802.11i standard. WPA was created by the WiFi Alliance, an industry trade group. WPA implements the majority of the IEEE 802.11i standard, and was intended as an intermediate measure to take the place of Wired Equivalent Privacy (WEP) while 802.11i was prepared. WPA2 implements the full standard.

`wpad` provides the following WPA/IEEE 802.11i features:

- WPA-PSK ("WPA-Personal")
- Key management for CCMP, TKIP, WEP104, WEP40

Stop and start the `wpad` daemon using `dladm(1M)`. Use:

```
# dladm connect-wifi
```

...to start the `wpad` daemon. Use:

```
# dladm disconnect-wifi
```

...to stop the daemon.

Options The following options are supported:

`-i` *interface*
Specify a WiFi Link interface to start the `wpad` daemon.

`-k` *pre_shared_key_name*
Specify the pre-shared key used for the WiFi Link.

Examples **EXAMPLE 1** Starting the `wpad` Daemon on Specific WiFi Link

To create the WPA key `psk`, enter the following command:

```
# dladm create-secobj -c wpa psk
```

To use key `psk` to connect to ESSID `wlan` on link `ath0`, enter the following command:

```
# dladm connect-wifi -k psk -e wlan ath0
```

EXAMPLE 2 Stopping the `wpad` Daemon on Specific WiFi Link

To stop the daemon on the link `ath0`, enter:

```
# dladm disconnect-wifi ath0
```

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	service/network/wpa
Interface Stability	Uncommitted

See Also svcs(1), dladm(1M), svcadm(1M), attributes(5), smf(5)

Notes The wpad service is managed by the service management facility, smf(5), under the service identifier:

svc:/network/wpa:<*link*>

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using svcadm(1M). The service's status can be queried using the svcadm(1M) command.

Name wracct – write extended accounting records for active processes and tasks

Synopsis

```
/usr/bin/wracct -i id_list [-t record_type]
     {process | task}
```

Description The `wracct` utility allows the administrator to invoke the extended accounting system, if active, to write intermediate records representing the resource usage of a selected set of processes or tasks. For tasks, a *record_type* option is also supported, allowing the administrator to request the writing of:

- an interval record, which reflects task usage since a previous interval record (or since task creation if there is no interval record), or
- a partial record, which reflects usage since task creation.

Options The following options are supported:

`-i` *id_list* Select the IDs of the tasks or processes to write records for. Specify *id_list* as a comma- or space-separated list of IDs, presented as a single argument. For some shells, this requires appropriate quoting of the argument.

`-t` *record_type* Select type of record to write for the selected task or process. For tasks, *record_type* can be `partial` or `interval`. `partial` is the default type, and the only type available for process records.

Operands The following operands are supported:

`process` Treat the given ID as a process ID for the purposes of constructing and writing an extended accounting record.

`task` Treat the given ID as a task ID for the purposes of constructing and writing an extended accounting record.

Examples **EXAMPLE 1** Writing a Partial Record

Write a partial record for all active `sendmail` processes.

```
# /usr/bin/wracct -i "`pgrep sendmail`" process
```

EXAMPLE 2 Writing an Interval Record

Write an interval record for the task with `ID 182`.

```
# /usr/bin/wracct -t interval -i 182 task
```

Exit Status The following exit values are returned:

`0` Successful completion.

`1` An error occurred.

`2` Invalid command line options were specified.

3 Pertinent components of extended accounting facility are not active.

Files

`/var/adm/exacct/task` Extended accounting task files.

`/var/adm/exacct/proc` Extended accounting data files.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also

`acctadm(1M)`, `attributes(5)`

Name wusbadm – administer wireless USB hosts and devices

Synopsis

```
wusbadm list [-h | -d] [-o field[,...]]

wusbadm associate [-h host-id] [[-c [-f]] | -n] [-o]

wusbadm remove-dev [[-d dev-id] | [-h host-id]] [-f]

wusbadm remove-host [-h host-id] [-f]

wusbadm enable-host [-h host-id]

wusbadm disable-host [-h host-id]
```

Description The `wusbadm` command provides a command line interface to administer wireless USB hosts and devices, including listing hosts and devices information, associating the host with the device, removing host or device information from the system, and enabling or disabling hosts.

Before connecting a wireless USB device to a host for the first time, a user needs to set up the association information between them by running the wusbadm associate subcommand. Following this, the user can connect or disconnect the device by simply turning on or off the device radio (perhaps a button on the device, depending on the manufacturer). The device radio's turning on and off are analogous to the hotplugging of wired USB devices.

The association information created by the `associate` subcommand is maintained in the non-volatile memory of the device and the host. On the host, it can be removed by the `remove-dev` or `remove-host` subcommands. On the device, it can be overwritten by another association. For a device is associated with multiple hosts, the way that the device prioritizes or updates its multiple records of association depends on the manufacturer.

Each `wusbadm` subcommand operates on one of the following objects:

host-id
: A two-digit number (in the range from 01 to 99) that uniquely identifies a wireless USB host on a system. It is generated when the `wusb` service (see `NOTES` section) is successfully enabled and finds the host instance for the first time. The number is maintained until removed by `remove-host` subcommand.

dev-id
: A five-digit number that uniquely identifies a wireless USB device associated with a wireless USB host. The first two digits are the host-id of the wireless USB host with which the device is associated. The last three-digit number (in the range from 001 to 999) is used to differentiate devices associated with the same host. In the five-digit number, the first two digits and the last three are separated by a dot.

  dev-id is generated during the device association process. It is maintained for the device until removed by the `remove-dev` subcommand or until updated by another association between the same host and device.

Sub-commands The following subcommands are supported. Except for the `list` subcommand, each subcommand displays subcommand-specific usage information if you run it without any options or operands.

`list` [`-h` | `-d`] [`-o` *field*[,...]]

List wireless USB hosts and devices on a system, displaying the ID, state, and type for all hosts and devices. By default, `list` will list all hosts and devices and all fields. Each host and its devices will be displayed as a group. This subcommand supports the following options.

`-o` *field*[,...], `--output=`*field*[,...]

A case-insensitive, comma-separated list of output fields to display. The field name must be one of the fields listed below, or the special value `all` to display all fields. By default (without `-o`), `list` displays all fields.

`ID`

The *host-id* or *dev-id*.

`TYPE`

The `host` or `device` types.

For `host`, the types include `whci` (on-board host) and `hwa` (hot-pluggable host).

For `device`, the types include `kbd`, `mouse`, `storage`, `printer`, `dwa` (wireless USB hub), `audio`, `video`, and so forth.

`STATE`

There are the following states for the host:

`enabled`

The host is ready to work or is already working, including performing association, connecting devices, performing data communication, and so forth.

`disabled`

The host is not ready to work with any devices and no devices are connected to the host. It might be stopped by a `disable-host` subcommand, or the host might not be available because it is physically unplugged or because of a driver detach.

`disconnected`

The host is not attached to the system. An `hwa` device is in this state after it is unplugged from the USB port on the system.

There are the folllowing states for the device:

`connected`

The device is connected with a host and ready to be opened, or it is already opened and working. By default, the device tries to get into this state after the association is complete and its radio is turned on.

`disconnected`
The device is not connected to a host or not ready to be opened yet. The device might be in this state because its radio is out of range, power is off, hardware problems, and so forth.

`-h`, `--host`
List the wireless USB hosts only.

`-d`, `--device`
List the wireless USB devices only.

`associate` [`-h` *host-id*] [[`-c` [`-f`]] | `-n`] [`-o`]
Designate the host to start an association process. Association is the initial step before a wireless USB device can be connected with a wireless USB host.

There are two association models:

Cable association
A user connects the device and host with a USB cable first, and then run this subcommand to designate the host to setup the association information with the device. After the association is in effect, the cable is no longer needed in the subsequent connections between the same host and the device.

Numeric association
A user turns on the device radio and runs this subcommand to designate the host to talk to the device. A short number is then displayed on both host and device. The user compares the values of the numbers and confirms on both the host and the device.

Following a successful association, the associated USB host and device are able to proceed with the wireless connection process. By default, the association information will be kept both on the host and the device until it is removed or overwritten.

If there are multiple devices available for association, this subcommand will list all of them, enabling a user to choose among them. This subcommand has the following options.

`-h` *host-id*, `-–host` *host-id*
Specify the host for which the association will be done. If this option is not specified, this subcommand lists all enabled hosts for users to choose.

`-c`, `-–cable`
Start the cable association process. A user plugs the wireless USB device to the host and runs the associate subcommand with this option.

`-n`, `-–numeric`
Start the numeric association process. This subcommand prompts the user to compare the number displayed on the host and the device.

If neither of the preceding two association model options (`-n` or `-c`) is specified, this subcommand prompts the user to specify one of the following association model options.

`-f`, `--force`
Start the cable association process. A user plugs the wireless USB device to the host and runs the associate subcommand with this option.

`-o`, `--onetime`
Indicate that this association is for a one-time connection. That is, after the association, if the device is connected and then disconnected, the association information for this device will be removed from the host system. A user would need to perform another association for the next connection.

`remove-dev` [[`-d` *dev-id*] | [`-h` *host-id*]][`-f`]
Remove the association information of the wireless USB device from the system. After the removal, the device cannot be connected with the host until the user runs the `associate` subcommand again, for the host and device. This subcommand has the following options.

`-d`, `--device=`*dev-id*
Remove the association information of the wireless USB device specified by *dev-id*.

`-h` *host-id*, `--host=`*host-id*
Remove the association information of all the wireless USB devices associated with the host specified by *host-id*.

`-f`, `--force`
Perform the removal without asking for confirmation. If the device is being connected with the host, then this subcommand will force it to disconnect.

`remove-host` [`-h` *host-id*] [`-f`]
Remove the host information from the system, including *host-id* and the association information of all the devices associated with the host. This subcommand is used most often for removing the temporarily used hot-pluggable wireless USB host, for example, a `hwa` dongle. The host can be brought back by being re-enumerated, for example, physically hot-plugging a `hwa` dongle. The host-id will then be updated and no device association information can be restored. It is not recommended to remove a on-board host. This subcommand has the following options.

`-h` *host-id*, `--host=`*host-id*
Specifies the *host-id* to be removed.

`-f`, `--force`
Perform the removal without asking for confirmation. If there are one or more devices connected with the host, then force them to disconnect.

`enable-host` [`-h` *host-id*]
Take the host to the enabled state. By default, the host is in the enabled state. This subcommand has the following option.

`-h` *host-id*, `--host=`*host-id*
Specifies the *host-id* to be enabled.

`disable-host` [`-h` *host-id*] [`-f`]
Take the host to the disabled state. The *host-id* and all the association information of the host are maintained. Issuing an `enable-host` subcommand brings the host back to the enabled state. This subcommand has the following options.

`-h` *host-id*, `--host=`*host-id*
Specifies the *host-id* to be disabled.

`-f`, `--force`
Perform the disable operation without asking for confirmation. If there are one or more devices connected with the host, this option forces them to disconnect.

Examples

EXAMPLE 1 Listing All Hosts and Devices

The following command lists all wireless USB hosts and devices.

```
# wusbadm list
01      enabled         hwa
01.001  connected       mouse
01.002  connected       kbd
02      enabled         whci
02.001  connected       printer
02.002  disconnected    storage
03      disabled        hwa
03.001  disconnected    storage
03.002  disconnected    dwa
```

EXAMPLE 2 Associating to a Device Using Cable

The following command associates a device to a specific host (host-id `01`), using the cable association approach.

```
# wusbadm associate -h 01 -c
Associate a device with host (01) via cable.
Continue (yes/no)?
```

EXAMPLE 3 Removing a Device's Association

The following command removes a device's association information from the host system.

```
# wusbadm remove-dev -d 01.002
Remove the information of device (01.002) from system.
This device can not be connected with the host until it is associated
again. Continue (yes/no)?
```

EXAMPLE 4 Removing Associations for All Devices

The following command removes the association information for all devices associated with a specific host.

```
# wusbadm remove-dev -h 02
Remove the information of all the devices associated with host (02)
```

EXAMPLE 4 Removing Associations for All Devices *(Continued)*

```
from the system.
All the devices associated with the host cannot be connected with it
until they are associated again. Continue (yes/no)?
```

Exit Status

The following exit values are returned:

`0`
Successful operation.

`1`
Error: the operation failed. For example, a device failed to associate with a host.

`2`
Usage error.

Attributes

See `attributes(5)` for descriptions of the following attributes:

`/usr/sbin`

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/io/usb
Interface Stability	Committed

See Also

`attributes(5)`, `hwahc(7D)`, `usba(7D)`

Notes

The `wusb` (wireless USB administration) service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/system/wusb:default
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. The service's status can be queried using the `svcs(1)` command.

The `wusb` service is implemented by the `wusbd` daemon, a private interface. As with the `wusb` service, the daemon is started by the SMF. Specify the daemon with the service instance:

```
svc:/system/wusbd:default
```

The `wusbd` daemon should not be invoked directly.

Name ypbind – NIS binder process

Synopsis

```
/usr/lib/netsvc/yp/ypbind [-broadcast | -ypset | -ypsetme]
```

Description NIS provides a simple network lookup service consisting of databases and processes. The databases are stored at the machine that runs an NIS server process. The programmatic interface to NIS is described in ypclnt(3NSL). Administrative tools are described in ypinit(1M), ypwhich(1), and ypset(1M). Tools to see the contents of NIS maps are described in ypcat(1), and ypmatch(1).

`ypbind` is a daemon process that is activated at system startup time from the `svc:/network/nis/client:default` service. By default, it is invoked as `ypbind -broadcast`. `ypbind` runs on all client machines that are set up to use NIS. The function of `ypbind` is to remember information that lets all NIS client processes on a node communicate with some NIS server process. `ypbind` must run on every machine which has NIS client processes. The NIS server may or may not be running on the same node, but must be running somewhere on the network.

The SMF service `svc:/network/nis/client` has the following properties in the `config` property group:

```
config.use_broadcast
config.use_ypsetme
```

The information `ypbind` remembers is called a *binding* — the association of a domain name with a NIS server. The process of binding is driven by client requests. As a request for an unbound domain comes in, if started with the `-broadcast` option, the `ypbind` process broadcasts on the net trying to find an NIS server, that is, a `ypserv` process serving the domain with a name the same as (case sensitive) the name of the domain in the client request. Since the binding is established by broadcasting, there must be at least one NIS server on the net. If started without the `-broadcast` option, `ypbind` process steps through the list of NIS servers that was created by `ypinit -c` for the requested domain. There must be an NIS server process on at least one of the hosts in the NIS servers file. It is recommended that you list each of these NIS servers by name and numeric IP address in `/etc/hosts`. Though the practice is not recommended, NIS allows you to list servers by numeric address only, bypassing `/etc/hosts`. In such a configuration, ypwhich(1) returns a numeric address instead of a name.

Once a domain is bound by `ypbind`, that same binding is given to every client process on the node. The `ypbind` process on the local node or a remote node may be queried for the binding of a particular domain by using the ypwhich(1) command.

If `ypbind` is unable to speak to the NIS server process it is bound to, it marks the domain as unbound, tells the client process that the domain is unbound, and tries to bind the domain once again. Requests received for an unbound domain will wait until the requested domain is bound. In general, a bound domain is marked as unbound when the node running the NIS server crashes or gets overloaded. In such a case, `ypbind` will try to bind to another NIS server using the process described above.`ypbind` also accepts requests to set its binding for a

particular domain. The request is usually generated by the ypset(1M) command. In order for ypset to work, ypbind must have been invoked with flags -ypset or -ypsetme.

Interaction with Location Profiles

NIS configuration and activation is managed in Location profiles (refer to netcfg(1M) for more information about location profiles). These profiles are either fixed, meaning the network configuration is being managed in the traditional way, or reactive, meaning the network configuration is being managed automatically, reacting to changes in the network environment according to policy rules specified in the profiles.

When a fixed location (there can currently be only one, the DefaultFixed location) is active, changes made to the SMF repository will be applied to the location when it is disabled, and thus will be restored if that location is later re-enabled.

When a reactive location is active, changes should not be applied directly to the SMF repository; these changes will not be preserved in the location profile, and will thus be lost if the location is disabled, or if the system's network configuration, as managed by svc:/network/physical:default and svc:/network/location:default, is refreshed or restarted. Changes should instead be applied to the location itself, using the netcfg(1M) command; this will save the change to the location profile repository, and will also apply it to the SMF repository (if the change is made to the currently active location).

The presence or absence of nis in the nameservices property of a location profile will determine whether or not svc:/network/nis/client:default is enabled. The nis-nameservice-servers property may be empty, indicating that -broadcast should be enabled, or it may contain the list of servers to which the client may bind.

Options

-broadcast
: Send a broadcast datagram using UDP/IP that requests the information needed to bind to a specific NIS server. This option is analogous to ypbind with no options in earlier Sun releases and is recommended for ease of use.

  Enabling the SMF property config.use_broadcast enables -broadcast.

-ypset
: Allow users from any remote machine to change the binding by means of the ypset command. By default, no one can change the binding. This option is insecure.

-ypsetme
: Only allow root on the local machine to change the binding to a desired server by means of the ypset command. ypbind can verify the caller is indeed a root user by accepting such requests only on the loopback transport. By default, no external process can change the binding.

  Enabling the SMF property config.use_ypsetme enables -ypsetme.

Files `/var/yp/binding/`*ypdomain*`/ypservers`
Lists the servers to which the NIS client is allowed to bind.

`/etc/inet/hosts`
File in which it is recommended that NIS servers be listed.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/network/nis

See Also `svcs(1)`, `ypcat(1)`, `ypmatch(1)`, `ypwhich(1)`, `ifconfig(1M)`, `netcfg(1M)`, `svcadm(1M)`, `ypinit(1M)`, `ypset(1M)`, `ypclnt(3NSL)`, `hosts(4)`, `ypfiles(4)`, `attributes(5)`, `smf(5)`

Notes `ypbind` supports multiple domains. The `ypbind` process can maintain bindings to several domains and their servers, the default domain is the one specified by the `domainname(1M)` command at startup time.

The `-broadcast` option works only on the UDP transport. It is insecure since it trusts "any" machine on the net that responds to the broadcast request and poses itself as an NIS server.

The `ypbind` service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/network/nis/client:default
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. The service's status can be queried using the `svcs(1)` command.

Name ypinit – set up NIS client

Synopsis `/usr/sbin/ypinit [-c] [-m] [-s` *master_server*`]`

Description `ypinit` can be used to set up an NIS client system. You must be the superuser to run this command. This script need not be used at all if `ypbind(1M)` is started with the `-broadcast` option (it is invoked with this option from the `svc:/network/nis/client:default service`).

Normally, `ypinit` is run only once after installing the system. It may be run whenever a new NIS server is added to the network or an existing one is decommissioned.

`ypinit` prompts for a list of NIS servers to bind the client to; this list should be ordered from the closest to the furthest server. It is recommended that you list each of these NIS servers by name and numeric IP address in `/etc/hosts`. Though the practice is not recommended, NIS allows you to list servers by numeric address only, bypassing `/etc/hosts`. In such a configuration, `ypwhich(1)` returns a numeric address instead of a name.

`ypinit` stores the list of servers to which a client can bind in the file `/var/yp/binding/`*domain*`/ypservers`. This file is used by `ypbind` when run without the `-broadcast` option.

Interaction with Location Profiles

NIS client configuration is managed in Location profiles (refer to `netcfg(1M)` for more information about location profiles). These profiles are either fixed, meaning the network configuration is being managed in the traditional way, or reactive, meaning the network configuration is being managed automatically, reacting to changes in the network environment according to policy rules specified in the profiles.

When a fixed location (there can currently be only one, the `DefaultFixed` location) is active, changes made to the SMF repository will be applied to the location when it is disabled, and thus will be restored if that location is later re-enabled.

When a reactive location is active, changes should not be applied directly to the SMF repository; these changes will not be preserved in the location profile, and will thus be lost if the location is disabled, or if the system's network configuration, as managed by `svc:/network/physical:default` and `svc:/network/location:default`, is refreshed or restarted. Changes should instead be applied to the location itself, using the `netcfg(1M)` command; this will save the change to the location profile repository, and will also apply it to the SMF repository (if the change is made to the currently active location).

NIS client configuration is stored the `default-domain` and `nis-nameservice-servers` properties of a location profile.

Options

`-c`
Set up a `ypclient` system.

`-m`
Build a master `ypserver` data base.

`-s` *master_server*
Slave data base. *master_server* must be the same master configured in the YP maps and returned by the `ypwhich -m` command.

Files `/etc/hosts`
File in which it is recommended that NIS servers be listed.

`/var/yp/binding/`*domain*`/ypservers`
Lists the servers to which the NIS client is allowed to bind.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/network/nis

See Also `svcs(1)`, `ypwhich(1)`, `netcfg(1M)`, `svcadm(1M)`, `ypbind(1M)`, `sysinfo(2)`, `hosts(4)`, `attributes(5)`, `smf(5)`

Notes The `NIS client` service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/network/nis/client:default
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. The service's status can be queried using the `svcs(1)` command.

Bugs `ypinit` sets up the list of NIS servers only for the current domain on the system when it is run, that is, the domain returned by the `SI_SRPC_DOMAIN` command to `sysinfo(2)`. Care should be taken to ensure that this is the same as the desired domain for NIS client processes.

Name ypmake – rebuild NIS database

Synopsis

```
cd /var/yp ; make [map]
```

Description The file called `Makefile` in `/var/yp` is used by `make(1S)` to build the Network Information Service (NIS) database. With no arguments, `make` creates `dbm` databases for any NIS maps that are out-of-date, and then executes `yppush(1M)` to notify slave databases that there has been a change.

If you supply a *map* on the command line, `make` will update that map only. Typing `make passwd` will create and `yppush` the password database (assuming it is out of date). Likewise, `make ipnodes` and `make networks` will create and `yppush` the `ipnodes` and `network` files, `$(INETDIR)/ipnodes` and `$(DIR)/networks`.

There are four special variables used by `make`: `DIR`, which gives the directory of the source files; `NOPUSH`, which when non-null inhibits doing a `yppush` of the new database files; `INETDIR`, which gives the directory of the `ipnodes` source file; and `DOM`, which is used to construct a domain other than the master's default domain. The default for `DIR` is `/etc`, and the default for `INETDIR` is `/etc/inet`. The default for `NOPUSH` is the null string.

Refer to `ypfiles(4)` and `ypserv(1M)` for an overview of the NIS service.

If a NIS to LDAP (N2L) configuration file, `/var/yp/NISLDAPmapping`, is present, the NIS server components run in N2L mode. In N2L mode, the server components use a new set of map files with an LDAP-prefix, based on the LDAP DIT. In N2L mode, authoritative NIS information is obtained from the DIT. The NIS source files and `ypmake` have no role, and they should not be used. If `ypmake` is accidentally run, then the server components will detect this, and will log a warning message. For additional information, see `ypfiles(4)`.

Files

`/var/yp`	Directory containing NIS configuration files.
`/etc/inet/hosts`	System hosts file.
`/etc`	Default directory for source files.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	service/network/nis
Interface Stability	Committed

See Also `make(1S)`, `makedbm(1M)`, `ypbind(1M)`, `yppush(1M)`, `ypserv(1M)`, `ypclnt(3NSL)`, `NISLDAPmapping(4)`, `ypfiles(4)`, `ypserv(4)`

Notes The NIS makefile is only used when running the ypserv(1M) server to provide NIS services. See ypfiles(4) for more details.

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same. Only the name has changed. The name Yellow Pages is a registered trademark in the United Kingdom of British Telecommunications plc, and may not be used without permission.

Name ypmap2src – convert NIS maps to NIS source files

Synopsis

```
/usr/lib/netsvc/yp/ypmap2src [-t]
     [ [-c custom-map-name]]... [-d domain] -o output-directory
     [ [ source-file]]...
```

Description Use the `ypmap2src` utility to convert standard NIS maps to approximations of the equivalent NIS source files. This utility functions like the reverse of `ypmake(1M)`.

The primary use for `ypmap2src` is to convert from a NIS server that uses the NIS to LDAP(N2L) transition mechanism, which does not use NIS source files, to traditional NIS, where source files are required. The `ypmap2src` utility is also used by NIS administrators who wish to discover the contents of NIS maps for which the sources are not available.

Generally, this operation is not necessary. More often, administrators will switch from traditional NIS to N2L in anticipation of the eventual transition to LDAP naming. When this switch is made, authoritative information is moved into the LDAP DIT, and the NIS sources have no further role. N2L supports NIS clients until such time as they can be converted to LDAP, and the NIS service suspended.

The `ypmap2src` utility does not guarantee that the files that are generated are identical to the original NIS source files. Some information might have been thrown away by `ypmake` and cannot be recovered. N2L also might have updated the maps to reflect changes made by LDAP clients. It is essential that the sources generated are checked to confirm no problems have occurred.

Per entry comment fields, from existing source files, are not merged into source files generated by `ypmap2src`. If a user wishes N2L to maintain comment information, then the `NISLDAPmapping` configuration file should be modified so that the comment fields are mapped into LDAP. This will ensure that the comments are visible to native LDAP clients and present in the N2L map files.

When `ypmap2src` is run, it will take up-to-date comments from the map file and insert them into the NIS source file generated.

Handling Custom Maps `ypmap2src` only knows about the standard NIS maps and standard source to map conversion. If an advanced user has changed these, that is, the user has modified the NIS makefile, the equivalent changes must also be made to the `ypmap2src` script.

Options `ypmap2src` supports the following options:

`-c` Specifies that *custom-map-name* should be converted to a source file by running `makedbm -u` on it. This is a short cut so that simple custom maps can be handled without editing `ypmap2src`.

`-d` *domain-name* Specifies the domain to convert. The *domain-name* can be a fully qualified file path, such as `/var/yp/a.b.c`, or just a domain name, a.b.c. In the latter case, `ypmaptosrc` looks in `/var/yp` for the domain directory.

`-o` *dest* — Specifies the destination directory for the converted files. A directory other than `/etc` should be specified. The maps generated are copied to the correct location, `/etc`, `/etc/security` or other source directory, as appropriate.

`-t` — Specifies that traditional NIS maps, without N2L's `LDAP_` prefix, should be converted. By default, maps with the `LDAP_` prefix are converted.

Operands `ypmap2src` supports the following operands:

source-file — Lists the standard source files to convert. If this option is not given, then all the standard source files, plus any custom files pecified by the `-c` option, are converted.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	service/network/nis
Interface Stability	Obsolete

See Also `ypmake(1M)`, `ypserv(1M)`, `NISLDAPmapping(4)`, `attributes(5)`

Oracle Solaris Administration: Naming and Directory Services

Name yppoll – return current version of a NIS map at a NIS server host

Synopsis

```
/usr/sbin/yppoll [-d ypdomain] [-h host] mapname
```

Description The `yppoll` command asks a `ypserv( )` process what the order number is, and which host is the master NIS server for the named map.

Options

`-d` *ypdomain* Use *ypdomain* instead of the default domain.

`-h` *host* Ask the `ypserv` process at *host* about the map parameters. If *host* is not specified, the NIS server for the local host is used. That is, the default host is the one returned by `ypwhich(1)`.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/network/nis

See Also `ypwhich(1)`, `ypfiles(4)`, `attributes(5)`

Name yppush – force propagation of changed NIS map

Synopsis

```
/usr/lib/netsvc/yp/yppush [-v] [-h host] [-d domain]
     [-p #parallel-xfrs] mapname
```

Description `yppush` copies a new version of a Network Information Service (NIS) map from the master NIS server to the slave NIS servers. It is normally run only on the master NIS server by the `Makefile` in `/var/yp` after the master databases are changed. It first constructs a list of NIS server hosts by reading the NIS `ypservers` map within the *domain*. Keys within the `ypservers` map are the ASCII names of the machines on which the NIS servers run.

A "transfer map" request is sent to the NIS server at each host, along with the information needed by the transfer agent (the program that actually moves the map) to call back the `yppush`. When the attempt has completed (successfully or not), and the transfer agent has sent `yppush` a status message, the results can be printed to `stdout`. Messages are also printed when a transfer is not possible, for instance, when the request message is undeliverable, or when the timeout period on responses has expired.

Refer to `ypfiles(4)` and `ypserv(1M)` for an overview of the NIS service.

Options The following options are supported:

`-d` *domain*	Specifies a *domain*.
`-h` *host*	Propagates only to the named *host*.
`-p` *#parallel-xfrs*	Allows the specified number of map transfers to occur in parallel.
`-v`	Verbose. This prints messages when each server is called, and for each response. If this flag is omitted, only error messages are printed.

Files

`/var/yp`	Directory where NIS configuration files reside.
`/var/yp/`*domain*`/ypservers.` `{`*dir*`,` *pag* `}`	Map containing list of NIS servers to bind to when running in server mode.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	service/network/nis

See Also `ypserv(1M)`, `ypxfr(1M)`, `ypfiles(4)`, `attributes(5)`

Notes The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed. The name Yellow Pages is a registered trademark in the United Kingdom of British Telecommunications PLC, and must not be used without permission.

Bugs In the current implementation (version 2 NIS protocol), the transfer agent is `ypxfr(1M)`, which is started by the `ypserv` program. If `yppush` detects that it is speaking to a version 1 NIS protocol server, it uses the older protocol, sending a version 1 `YPPROC_GET` request and issues a message to that effect. Unfortunately, there is no way of knowing if or when the map transfer is performed for version 1 servers. `yppush` prints a message saying that an "old-style" message has been sent. The system administrator should later check to see that the transfer has actually taken place.

Name ypserv, ypxfrd – NIS server and binder processes

Synopsis

```
/usr/lib/netsvc/yp/ypserv [-dv] [-i | -I] [-r | -R]

/usr/lib/netsvc/yp/ypxfrd
```

Description The Network Information Service (NIS) provides a simple network lookup service consisting of databases and processes. The databases are `ndbm` files in a directory tree rooted at `/var/yp`. See `ndbm(3C)`. These files are described in `ypfiles(4)`. The processes are `/usr/lib/netsvc/yp/ypserv`, the NIS database lookup server, and `/usr/lib/netsvc/yp/ypbind`, the NIS binder. The programmatic interface to the NIS service is described in `ypclnt(3NSL)`. Administrative tools are described in `yppoll(1M)`, `yppush(1M)`, `ypset(1M)`, `ypxfr(1M)`, and `ypwhich(1)`. Tools to see the contents of NIS maps are described in `ypcat(1)`, and `ypmatch(1)`. Database generation and maintenance tools are described in `ypinit(1M)`, `ypmake(1M)`, and `makedbm(1M)`.

The `ypserv` utility is a daemon process typically activated at system startup from `svc:/network/nis/server:default`. Alternatively, you can, as the root user, start NIS services using `ypstart(1M)` from the command-line. `ypserv` runs only on NIS server machines with a complete NIS database. You can halt all NIS services using the `ypstop(1M)` command.

The `ypxfrd` utility transfers entire NIS maps in an efficient manner. For systems that use this daemon, map transfers are 10 to 100 times faster, depending on the map. To use this daemon, be sure `ypxfrd` is running on the master server. See `/usr/lib/netsvc/yp/ypstart`. `ypxfr` attempts to use `ypxfrd` first. If that fails, it prints a warning, then uses the older transfer method.

The `ypserv` daemon's primary function is to look up information in its local database of NIS maps.

The operations performed by `ypserv` are defined for the implementor by the *YP Protocol Specification*, and for the programmer by the header file <`rpcsvc/yp_prot.h`>.

Communication to and from `ypserv` is by means of RPC calls. Lookup functions are described in `ypclnt(3NSL)`, and are supplied as C-callable functions in the `libnsl(3LIB)` library. There are four lookup functions, all of which are performed on a specified map within some NIS domain: `yp_match(3NSL)`, `yp_first(3NSL)`, `yp_next(3NSL)`, and `yp_all(3NSL)`. The `yp_match` operation takes a key, and returns the associated value. The `yp_first` operation returns the first key-value pair from the map, and `yp_next` can be used to enumerate the remainder. `yp_all` ships the entire map to the requester as the response to a single RPC request.

The SMF service `svc:/network/nis/server` manages the configuration of the `ypserv` daemon.

A number of special keys in the DBM files can alter the way in which `ypserv` operates. The keys of interest are:

`YP_INTERDOMAIN`	The presence of this key causes `ypserv` to forward to a DNS server host lookups that cannot be satisfied by the DBM files.
`YP_SECURE`	This key causes `ypserv` to answer only questions coming from clients on reserved ports.
`YP_MULTI_`*hostname*	This is a special key in the form, `YP_MULTI_`*hostname addr1,...,addrN*. A client looking for *hostname* receives the closest address.

Two other functions supply information about the map, rather than map entries: `yp_order(3NSL)`, and `yp_master(3NSL)`. In fact, both order number and master name exist in the map as key-value pairs, but the server will not return either through the normal lookup functions. If you examine the map with `makedbm(1M)`, however, they are visible. Other functions are used within the NIS service subsystem itself, and are not of general interest to NIS clients. These functions include `do_you_serve_this_domain?`, `transfer_map`, and `reinitialize_internal_state`.

On start up, `ypserv` checks for the existence of the NIS to LDAP (N2L) configuration file `/var/yp/NISLDAPmapping`. If it is present then a master server starts in N2L mode. If the file is not present it starts in "traditional" (non N2L) mode. Slave servers always start in traditional mode.

In N2L mode, a new set of map files, with an `LDAP_` prefix, are generated, based on the contents of the LDAP DIT. The old map files, NIS source files and `ypmake(1M)` are not used.

It is possible that `ypmake(1M)` can be accidentally run in N2L mode. If the occurs, the old style map files are overwritten. That the map files are overwritten is harmless. However, any resulting `yppush(1M)` operation will push information based on the DIT rather than the source files. The user may not expect information based on the DIT. `ypserv` keeps track of the last modification date of the old style map files. If the map files have been updated, a warning is logged that suggests that the user call `yppush` directly instead of `ypmake`.

If a server attempts to run in N2L mode and a LDAP server cannot be contacted, it behaves as follows:

1. When `ypserv` is started, a warning will be logged.
2. When a NIS read access is made and the TTL entry has expired, a warning is logged.Information that is returned from the cache has not been updated.
3. When a NIS write access is made, a warning is logged. The cache will not be updated, and a NIS failure will be returned.

If `ypxfrd` is running in N2L mode and is asked to transfer a map, `ypxfrd` first checks whether the map is out of date. If the map is out of date, `ypxfrd` initiates an update from the DIT. `ypxfrd` cannot wait for the update to complete. If `ypxfrd` waited, the client end `ypxfr`

operation could time out. To prevent `ypxfrd` from timing out, the existing map is transferred from the cache. The most up to date map will be transferred on subsequent `ypxfrd` operations.

Options

ypserv `-d` The NIS service should go to the DNS for more host information. This requires the existence of a correct `/etc/resolv.conf` file pointing to a DNS server. This option turns on DNS forwarding regardless of whether or not the `YP_INTERDOMAIN` flag is set in the `hosts` maps. See makedbm(1M). In the absence of an `/etc/resolv.conf` file, `ypserv` complains, but ignores the `-d` option.

If enabled, the property `group/property config.service_dns` tells `ypserv` to enable the `-d` option.

`-i` If in N2L mode, initialize the NIS related parts of the `DIT` based on the current, non `LDAP_` prefixed, map files. The `LDAP_` prefixed maps are not created or updated. If you require that `LDAP_` prefixed maps be updated or created, then use the `-ir` option.

The `-i` option does not attempt to create any NIS domain or container objects. If any NIS domain or container objects have not already been created, then errors will occur, as entries are written to nonexistent containers.

`-I` Identical to `-i`, except that any missing domain and container objects are created.

`-r` If in N2L mode, then refresh the `LDAP_` prefixed map files based on the contents of the `DIT`.

`-ir` If both `-i` and `-r` are specified in N2L mode, then the `DIT` will first be initialized from the current non `LDAP_` prefixed map files. A new set of `LDAP_` prefixed maps will then be generated from the contents of the `DIT`. A new set of `LDAP_` prefixed maps is required when moving from traditional NIS to N2L mode NIS.

`-Ir` Identical to `-ir`, except that any missing domain and container objects are created.

`-v` Operate in the verbose mode, printing diagnostic messages to stderr.

When run with the `-i`, `-r`, `-I`, `-ir` or `-Ir` options, the `ypserv` command runs in the foreground and exits once map initialization has been completed. Once the `ypserv` command exits, the user knows the maps are ready and can restart `ypserv` and the other `yp` daemons by running `ypstart`(1M).

If there is a requirement to initialize the `DIT` from the NIS source files, which may have been modified since the maps were last remade, run `ypmake` before running `ypserv -i` or `ypserv -ir`. `ypmake` regenerated old style NIS maps. Then `ypserv -ir` dumps them into the `DIT`. When the `-ir` option is used, the `LDAP_` prefixe maps are also generated or updated. Since these maps will be more recent than the old style maps, `ypmake` will not be reported as erroneous when it is run.

Files

`/var/yp/securenets`	Defines the hosts and networks that are granted access to information in the served domain. It is read at startup time by both `ypserv` and `ypxfrd`.
`/var/yp/ypserv.log`	If the `/var/yp/ypserv.log` file exists when `ypserv` starts up, log information is written to it when error conditions arise.
`/var/yp/binding/domainname/ypservers`	Lists the NIS server hosts that `ypbind` can bind to.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	service/network/nis

See Also `svcs(1)`, `ypcat(1)`, `ypmatch(1)`, `ypwhich(1)`, `domainname(1M)`, `makedbm(1M)`, `svcadm(1M)`, `ypbind(1M)`, `ypinit(1M)`, `ypmake(1M)`, `yppoll(1M)`, `yppush(1M)`, `ypset(1M)`, `ypstart(1M)`, `ypstop(1M)`, `ypxfr(1M)`, `ndbm(3C)`, `ypclnt(3NSL)`, `libnsl(3LIB)`, `NISLDAPmapping(4)`, `securenets(4)`, `ypfiles(4)`, `ypserv(4)`, `attributes(5)`, `smf(5)`

Network Interface Guide

Oracle Solaris Administration: Naming and Directory Services

Notes `ypserv` supports multiple domains. The `ypserv` process determines the domains it serves by looking for directories of the same name in the directory `/var/yp`. It replies to all broadcasts requesting yp service for that domain.

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed. The name Yellow Pages is a registered trademark in the United Kingdom of British Telecommunications PLC, and must not be used without permission.

NIS uses `ndbm()` files to store maps. Therefore, it is subject to the 1024 byte limitations described in the USAGE and NOTES sections of the `ndbm(3C)` man page.

The NIS server service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/network/nis/server:default
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. The service's status can be queried using the `svcs(1)` command.

Name ypserv_resolv – DNS forwarding service daemon for NIS

Synopsis

```
ypserv_resolv [-v | -V] [-F [-C fd]] [-t xx] [-p yy]
```

Description `ypserv_resolv` is an auxiliary process that provides DNS forwarding service for NIS hosts requests to `ypserv` that is running in the NIS compatibility mode. It is generally started by invoking `ypserv(1M)` with the `-d` option. Although it is not recommended, `ypserv_resolv` can also be started independently with the options shown below.

This command requires that the `/etc/resolv.conf` file be setup for communication with a DNS nameserver. The `dig(1M)` utility can be used to verify communication with a DNS nameserver. See `resolv.conf(4)` and `dig(1M)`.

Options

`-F` Run in foreground.

`-C` *fd* Use *fd* for service xprt.

`-v` Verbose. Send output to the syslog daemon.

`-V` Verbose. Send output to stdout.

`-t` *xx* Use transport *xx*.

`-p` *yy* Use transient program# *yy*.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/network/nis

See Also `dig(1M)`, `resolv.conf(4)`, `attributes(5)`

Name ypset – point ypbind at a particular server

Synopsis

```
/usr/sbin/ypset [-d ypdomain] [-h host] server
```

Description In order to run `ypset`, `ypbind` must be initiated with the –`ypset` or –`ypsetme` options. See `ypbind(1M)`. `ypset` tells `ypbind` to get NIS services for the specified *ypdomain* from the `ypserv` process running on *server*. If *server* is down, or is not running `ypserv`, this might not be discovered until an NIS client process tries to obtain a binding for the domain. At this point, the binding set by `ypset` is tested by `ypbind`. If the binding is invalid, `ypbind` attempts to rebind for the same domain.

`ypset` is useful for binding a client node that is not on a broadcast net, or is on a broadcast net that is not running an NIS server host. It is also useful for debugging NIS client applications, for instance, where an NIS map exists only at a single NIS server host.

Where several hosts on the local net are supplying NIS services, `ypbind` can rebind to another host, even while you attempt to find out if the `ypset` operation succeeded. For example, if you enter the `ypset` command below, you might get the subsequent response from `ypwhich`:

```
example% ypset host1
example% ypwhich
host2
```

The sequence shown above is a function of the NIS subsystem's attempt to load-balance among the available NIS servers, and occurs when `host1` does not respond to `ypbind` because it is not running `ypserv` (or is overloaded), and `host2`, running `ypserv`, obtains the binding.

server indicates which NIS server to bind to, and must be specified as a name or an IP address. This works only if the node has a current valid binding for the domain in question and `ypbind` has been set to allow use of `ypset`. In most cases, *server* should be specified as an IP address.

`ypset` tries to bind over a connectionless transport. The NIS library call, `yp_all()`, uses connection-oriented transport and derives the NIS server's address based on the connectionless address supplied by `ypset`.

Refer to `ypfiles(4)` for an overview of the NIS name service.

Options

`-d` *ypdomain* Use *ypdomain*, instead of the default domain.

`-h` *host* Set `ypbind`'s binding on *host*, instead of locally. Specify *host* as a name.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/network/nis

See Also ypwhich(1), ypfiles(4), attributes(5)

Name ypstart, ypstop – Start and stop NIS services

Synopsis

```
/usr/lib/netsvc/yp/ypstart

/usr/lib/netsvc/yp/ypstop
```

Description The `ypstart` command is used to start the Network Information Service (NIS). After the host has been configured using the `ypinit(1M)` command, `ypstart` automatically determines the NIS status of the machine and starts the appropriate daemons.

The `ypstop` command is used to stop the Network Information Service (NIS).

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	service/network/nis

See Also `ypinit(1M)`, `attributes(5)`

Oracle Solaris Administration: Common Tasks

Notes The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two services remains the same; only the name has changed. The name Yellow Pages is a registered trademark in the United Kingdom of British Telecommunications PLC, and must not be used without permission.

Name ypxfr, ypxfr_1perday, ypxfr_1perhour, ypxfr_2perday – transfer NIS map from a NIS server to host

Synopsis

```
/usr/lib/netsvc/yp/ypxfr [-c] [-f] [-C tid prog server]
     [-d ypdomain] [-h host] [-s ypdomain] mapname
```

Description The `ypxfr` command moves an NIS map in the default domain for the local host to the local host by making use of normal NIS services. It creates a temporary map in the directory `/var/yp/`*ypdomain* (this directory must already exist; *ypdomain* is the default domain for the local host), fills it by enumerating the map's entries, fetches the map parameters (master and order number), and loads them. It then deletes any old versions of the map and moves the temporary map to the real *name*.

If run interactively, `ypxfr` writes its output to the terminal. However, if it is started without a controlling terminal, and if the log file `/var/yp/ypxfr.log` exists, it appends all its output to that file. Since `ypxfr` is most often run from the privileged user's `crontab` file, or by `ypserv`, the log file can retain a record of what was attempted, and what the results were.

For consistency between servers, `ypxfr` should be run periodically for every map in the NIS data base. Different maps change at different rates: a map might not change for months at a time, for instance, and can therefore be checked only once a day. Some maps might change several times per day. In such a case, you might want to check hourly for updates. A `crontab(1)` entry can be used to automatically perform periodic updates. Rather than having a separate `crontab` entry for each map, you can group commands to update several maps in a shell script. Examples (mnemonically named) are in `/usr/sbin/yp`: `ypxfr_1perday`, `ypxfr_2perday`, and `ypxfr_1perhour`.

Refer to `ypfiles(4)` for an overview of the NIS name service.

Options

`-c`	Do not send a “Clear current map” request to the local `ypserv` process. Use this flag if `ypserv` is not running locally at the time you are running `ypxfr`. Otherwise, `ypxfr` complains that it cannot communicate with the local `ypserv`, and the transfer fails.
`-f`	Force the transfer to occur even if the version at the master is not more recent than the local version.
`-C` *tid prog server*	This option is for use *only* by `ypserv`. When `ypserv` starts `ypxfr`, it specifies that `ypxfr` should call back a `yppush` process at the host *server*, registered as program number *prog*, and waiting for a response to transaction *tid*.
`-d` *ypdomain*	Specify a domain other than the default domain.
`-h` *host*	Get the map from *host*, regardless of the master. If *host* is not specified, `ypxfr` asks the NIS service for the name of the master, and tries to get the map from there. *host* must be a valid host name.

`-s` *ypdomain* Specify a source domain from which to transfer a map that should be the same across domains.

Files

`/var/yp/ypxfr.log`	Log file
`/usr/lib/netsvc/yp/ypxfr_1perday`	Script to run one transfer per day, for use with `cron`(1M)
`/usr/lib/netsvc/yp/ypxfr_2perday`	Script to run two transfer per day, for use with `cron`(1M)
`/usr/lib/netsvc/yp/ypxfr_1perhour`	Script for hourly transfers of volatile maps
`/var/yp/`*ypdomain*	NIS domain
`/usr/spool/cron/crontabs/root`	Privileged user's `crontab` file

Attributes See `attributes`(5) for descriptions of the following attributes:

ypxfr Only

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/network/nis

ypxfr_1perday, ypxfr_1perhour, and ypxfr_2perday

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	service/network/nis

See Also `crontab`(1), `cron`(1M), `ypinit`(1M), `yppush`(1M), `ypserv`(1M), `ypfiles`(4), `attributes`(5)

Name zdb – ZFS debugger

Synopsis `zdb` *pool*

Description The `zdb` command is used by support engineers to diagnose failures and gather statistics. Since the ZFS file system is always consistent on disk and is self-repairing, `zdb` should only be run under the direction by a support engineer.

If no arguments are specified, `zdb`, performs basic consistency checks on the pool and associated datasets, and report any problems detected.

Any options supported by this command are internal to Sun and subject to change at any time.

Exit Status The following exit values are returned:

`0` The pool is consistent.

`1` An error was detected.

`2` Invalid command line options were specified.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/file-system/zfs
Interface Stability	Uncommitted

See Also `zfs(1M)`, `zpool(1M)`, `attributes(5)`

Name zdump – time zone dumper

Synopsis

```
zdump [--version] [-v] [-c [loyear,]hiyear] [zonename]...
```

Description The `zdump` command prints the current time for each time zone (*zonename*) listed on the command line. Specify *zonename* as the name of the time zone database file relative to `/usr/share/lib/zoneinfo`.

Specifying an invalid time zone (*zonename*) to `zdump` does not return an error, rather `zdump` uses UTC. This is consistent with the behavior of the library calls; `zdump` reflects the same behavior of the time routines in `libc`. See `ctime(3C)` and `mktime(3C)`.

Options The following options are supported:

`--`*version* Outputs version information and exits.

`-v` Displays the entire contents of the time zone database file for *zonename*. Prints the time at the lowest possible time value; the time one day after the lowest possible time value; the times both one second before and exactly at each time at which the rules for computing local time change; the time at the highest possible time value; and the time at one day less than the highest possible time value. See `mktime(3C)` and `ctime(3C)` for information regarding time value (`time_t`). Each line of output ends with `isdst=1` if the given time is Daylight Saving Time, or `isdst=0` otherwise.

`-c` [*loyear*],*hiyear* Cuts off the verbose output near the start of the given year(s) . By default, the program cuts off verbose output near the start of the years -500 and 2500.

Exit Status The following exit values are returned:

`0` Successful completion.

`1` An error occurred.

Files `/usr/share/lib/zoneinfo` Standard zone information directory

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os
Interface Stability	Committed

See Also `zic(1M)`, `ctime(3C)`, `mktime(3C)`, `attributes(5)`, `environ(5)`

Name zfs – configures ZFS file systems

Synopsis

```
zfs [-?]

zfs help subcommand | help | property property-name | permission

zfs help -l properties

zfs allow filesystem|volume

zfs allow [-ldug] everyone|user|group[,...] perm|@setname[,...]
     filesystem|volume

zfs allow [-ld] -e perm|@setname[,...] filesystem|volume

zfs allow -c perm|@setname[,...] filesystem|volume

zfs allow -s @setname perm|@setname[,...] filesystem|volume

zfs clone [-p] [-K] [-o property=value] ... snapshot filesystem|volume

zfs create [-p] [-o property=value] ... filesystem

zfs create [-ps] [-b blocksize] [-o property=value] ... -V size volume

zfs destroy [-rRf] filesystem|volume

zfs destroy [-rRd] snapshot

zfs destroy share

zfs diff [-FHte] [-o field] ... snapshot [snapshot|filesystem]

zfs diff -E [-FHt] [-o field] ... snapshot|filesystem

zfs get [-rHpe|-d max][-o all | field[,...]] [-s source[,...]]
     all | property[,...] filesystem|volume|snapshot|share ...

zfs get share [filesystem]

zfs groupspace [-hniHp] [-o field[,...]] [-sS field] ...
     [-t type [,...]] filesystem|snapshot

zfs hold [-r] tag snapshot...

zfs holds [-r] snapshot...

zfs key -l {-a | [-r] filesystem|volume}

zfs key -u [-f] {-a | [-r] filesystem|volume}

zfs key -c [-o keysource=value] {-a | [-r] filesystem|volume}

zfs key -K {-a | [-r] filesystem|volume}

zfs list [-rH|-d max][-o property[,...]] [-t type[,...]]
     [-s property] ... [-S property] ...
     [filesystem|volume|snapshot|share|path] ...

zfs inherit [-rS] property filesystem|volume|snapshot|share ...
```

```
zfs mount

zfs mount [-vOS] [-o options] -a | filesystem

zfs promote clone-filesystem

zfs receive [-vnFu] [[-o property=value] | [-x property]] ...
     filesystem|volume|snapshot

zfs receive [-vnFu] [[-o property=value] | [-x property]] ...
     [-d | -e] filesystem

zfs release [-r] tag snapshot...

zfs rename filesystem|volume|snapshot
     filesystem|volume|snapshot

zfs rename [-p] filesystem|volume filesystem|volume

zfs rename -r snapshot snapshot

zfs rename share share

zfs rollback [-rRf] snapshot

zfs send [-DRbpv] [-[iI] snapshot] snapshot

zfs send -r [-Dbcpv] [-i snapshot] snapshot

zfs set [-r] property=value filesystem|volume|snapshot ...

zfs share -u [-o property=value] filesystem%share

zfs share filesystem|mountpoint|filesystem%share

zfs share -a| -r | filesystem

zfs snapshot [-r] [-o property=value]...
      filesystem@snapname|volume@snapname

zfs unmount [-f] -a | filesystem|mountpoint

zfs unshare filesystem|mountpoint|filesystem%share

zfs unshare -a| -r filesystem|

zfs upgrade

zfs upgrade [-v]

zfs upgrade [-r] [-V version] -a | filesystem

zfs userspace [-hniHp] [-o field[,...]] [-sS field] ...
     [-t type [,...]] filesystem|snapshot

zfs unallow [-rldug] everyone|user|group[,...] [perm|@setname[,... ]]
     filesystem|volume

zfs unallow [-rld] -e [perm|@setname[,... ]] filesystem|volume

zfs unallow [-r] -c [perm|@setname[ ... ]] filesystem|volume
```

```
zfs unallow [-r] -s @setname [perm|@setname[,... ]] filesystem|volume
```

Description The `zfs` command configures ZFS datasets within a ZFS storage pool, as described in `zpool(1M)`. A dataset is identified by a unique path within the ZFS namespace. For example:

`pool/{`*filesystem*`,`*volume*`,`*snapshot*`}`

where the maximum length of a dataset name is `MAXNAMELEN` (256 bytes).

A dataset can be one of the following:

file system
: A ZFS dataset of type `filesystem` can be mounted within the standard system namespace and behaves like other file systems. While ZFS file systems are designed to be POSIX compliant, known issues exist that prevent compliance in some cases. Applications that depend on standards conformance might fail due to nonstandard behavior when checking free file system space.

volume
: A logical volume exported as a raw or block device. This type of dataset should only be used under special circumstances. File systems are typically used in most environments.

snapshot
: A read-only version of a file system or volume at a given point in time. It is specified as *filesystem@name* or *volume@name*.

ZFS File System Hierarchy

A ZFS storage pool is a logical collection of devices that provide space for datasets. A storage pool is also the root of the ZFS file system hierarchy.

The root of the pool can be accessed as a file system, such as mounting and unmounting, taking snapshots, and setting properties. The physical storage characteristics, however, are managed by the `zpool(1M)` command.

See `zpool(1M)` for more information on creating and administering pools.

Snapshots

A snapshot is a read-only copy of a file system or volume. Snapshots can be created extremely quickly, and initially consume no additional space within the pool. As data within the active dataset changes, the snapshot consumes more data than would otherwise be shared with the active dataset.

Snapshots can have arbitrary names. Snapshots of volumes can be cloned or rolled back, but cannot be accessed independently.

File system snapshots can be accessed under the `.zfs/snapshot` directory in the root of the file system. Snapshots are automatically mounted on demand and may be unmounted at regular intervals. The visibility of the `.zfs` directory can be controlled by the `snapdir` property.

Clones

A clone is a writable volume or file system whose initial contents are the same as another dataset. As with snapshots, creating a clone is nearly instantaneous, and initially consumes no additional space.

Clones can only be created from a snapshot. When a snapshot is cloned, it creates an implicit dependency between the parent and child. Even though the clone is created somewhere else in the dataset hierarchy, the original snapshot cannot be destroyed as long as a clone exists. The `origin` property exposes this dependency, and the `destroy` command lists any such dependencies, if they exist.

The clone parent-child dependency relationship can be reversed by using the `promote` subcommand. This causes the "origin" file system to become a clone of the specified file system, which makes it possible to destroy the file system that the clone was created from.

Mount Points

Creating a ZFS file system is a simple operation, so the number of file systems per system is likely to be numerous. To cope with this, ZFS automatically manages mounting and unmounting file systems without the need to edit the `/etc/vfstab` file. All automatically managed file systems are mounted by ZFS at boot time.

By default, file systems are mounted under */path*, where *path* is the name of the file system in the ZFS namespace. Directories are created and destroyed as needed.

A file system can also have a mount point set in the `mountpoint` property. This directory is created as needed, and ZFS automatically mounts the file system when the `zfs mount -a` command is invoked (without editing `/etc/vfstab`). The `mountpoint` property can be inherited, so if `pool/home` has a mount point of `/export/stuff`, then `pool/home/user` automatically inherits a mount point of `/export/stuff/user`.

A file system can be mounted temporarily at a location other than the file systems's persistent mount point by specifying the `-o mountpoint=`*value* option to the `zfs mount` command. This is only permitted for file systems with non-legacy mount points.

A file system `mountpoint` property of `none` prevents the file system from being mounted.

If needed, ZFS file systems can also be managed with traditional tools (`mount`, `umount`, `/etc/vfstab`). If a file system's mount point is set to `legacy`, ZFS makes no attempt to manage the file system, and the administrator is responsible for mounting and unmounting the file system.

Zones

The physical properties of an added file system are controlled by the global administrator. However, the zone administrator can create, modify, or destroy files within the added file system, depending on how the file system is mounted.

A dataset can also be delegated to a non-global zone by using the `zonecfg add dataset` subcommand. You cannot delegate a dataset to one zone and the children of the same dataset to another zone. The zone administrator can change properties of the dataset or any of its children. However, the `quota` property is controlled by the global administrator.

A ZFS volume can be added as a device to a non-global zone by using the `zonecfg add device` subcommand. However, its physical properties can be modified only by the global administrator.

For more information about `zonecfg` syntax, see `zonecfg(1M)`.

After a dataset is delegated to a non-global zone, the `zoned` property is automatically set. A zoned file system can only be mounted in the global zone by using a temporary `mountpoint` property (see "Temporary Mount Point Properties").

The global administrator can forcibly clear the `zoned` property, though this should be done with extreme care. The global administrator should verify that all the mount points are acceptable before clearing the property.

Deduplication

Deduplication is the process of removing redundant data at the block-level, reducing the total amount of data stored. Deduplication is pool-wide; each dataset can opt in or out using its own `dedup` property. If a file system has the `dedup` property enabled, duplicate data blocks are removed synchronously on write. The result is that only unique data are stored and common components are shared among files in all datasets in the pool that have `dedup` enabled.

Encryption

For a full description of ZFS encryption and the ZFS encryption syntax, see `zfs_encrypt(1M)`.

Native Properties

Properties are divided into two types, native properties and user-defined (or *user*) properties. Native properties either provide internal statistics or control ZFS behavior. In addition, native properties are either editable or read-only. User properties have no effect on ZFS behavior, but you can use them to annotate datasets in a way that is meaningful in your environment. For more information about user properties, see the "User Properties" section, below.

Every dataset has a set of properties that provide statistics about the dataset as well as control various behaviors. Properties are inherited from the parent unless overridden by the child. Some properties apply only to certain types of datasets (file systems, volumes, or snapshots).

The values of numeric properties can be specified using human-readable suffixes (for example, `k`, `KB`, `M`, `Gb`, and so forth, up to `Z` for zettabyte). The following are all valid (and equal) specifications:

```
1536M, 1.5g, 1.50GB
```

The values of non-numeric properties are case-sensitive and must be lowercase, except for the `mountpoint` property.

The following native properties consist of read-only statistics about the dataset. These properties can be neither set, nor inherited. Native properties apply to all dataset types unless otherwise noted.

`available`
The amount of space available to the dataset and all its children, assuming that there is no other activity in the pool. Because space is shared within a pool, availability can be limited by any number of factors, including physical pool size, quotas, reservations, or other datasets within the pool.

This property can also be referred to by its shortened column name, `avail`.

`compressratio`
The compression ratio achieved for this dataset, expressed as a multiplier. Compression can be turned on by running: `zfs set compression=on` *dataset*. The default value is `off`.

`creation`
The time this dataset was created.

`defer_destroy`
This property is `on` if the snapshot has been marked for deferred destroy by using the `zfs destroy -d` command. Otherwise, the property is `off`.

`keychangedate`
For more information, see `zfs_encrypt(1M)`.

`keystatus`
For more information, see `zfs_encrypt(1M)`.

`mounted`
For file systems, indicates whether the file system is currently mounted. This property can be either `yes` or `no`.

`origin`
For cloned file systems or volumes, the snapshot from which the clone was created. The origin cannot be destroyed (even with the `-r` or `-f` options) so long as a clone exists.

`referenced`
The amount of data that is accessible by this dataset, which may or may not be shared with other datasets in the pool. When a snapshot or clone is created, it initially references the same amount of space as the file system or snapshot it was created from, since its contents are identical.

This property can also be referred to by its shortened column name, `refer`.

`rekeydate`
For more information, see `zfs_encrypt(1M)`.

`type`
The type of dataset: `filesystem`, `volume`, or `snapshot`.

`used`
The amount of space consumed by this dataset and all its descendents. This is the value that is checked against this dataset's quota and reservation. The space used does not include this dataset's reservation, but does take into account `refreservation` (through

usedbyrefreservation) and the reservations of any descendent datasets (through usedbychildren). The amount of space that a dataset consumes from its parent, as well as the amount of space that are freed if this dataset is recursively destroyed, is the greater of its space used and its reservation.

When snapshots (see the "Snapshots" section) are created, their space is initially shared between the snapshot and the file system, and possibly with previous snapshots. As the file system changes, space that was previously shared becomes unique to the snapshot, and counted in the snapshot's space used. Additionally, deleting snapshots can increase the amount of space unique to (and used by) other snapshots.

The amount of space used, available, or referenced does not take into account pending changes. Pending changes are generally accounted for within a few seconds. Committing a change to a disk using fsync(3C) or O_SYNC does not necessarily guarantee that the space usage information is updated immediately.

usedby*
The usedby* properties decompose the used properties into the various reasons that space is used. Specifically, used = usedbychildren + usedbydataset + usedbyrefreservation +, usedbysnapshots. These properties are only available for datasets created on pools that are version 13 or higher.

usedbychildren
The amount of space used by children of this dataset, which would be freed if all the dataset's children were destroyed.

usedbydataset
The amount of space used by this dataset itself, which would be freed if the dataset was destroyed (after first removing any refreservation and destroying any necessary snapshots or descendents).

usedbyrefreservation
The amount of space used by a refreservation set on this dataset, which would be freed if the refreservation was removed.

Space accounted for by this property represents potential consumption by future writes, reserved in advance to prevent write allocation failures in this dataset. This can include unwritten data, space currently shared with snapshots, and compression savings for volumes (which may be lost when replaced with less compressible data). When allocations for later writes increase usedbydataset or usedbysnapshots, usedbyrefreservation will decrease accordingly.

usedbysnapshots
The amount of space consumed by snapshots of this dataset. In particular, it is the amount of space that would be freed if all of this dataset's snapshots were destroyed. Note that this is not simply the sum of the snapshots' used properties because space can be shared by multiple snapshots.

`userused@`*user*
The amount of space consumed by the specified user in this dataset. Space is charged to the owner of each file, as displayed by `ls -l`. The amount of space charged is displayed by `du` and `ls -s`. See the `zfs userspace` subcommand for more information.

Unprivileged users can access only their own space usage. The root user, or a user who has been granted the `userused` privilege with `zfs allow`, can access everyone's usage.

The `userused@...` properties are not displayed by `zfs get all`. The user's name must be appended after the `@` symbol, using one of the following forms:

- *POSIX name* (for example, `joe`)
- *POSIX numeric ID* (for example, `789`)
- *SID name* (for example, `joe.smith@mydomain`)
- *SID numeric ID* (for example, `S-1-123-456-789`)

`userrefs`
This property is set to the number of user holds on this snapshot. User holds are set by using the `zfs hold` command.

`groupused@`*group*
The amount of space consumed by the specified group in this dataset. Space is charged to the group of each file, as displayed by `ls -l`. See the `userused@`*user* property for more information.

Unprivileged users can only access their own groups' space usage. The root user, or a user who has been granted the `groupused` privilege with `zfs allow`, can access all groups' usage.

`volblocksize=`*blocksize*
For volumes, specifies the block size of the volume. The `blocksize` cannot be changed once the volume has been written, so it should be set at volume creation time. The default `blocksize` for volumes is 8 KB. Any power of 2 from 512 bytes to 1 MB is valid.

This property can also be referred to by its shortened column name, `volblock`.

The following native properties can be used to change the behavior of a ZFS dataset.

`aclmode=discard | mask | passthrough`
Controls how an ACL is modified during chmod(2). A file system with an `aclmode` property of `discard` (the default) deletes all ACL entries that do not represent the mode of the file. An `aclmode` property of `mask` reduces user or group permissions. The permissions are reduced so that they are no greater than the group permission bits, unless it is a user entry that has the same UID as the owner of the file or directory. In this case, the ACL permissions are reduced so that they are no greater than owner permission bits. `mask` also preserves the ACL across mode changes (without an explict ACL set [by means of chmod(1)] between the mode changes). A file system with an `aclmode` property of `passthrough` indicates that no changes will be made to the ACL other than generating the necessary ACL entries to represent the new mode of the file or directory.

`aclinherit=discard | noallow | restricted | passthrough | passthrough-x`
Controls how ACL entries are inherited when files and directories are created. A file system with an `aclinherit` property of `discard` does not inherit any ACL entries. A file system with an `aclinherit` property value of `noallow` only inherits inheritable ACL entries that specify "deny" permissions. The property value `restricted` (the default) removes the `write_acl` and `write_owner` permissions when the ACL entry is inherited. A file system with an `aclinherit` property value of `passthrough` inherits all inheritable ACL entries without any modifications made to the ACL entries when they are inherited. A file system with an `aclinherit` property value of `passthrough-x` has the same meaning as `passthrough`, except that all ACEs inherit the execute permission only if the file creation mode also requests the execute bit.

When the property value is set to `passthrough`, files are created with a mode determined by the inheritable ACEs. If no inheritable ACEs exist that affect the mode, then the mode is set in accordance to the requested mode from the application.

`atime=on | off`
Controls whether the access time for files is updated when they are read. Turning this property off avoids producing write traffic when reading files and can result in significant performance gains, though it might confuse mailers and other similar utilities. The default value is `on`.

`canmount=on | off | noauto`
If this property is set to `off`, the file system cannot be mounted, and is ignored by `zfs mount -a`. Setting this property to `off` is similar to setting the `mountpoint` property to `none`, except that the dataset still has a normal `mountpoint` property, which can be inherited. Setting this property to `off` allows datasets to be used solely as a mechanism to inherit properties. One example of setting `canmount=off` is to have two datasets with the same `mountpoint`, so that the children of both datasets appear in the same directory, but might have different inherited characteristics.

When the `noauto` option is set, a dataset can only be mounted and unmounted explicitly. The dataset is not mounted automatically when the dataset is created or imported, nor is it mounted by the `zfs mount -a` command or unmounted by the `zfs unmount -a` command.

This property is not inherited.

`checksum=on | off | fletcher2, | fletcher4 | sha256 | sha256+mac`
Controls the checksum used to verify data integrity. The default value is `on`, which automatically selects an appropriate algorithm (currently, `fletcher4`, but this may change in future releases). The value `off` disables integrity checking on user data. Disabling checksums is *NOT* a recommended practice.

Changing this property affects only newly-written data.

The value of `sha256+mac` is only available when encryption is enabled. The checksum property becomes `readonly` when encryption is enabled, and then is always set to `sha256+mac`.

`compression=on | off | lzjb | gzip | gzip-`*N* `| zle`
Controls the compression algorithm used for this dataset. The `lzjb` compression algorithm is optimized for performance while providing decent data compression. Setting compression to `on` uses the `lzjb` compression algorithm. The `gzip` compression algorithm uses the same compression as the `gzip`(1) command. You can specify the `gzip` level by using the value `gzip-`*N* where *N* is an integer from 1 (fastest) to 9 (best compression ratio). Currently, `gzip` is equivalent to `gzip-6` (which is also the default for `gzip`(1)).

This property can also be referred to by its shortened column name `compress`. Changing this property affects only newly-written data.

`copies=1 | 2 | 3`
Controls the number of copies of data stored for this dataset. These copies are in addition to any redundancy provided by the pool, for example, mirroring or RAID-Z. The copies are stored on different disks, if possible. The space used by multiple copies is charged to the associated file and dataset, changing the `used` property and counting against quotas and reservations.

Changing this property only affects newly-written data. Therefore, set this property at file system creation time by using the `-o copies=`*N* option.

When encryption is enabled on a dataset, copies can be set to a maximum of 2.

`dedup=on | off | verify | sha256[,verify]`
Controls whether deduplication is in effect for a dataset. The default value is `off`. The default checksum used for deduplication is `sha256` (subject to change). When `dedup` is enabled, the `dedup` checksum algorithm overrides the `checksum` property. Setting the value to `verify` is equivalent to specifying `sha256,verify`.

If the property is set to `verify`, then, whenever two blocks have the same signature, ZFS will do a byte-for-byte comparison with the existing block to ensure that the contents are identical.

`devices=on | off`
Controls whether device nodes can be opened on this file system. The default value is `on`.

`exec=on | off`
Controls whether processes can be executed from within this file system. The default value is `on`.

`logbias = latency | throughput`
Controls how ZFS optimizes synchronous requests for this dataset. If `logbias` is set to `latency`, ZFS uses the pool's separate log devices, if any, to handle the requests at low latency. If `logbias` is set to `throughput`, ZFS does not use the pool's separate log devices.

Instead, ZFS optimizes synchronous operations for global pool throughput and efficient use of resources. The default value is `latency`.

`mlslabel=`*label* | `none`
See the `multilevel` property for a description of the behavior of the `mlslabel` property on multilevel file systems. The following `mlslabel` description applies to non-multilevel file systems

The `mlslabel` property is a sensitivity label that determines if a dataset can be mounted in a zone on a system with Trusted Extensions enabled. If the labeled dataset matches the labeled zone, the dataset can be mounted and accessed from the labeled zone.

When the `mlslabel` property is not set, the default value is `none`. Setting the `mlslabel` property to `none` is equivalent to removing the property.

The `mlslabel` property can be modified only when Trusted Extensions is enabled and only with appropriate privilege. Rights to modify it cannot be delegated. When changing a label to a higher label or setting the initial dataset label, the `{PRIV_FILE_UPGRADE_SL}` privilege is required. When changing a label to a lower label or the default (`none`), the `{PRIV_FILE_DOWNGRADE_SL}` privilege is required. Changing the dataset to labels other than the default can be done only when the dataset is not mounted. When a dataset with the default label is mounted into a labeled-zone, the mount operation automatically sets the `mlslabel` property to the label of that zone.

When Trusted Extensions is *not* enabled, only datasets with the default label (`none`) can be mounted.

`mountpoint=`*path* | `none` | `legacy`
Controls the mount point used for this file system. See the "Mount Points" section for more information on how this property is used.

When the `mountpoint` property is changed for a file system, the file system and any children that inherit the mount point are unmounted. If the new value is `legacy`, then they remain unmounted. Otherwise, they are automatically remounted in the new location if the property was previously `legacy` or `none`, or if they were mounted before the property was changed. In addition, any shared file systems are unshared and shared in the new location.

`nbmand=on` | `off`
For more information, see `zfs_share(1M)`.

`primarycache=all` | `none` | `metadata`
Controls what is cached in the primary cache (ARC). If this property is set to `all`, then both user data and metadata is cached. If this property is set to `none`, then neither user data nor metadata is cached. If this property is set to `metadata`, then only metadata is cached. The default value is `all`.

`quota=`*size* | `none`
Limits the amount of space a dataset and its descendents can consume. This property enforces a hard limit on the amount of space used. This includes all space consumed by descendents, including file systems and snapshots. Setting a quota on a descendent of a dataset that already has a quota does not override the ancestor's quota, but rather imposes an additional limit.

Quotas cannot be set on volumes, as the `volsize` property acts as an implicit quota.

`sync=standard` | `always` | `disabled`
Determines the degree to which file system transactions are synchronized. This property can be set when a dataset is created, or dynamically, and will take effect immediately. This property can have one of the following settings:

`standard`
The default option. Synchronous file system transactions are written to the intent log and then all devices written are flushed to ensure the data is stable (that is, not cached by device controllers).

`always`
Each file system transaction is written and flushed to stable storage. This value has a significant performance penalty but might be appropriate for troubleshooting synchronous file system transactions.

`disabled`
Synchronous requests are disabled. File system transactions commit to stable storage only on the next DMU transaction group commit, which might be after many seconds. This setting gives the highest performance. However, it is very dangerous as ZFS would be ignoring the synchronous transaction demands of applications such as databases or NFS. Furthermore, when this setting is in effect for the currently active root or `/var` filesystem, out-of-spec behavior, application data loss, and increased vulnerability to replay attacks can result. Administrators should only use this option only when these risks are understood.

`userquota@`*user*`=`*size* | `none`
Limits the amount of space consumed by the specified user. Similar to the `refquota` property, the `userquota` space calculation does not include space that is used by descendent datasets, such as snapshots and clones. User space consumption is identified by the `userspace@`*user* property.

Enforcement of user quotas may be delayed by several seconds. This delay means that a user might exceed her quota before the system notices that she is over quota. The system would then begin to refuse additional writes with the `EDQUOT` error message . See the `zfs userspace` subcommand for more information.

Unprivileged users can only access their own groups' space usage. The root user, or a user who has been granted the `userquota` privilege with `zfs allow`, can get and set everyone's quota.

This property is not available on volumes, on file systems before version 4, or on pools before version 15. The `userquota@...` properties are not displayed by `zfs get all`. The user's name must be appended after the `@` symbol, using one of the following forms:

- *POSIX name* (for example, `joe`)
- *POSIX numeric ID* (for example, `789`)
- *SID name* (for example, `joe.smith@mydomain`)
- *SID numeric ID* (for example, `S-1-123-456-789`)

`groupquota@`*group*`=`*size* | `none`
Limits the amount of space consumed by the specified group. Group space consumption is identified by the `userquota@`*user* property.

Unprivileged users can access only their own groups' space usage. The root user, or a user who has been granted the `groupquota` privilege with `zfs allow`, can get and set all groups' quotas.

`readonly=on` | `off`
Controls whether this dataset can be modified. The default value is `off`.

This property can also be referred to by its shortened column name, `rdonly`.

`recordsize=`*size*
Specifies a suggested block size for files in the file system. This property is designed solely for use with database workloads that access files in fixed-size records. ZFS automatically tunes block sizes according to internal algorithms optimized for typical access patterns.

For databases that create very large files but access them in small random chunks, these algorithms may be suboptimal. Specifying a `recordsize` greater than or equal to the record size of the database can result in significant performance gains. Use of this property for general purpose file systems is strongly discouraged, and may adversely affect performance.

The default `recordsize` is 128 KB. The size specified must be a power of two greater than or equal to 512 and less than or equal to 1 MB.

Changing the file system's `recordsize` affects only files created afterward; existing files and received data are unaffected.

This property can also be referred to by its shortened column name, `recsize`.

`refquota=`*size* | `none`
Limits the amount of space a dataset can consume. This property enforces a hard limit on the amount of space used. This hard limit does not include space used by descendents, including file systems and snapshots.

`refreservation=`*size* | `none`
The minimum amount of space guaranteed to a dataset, not including its descendents. When the `usedbydataset` space is below this value, the dataset is treated as if it were taking up the amount of space specified by `refreservation`. The `usedbyrefreservation` figure

represents this extra space, adding to the total `used` space charged to the dataset, and in turn consuming from the parent datasets' usage, quotas, and reservations. This protects the dataset from overcommitment of pool resources, by ensuring that space for future writes is reserved in advance.

Space shared with snapshots can later be replaced with new data, and the snapshot represents a committment to keep both copies. If `refreservation` is set, `usedbyrefreservation` must be increased to the full size of `refreservation` when taking a new snapshot, accounting for this commitment. If there is insufficient space available to the dataset for this increase, snapshot creation will be denied.

This property can also be referred to by its shortened column name, `refreserv`.

`reservation=`*size* | `none`
The minimum amount of space guaranteed to a dataset and its descendents. When the amount of space used is below this value, the dataset is treated as if it were taking up the amount of space specified by its reservation. Reservations are accounted for in the parent datasets' space used, and count against the parent datasets' quotas and reservations.

This property can also be referred to by its shortened column name, `reserv`.

`rstchown=on` | `off`
Indicates whether the file system restricts users from giving away their files by means of chown(1) or the chown(2) system call. The default is to restrict `chown`. When `rstchown` is `off` then `chown` will act as if the user has the `PRIV_FILE_CHOWN_SELF` privilege.

`secondarycache=all` | `none` | `metadata`
Controls what is cached in the secondary cache (L2ARC). If this property is set to `all`, then both user data and metadata is cached. If this property is set to `none`, then neither user data nor metadata is cached. If this property is set to `metadata`, then only metadata is cached. The default value is `all`.

`setuid=on` | `off`
Controls whether the set-UID bit is respected for the file system. The default value is `on`.

`shadow=`*URI* | `none`
Identifies a ZFS file system as a *shadow* of the file system described by the *URI*. Data is migrated to a shadow file system with this property set from the file system identified by the *URI*. The file system to be migrated must be read-only for a complete migration.

Access to a directory that is not yet migrated in the shadow file system is blocked until the entire directory is migrated. Access to a file that is not yet migrated in the shadow file system causes only a portion of the file being accessed to be migrated. Multiple processes can migrate different portions of a file at the same time.

Two forms of URI are accepted, one for migrating a local file system to another file system on the same physical system, and one for remotely migrating a file system from an NFS server. The forms are:

`file:///`*path*
`nfs://`*host*/*path*

If `shadowd`(1M) is still running when the migration is complete, the file system is automatically remounted with the shadow property set to `none`. Or, when the migration is complete, you can manually set the shadow property to `none`.

`sharenfs=on | off`
For more information, see `zfs_share`(1M).

`sharesmb=on | off`
For more information, see `zfs_share`(1M).

`snapdir=hidden | visible`
Controls whether the `.zfs` directory is hidden or visible in the root of the file system as discussed in the "Snapshots" section. The default value is `hidden`.

`version=1 | 2 | current`
The on-disk version of this file system, which is independent of the pool version. This property can only be set to later supported versions. See the `zfs upgrade` command.

`volsize=`*size*
Specifies the logical size of the volume. By default, creating a volume establishes a `refreservation` that is a somewhat larger than the actual logical volume size, to account for ZFS metadata overhead. Any changes to `volsize` are reflected in an equivalent change to the `refreservation`. The `volsize` can only be set to a multiple of `volblocksize`, and cannot be zero.

The `refreservation` is set on the volume to prevent unexpected behavior for consumers. Without the reservation, the volume could run out of space, resulting in undefined behavior or data corruption, depending on how the volume is used. These effects can also occur when the volume size is changed while it is in use (particularly when shrinking the size). Extreme care should be used when adjusting the volume size.

Though not recommended, a *sparse volume* (also known as *thin provisioning*) can be created by specifying the `-s` option to the `zfs create -V` command. A sparse volume is a volume where the reservation is less then the volume size. Consequently, writes to a sparse volume can fail with ENOSPC when the pool is low on space. For a sparse volume, changes to `volsize` are not reflected in the reservation.

`vscan=on | off`
For more information, see `zfs_share`(1M).

`xattr=on | off`
Controls whether extended attributes are enabled for this file system. The default value is `on`.

`zoned=on | off`
Controls whether the dataset is managed from a non-global zone. See the "Zones" section for more information. The default value is `off`.

The following properties cannot be changed after the file system is created and, therefore, should be set when the file system is created. If the properties are not set with the `zfs create` or `zpool create` commands, these properties are inherited from the parent dataset. If the parent dataset lacks these properties due to having been created prior to these features being supported, the new file system will have the default values for these properties.

`casesensitivity=sensitive | insensitive | mixed`
For more information, see `zfs_share(1M)`.

`normalization = none | formC | formD | formKC | formKD`
For more information, see `zfs_share(1M)`.

`utf8only=on | off`
For more information, see `zfs_share(1M)`.

`encryption=off | on | aes-128-ccm | aes-192-ccm | aes-256-ccm | aes-128-gcm | aes-192-gcm | aes-256-gcm`
For more information, see `zfs_encrypt(1M)`.

`multilevel=on | off`
This property can only be used on a system with Trusted Extensions enabled. The default value is `off`.

Objects in a multilevel file system are individually labeled with an explicit sensitivity label attribute that is automatically generated. Objects can be relabeled in place by changing this label attribute, by using the `setlabel(1)` or `setflabel(3TSOL)` interfaces.

Zone datasets, system root datasets, and other datasets containing packaged Solaris code should not be multilevel.

There are differences in the `mlslabel` property on multilevel file systems. The `mlslabel` value defines the highest possible label that objects in the file system can have. Attempts to create a file at (or relabel a file to) a label higher than the `mlslabel` are disallowed. Mount policy based on `mlslabel` does not apply to multilevel file systems.

For multilevel file systems, the `mlslabel` property can be set explicitly during file system creation, otherwise a default `mlslabel` property of `ADMIN_HIGH` will be automatically created. After creating a multilevel file system, the `mlslabel` can be changed, but it cannot be changed to a lower label, nor be removed, nor set to `none`.

The following property must be specified at creation time and can modified by using special commands:

`keysource=`*raw* | *hex* | *passphrase*,*prompt* | *file*
For more information, see `zfs_encrypt(1M)`.

Temporary Mount Point Properties

When a file system is mounted, either through the legacy mount(1M) command or the `zfs mount` command, its mount options are set according to its properties. The correlation between properties and mount options is as follows:

PROPERTY	MOUNT OPTION
devices	devices/nodevices
mountpoint	mountpoint
exec	exec/noexec
readonly	ro/rw
setuid	setuid/nosetuid
xattr	xattr/noxattr
rstchown	rstchown/norstchown

In addition, these options can be set on a per-mount basis using the `-o` option, without affecting the property that is stored on disk. The values specified on the command line override the values stored in the dataset. The `-nosuid` option is an alias for `nodevices,nosetuid`. These properties are reported as *temporary* by the `zfs get` command. For properties other than `mountpoint`, if the properties are changed while the dataset is mounted, the new setting overrides any temporary settings. The `mountpoint` property cannot be changed while a temporary `mountpoint` property is in effect (that is, while the dataset is mounted at a temporary location).

User Properties

In addition to the standard native properties, ZFS supports arbitrary user properties. User properties have no effect on ZFS behavior, but applications or administrators can use them to annotate datasets (file systems, volumes, and snapshots).

User property names must contain a colon (`:`) character to distinguish them from native properties. They may contain lowercase letters, numbers, and the following punctuation characters: colon (`:`), dash (`-`), period (`.`), and underscore (`_`). The expected convention is that the property name is divided into two portions such as *module*:*property*, but this namespace is not enforced by ZFS. User property names can be at most 256 characters, and cannot begin with a dash (`-`).

When making programmatic use of user properties, it is strongly suggested to use a reversed DNS domain name for the *module* component of property names to reduce the chance that two independently-developed packages use the same property name for different purposes. In the Oracle Solaris release, the `com.oracle` user property is reserved for `beadm` command and library

The values of user properties are arbitrary strings, are always inherited, and are never validated. All of the commands that operate on properties (`zfs list`, `zfs get`, , and so forth) can be used to manipulate both native properties and user properties. Use the `zfs inherit` command to clear a user property . If the property is not defined in any parent dataset, it is removed entirely. Property values are limited to 1024 characters.

ZFS Volumes as Swap or Dump Devices

During an initial installation, a swap device and dump device are created on ZFS volumes in the ZFS root pool. Separate ZFS volumes must be used for the swap area and dump devices. Do not swap to a file on a ZFS file system. A ZFS swap file configuration is not supported.

You can encrypt a ZFS volume used as a swap device by specifying the `encryption` property for that device and specifying the `encrypted` option in `vfstab(4)`. For more information about the encryption property, see zfs_encrypt.1m.

If you need to change your swap area or dump device after the system is installed or upgraded, use the `swap(1M)` and `dumpadm(1M)` commands. If you need to change the size of your swap area or dump device, see the *Oracle Solaris 11.1 Administration: ZFS File Systems*

Subcommands

All subcommands that modify state are logged persistently to the pool in their original form.

`zfs ?`
Displays a help message.

`zfs help` *command* | help | *property property-name* | *permission*
Displays `zfs` command usage information. You can display help for a specific command, property, or delegated permission. If you display help for a specific command or property, the command syntax or property value is displayed. Using `zfs help` without any arguments displays a complete list of `zfs` commands.

`zfs help -l properties`
Displays `zfs` property information, including whether the property value is editable and inheritable, and their possible values.

`zfs allow` *filesystem* | *volume*
`zfs allow` [`-ldug`] `everyone`|*user*|*group*[,...] *perm*|*@setname*[,...] *filesystem*| *volume*
`zfs allow` [`-ld`] `-e` *perm*|*@setname*[,...] *filesystem* | *volume*
`zfs allow -c` *perm*|*@setname*[,...] *filesystem*|*volume*
`zfs allow -s` *@setname perm*|*@setname*[,...] *filesystem*|*volume*
For a full description of the `zfs allow` syntax and examples, see `zfs_allow(1M)`.

`zfs clone` [`-p`] [`-K`] [`-o` *property=value*] ... *snapshot filesystem*|*volume*
Creates a clone of the given snapshot. See the "Clones" section for details. The target dataset can be located anywhere in the ZFS hierarchy, and is created as the same type as the original.

`-p`
Creates all the non-existing parent datasets. Datasets created in this manner are automatically mounted according to the `mountpoint` property inherited from their parent. If the target file system or volume already exists, the operation completes successfully.

`-o` *property=value*
Sets the specified property; see `zfs create` for details.

`-K`
For information, see `zfs_encrypt(1M)`.

`zfs create` [`-p`] [`-o` *property=value*] ... *filesystem*
Creates a new ZFS file system. The file system is automatically mounted according to the `mountpoint` property inherited from the parent.

`-p`
Creates all the non-existing parent datasets. Datasets created in this manner are automatically mounted according to the `mountpoint` property inherited from their parent. Any property specified on the command line using the `-o` option is ignored. If the target filesystem already exists, the operation completes successfully.

`-o` *property=value*
Sets the specified property as if the command *property=value* was invoked at the same time the dataset was created. Any editable ZFS property can also be set at creation time. Multiple `-o` options can be specified. An error results if the same property is specified in multiple `-o` options.

`zfs create` [`-ps`] [`-b` *blocksize*] [`-o` *property=value*] ... `-V` *size volume*
Creates a volume of the given size. The volume is exported as a block device in `/dev/zvol/{dsk,rdsk}/`*path*, where *path* is the name of the volume in the ZFS namespace. The size represents the logical size as exported by the device. By default, a reservation of equal size is created.

size is automatically rounded up to the nearest 128 KB to ensure that the volume has an integral number of blocks regardless of *blocksize*.

`-p`
Creates all the non-existing parent datasets. Datasets created in this manner are automatically mounted according to the `mountpoint` property inherited from their parent. Any property specified on the command line using the `-o` option is ignored. If the target filesystem already exists, the operation completes successfully.

`-s`
Creates a sparse volume with no reservation. See `volsize` in the Native Properties section for more information about sparse volumes.

`-o` *property=value*
Sets the specified property as if the *property=value* command was invoked at the same time the dataset was created. Any editable ZFS property can also be set at creation time. Multiple `-o` options can be specified. An error results if the same property is specified in multiple `-o` options.

`-b` *blocksize*
Equivalent to `-o volblocksize=`*blocksize*. If this option is specified in conjunction with `-o volblocksize`, the resulting behavior is undefined.

`zfs destroy [-rRf] filesystem|volume`
Destroys the given dataset. By default, the command unshares any file systems that are currently shared, unmounts any file systems that are currently mounted, and refuses to destroy a dataset that has active dependents (children or clones).

`-r`
Recursively destroy all children.

`-R`
Recursively destroy all dependents, including cloned file systems outside the target hierarchy.

`-f`
Force an unmount of any file systems using the `unmount -f` command. This option has no effect on non-file systems or unmounted file systems.

Extreme care should be taken when applying either the `-r` or the `-f` options, as they can destroy large portions of a pool and cause unexpected behavior for mounted file systems in use.

`zfs destroy [-rRd] snapshot`
The given snapshot is destroyed immediately if and only if the `zfs destroy` command without the `-d` option would have destroyed it. Such immediate destruction would occur, for example, if the snapshot had no clones and the user-initiated reference count were zero.

If the snapshot does not qualify for immediate destruction, it is marked for deferred deletion. In this state, it exists as a usable, visible snapshot until both of the preconditions listed above are met, at which point it is destroyed.

`-d`
Defer snapshot deletion.

`-r`
Destroy (or mark for deferred deletion) all snapshots with this name in descendent file systems.

`-R`
Recursively destroy all dependents.

`zfs destroy [share`
The specified file system share is destroyed.

`zfs diff [-FHte] [-o field] ... snapshot [snapshot | filesystem]`
`zfs diff -E [-FHt] [-o field] ... snapshot | filesystem`
Gives a high-level description of the differences between a snapshot and a descendent dataset. The descendent can be either a snapshot of the dataset or the current dataset.

If a single snapshot is specified, then differences between that snapshot and the current dataset are given.

For each file that has undergone a change between the original snapshot and the descendent, the type of change is described along with the name of the file. In the case of a rename, both the old and new names are shown. Whitespace characters, backslash characters, and other non-printable or non-7-bit ASCII characters found in file names are displayed as a backslash character followed by the three-digit octal representation of the byte value.

If the `-t` option is specified, the first column of output from the command is the file's `st_ctim` value. For deleted files, this is the final `st_ctim` in the earlier snapshot.

The type of change follows any timestamp displayed, and is described with a single character:

`+` Indicates the file was added in the later dataset.

`-` Indicates the file was removed in the later dataset.

`M` Indicates the file was modified in the later dataset.

`R` Indicates the file was renamed in the later dataset.

If the `-F` option is specified, the next column of output is a single character describing the type of the file. The mappings are:

`F` Regular file

`/` Directory

`B` Block device

`>` Door

`|` FIFO

`@` Symbolic link

`P` Event portal

`=` Socket

If the modification involved a change in the link count of a non-directory file, the change is expressed as a delta within parentheses on the modification line. If the file was renamed, the old name is separated from the new with the string `->`.

If the `-H` option is selected, easier-to-parse output is produced. Fields are separated by a single tab, and no arrow string (`->`) is placed between the old and new names of a rename. No guarantees are made on the spacing between fields of non `-H` output.

If the `-e` option is selected, then all files added or modified between the two snapshots are enumerated and no deleted files are displayed. The change type always reports as `+` regardless of the type of modification.

If the `-E` option is selected, then differences are given as if from an empty snapshot to the specified snapshot or dataset.

If the `-o` field option is selected then only selected fields are displayed. Each line starts with the standard fields requested by the `-F` and `-t` options, followed by the fields requested in successive `-o` options. As with the `-H` option, all fields are separated by a single tab. The allowable field names include:

`object`	The number printed by `ls -i` for the file
`parent`	The number printed by `ls -i` for enclosing directory of the file
`size`	The file size as displayed by `ls -s`
`links`	The number of links to the file
`linkschange`	The change in the number of links to the file
`name`	The name of the file
`oldname`	The name of the file before the rename, or – (hyphen) if the file was not renamed
`user`	The owner name of the file as displayed by `ls`
`group`	The group name of the file as displayed by `ls`
`ctime`	Timestamp when the file's metadata was last modified
`mtime`	Timestamp when the file was last modified
`atime`	Timestamp when a file was last accessed
`crtime`	Timestamp when a file was created

You must be granted the `diff` permission with `zfs allow` to use this subcommand, unless you already have the `{PRIV_SYS_CONFIG}` or `{PRIV_SYS_MOUNT}` privilege.

`zfs get` [`-rHpe`|`-d` *max*] [`-o all` | *field*[,...] [`-s` *source*[,...]] `all` | *property*[,...] *filesystem*|*volume*|*snapshot*|*share* ...

Displays properties for the given datasets. If no datasets are specified, then the command displays properties for all datasets on the system. For each property, the following columns are displayed:

```
name      Dataset name
property  Property name
value     Property value
source    Property source. Can either be local, default,
          temporary, inherited, or none (-).
```

All columns except the `RECEIVED` column are displayed by default; specify particular or all columns, using the `-o` option. This command takes a comma-separated list of properties as described in the “Native Properties” and “User Properties” sections.

The special value `all` can be used to display all properties that apply to the given dataset's type (filesystem, volume, or snapshot).

`-r`
Recursively display properties for any children.

`-H`
Display output in a form more easily parsed by scripts. Any headers are omitted, and fields are explicitly separated by a single tab instead of an arbitrary amount of space.

`-p`
Displays numbers in parseable (exact) values.

`-e`
Expands property sublists to any depth.

`-d` *max*
Recursively displays any children of the dataset, limiting the recursion to *depth*. A depth of `1` will display only the dataset and its direct children.

`-o` *field*
Set of fields to display. One or more of:

```
name,property,value,received,source
```

Present multiple fields as a comma-separated list. The default value is:

```
name,property,value,source
```

The keyword `all` specifies all sources.

`-s` *source*
A comma-separated list of sources to display. Those properties coming from a source other than those in this list are ignored. Each source must be one of the following:

```
local,default,inherited,temporary,received,none
```

The default value is all sources.

`zfs groupspace` [`-hniHp`] [`-o` *field*[,...]] [`-sS` *field*]... [`-t` *type* [,...]] *filesystem* | *snapshot*
Displays space consumed by, and quotas on, each group in the specified filesystem or snapshot. This subcommand is identical to `zfs userspace`, except that the default types to display are `-t posixgroup,smbgroup`.

-

`zfs hold` [`-r`] *tag snapshot* ...
Adds a single reference, named with the *tag* argument, to the specified snapshot or snapshots. Each snapshot has its own tag namespace, and tags must be unique within that space.

If a hold exists on a snapshot, attempts to destroy that snapshot by using the `zfs destroy` command return `EBUSY`.

`-r`
Specifies that a hold with the given tag is applied recursively to the snapshots of all descendent file systems.

`zfs holds` [`-r`] *snapshot* ...
Lists all existing user references for the given snapshot or snapshots.

`-r`
Lists the holds that are set on the named descendent snapshots, in addition to listing the holds on the named snapshot.

`zfs inherit` [`-rS`] *property filesystem*|*volume*|*snapshot*|*share* ...
Clears the specified property, causing it to be inherited from an ancestor. If no ancestor has the property set, then the default value is used. See the "Properties" section for a listing of default values, and details on which properties can be inherited.

`-r`
Recursively inherits the given property for all children.

`-S`
Reverts to the received property value, if any. If the property does not have a received value, the behavior of `zfs inherit -S` is the same as `zfs inherit` without `-S`. If the property does have a received value, `zfs inherit` masks the received value with the inherited value until `zfs inherit -S` reverts to the received value.

`zfs key -l` | {`-a` | [`-r`] *filesystem*|*volume*}
`zfs key -u` [`-f`] | {`-a` | [`-r`] *filesystem*|*volume*
`zfs key -c` [`-o` *keysource=value*] {`-a` | [`-r`] *filesystem*|*volume*}
`zfs key -K` {`-a` | [`-r`] *filesystem*|*volume*}
For a full description of the `zfs key` syntax and examples, see `zfs_encrypt(1M)`.

`zfs list` [`-rH`|`-d` *max*] [`-o` *property*[,...]] [`-t` *type*[,...]] [`-s` *property*] ... [`-S` *property*] ... [*filesystem*|*volume*|*snapshot*|*share*|*path*] ...
Lists the property information for the given datasets in tabular form. If specified, you can list property information by the absolute pathname or the relative pathname. By default, all file systems and volumes are displayed. Snapshots are displayed if the `listsnaps` property is `on`. The default is `off`. The following fields are displayed: `name`, `used`, `available`, `referenced`, `mountpoint`.

`-H`
Used for scripting mode. Do not print headers and separate fields by a single tab instead of arbitrary white space.

`-r`
Recursively displays any children of the dataset on the command line.

`-d` *depth*
Recursively displays any children of the dataset, limiting the recursion to maximum *depth*. A depth of `1` will display only the dataset and its direct children.

`-o` *property*
A comma-separated list of properties to display. The property must be:

- One of the properties described in the "Native Properties" section
- A user property
- The value `name` to display the dataset name
- The value `space` to display space usage properties on file systems and volumes. This is a shortcut for specifying
  `-o name,avail,used,usedsnap,usedds,usedrefreserv,usedchild`
  `-t filesystem,volume` syntax.

`-s` *property*
A property for sorting the output by column in ascending order based on the value of the property. The property must be one of the properties described in the "Properties" section, or the special value `name` to sort by the dataset name. Multiple properties can be specified at one time using multiple `-s` property options. Multiple `-s` options are evaluated from left to right in decreasing order of importance.

The following is a list of sorting criteria:

- Numeric types sort in numeric order.
- String types sort in alphabetical order.
- Types inappropriate for a row sort that row to the literal bottom, regardless of the specified ordering.
- If no sorting options are specified the existing behavior of `zfs list` is preserved.

`-S` *property*
Same as the `-s` option, but sorts by property in descending order.

`-t` *type*
A comma-separated list of types to display, where *type* is one of `filesystem`, `snapshot` , `volume`, or `all`. For example, specifying `-t snapshot` displays only snapshots. The following aliases can be used in place of the type specifiers: `fs` (filesystem), `snap` (snapshot), and `vol` (volume).

`zfs mount`
Displays all ZFS file systems currently mounted.

`zfs mount` [`-vOS`] [`-o` *options*] `-a` | *filesystem*
Mounts ZFS file systems. Invoked automatically as part of the boot process.

`-o` *options*
An optional, comma-separated list of mount options to use temporarily for the duration of the mount. See the "Temporary Mount Point Properties" section for details.

`-O`
Perform an overlay mount. See `mount(1M)` for more information.

`-v`
Report mount progress.

`-a`
Mount all available ZFS file systems. Invoked automatically as part of the boot process.

filesystem
Mount the specified filesystem.

A `zfs mount` operation for an encrypted dataset might prompt you for a key, depending on the `keysource` property value. This might occur, for example, if the `keysource` locator is set to `prompt`.

`zfs promote` *clone-filesystem*
Promotes a clone file system to no longer be dependent on its *origin* snapshot. This makes it possible to destroy the file system that the clone was created from. The clone parent-child dependency relationship is reversed, so that the origin file system becomes a clone of the specified file system.

The snapshot that was cloned, and any snapshots previous to this snapshot, are now owned by the promoted clone. The space they use moves from the origin file system to the promoted clone, so enough space must be available to accommodate these snapshots. No new space is consumed by this operation, but the space accounting is adjusted. The promoted clone must not have any conflicting snapshot names of its own. The `rename` subcommand can be used to rename any conflicting snapshots.

`zfs receive` [`-vnFu`] [[`-o` *property=value*] | [`-x` *property*]] ... *filesystem*|*volume*|*snapshot*
`zfs receive` [`-vnFu`] [[`-o` *property=value*] | [`-x` *property*]] ... [`-d` | `-e`] *filesystem*
Creates a snapshot whose contents are as specified in the stream provided on standard input. If a full stream is received, then a new file system is created as well. Streams are created using the `zfs send` subcommand, which by default creates a full stream. `zfs recv` can be used as an alias for `zfs receive`.

If an incremental stream is received, then the destination file system must already exist, and its most recent snapshot must match the incremental stream's source. For ZFS volumes, the destination device link is destroyed and recreated, which means the volume cannot be accessed during the `receive` operation.

When a snapshot replication package stream that is generated by using the `zfs send -R` command is received, any snapshots that do not exist on the sending location are destroyed by using the `zfs destroy -d` command. If `-o` *property=value* or `-x` property is specified, it applies to the effective value of the property throughout the entire subtree of replicated datasets. Effective property values may be set or inherited, depending on the property and whether the dataset is the topmost in the replicated subtree. Received properties are retained in spite of being overridden and may be restored with `zfs inherit -rS` or `zfs send -Rb`.

The name of the snapshot (and file system, if a full stream is received) that this subcommand creates depends on the argument type and the `-d` or `-e` option.

If the argument is a snapshot name, the specified *snapshot* is created. If the argument is a file system or volume name, a snapshot with the same name as the sent snapshot is created within the specified *filesystem* or *volume*. If the `-d` or `-e` option is specified, the snapshot name is determined by appending the sent snapshot's name to the specified filesystem. If the `-d` option is specified, all but the pool name of the sent snapshot path is appended (for example, `b/c@1` appended from sent snapshot `a/b/c@1`), and if the `-e` option is specified, only the tail of the sent snapshot path is appended (for example, `c@1` appended from sent snapshot `a/b/c@1`). In the case of `-d`, any file systems needed to replicate the path of the sent snapshot are created within the specified file system.

`-d`
: Uses all but the first element of the sent snapshot path (all but the pool name) to determine the name of the new snapshot as described in the paragraph above.

`-e`
: Uses the last element of the sent snapshot path to determine the name of the new snapshot as described in the paragraph above.

`-F`
: Forces a rollback of the file system to the most recent snapshot before performing the receive operation. If receiving an incremental replication stream (for example, one generated by `zfs send -R -[iI]`), destroy snapshots and file systems that do not exist on the sending side.

`-n`
: Do not actually receive the stream. This can be useful in conjunction with the `-v` option to verify the name the receive operation would use.

`-o` *property=value*
: Sets the specified property as if the command *property=value* is invoked at the same time the received dataset is created from the non-incremental `send` stream or updated from the incremental `send` stream. Any editable ZFS property can also be set at receive time. Set-once properties bound to the received data, such as `normalization` and `casesensitivity`, cannot be set at receive time even when the datasets are newly created by `zfs receive`. Multiple `-o` options can be specified. An error results if the same property is specified in multiple `-o` or `-x` options.

`-u`
: File system that is associated with the received stream is not mounted.

`-v`
: Print verbose information about the stream and the time required to perform the receive operation.

`-x` *property*
Ensures that the effective value of the specified property after the `receive` is unaffected by the value of that property in the `send` stream (if any), as if the property had been excluded from the `send` stream. If the specified property is not present in the `send` stream, this option does nothing. If a received property needs to be overridden, the effective value can be set or inherited, depending on the property. In the case of an incremental update, `-x` leaves any existing local setting or explicit inheritance unchanged (since the received property is already overridden). All `-o` restrictions apply equally to `-x`.

`zfs release` [`-r`] *tag snapshot...*
Removes a single reference, named with the *tag* argument, from the specified snapshot or snapshots. The tag must already exist for each snapshot.

If a hold exists on a snapshot, attempts to destroy that snapshot by using the `zfs destroy` command return `EBUSY`.

`-r`
Recursively releases a hold with the given tag on the snapshots of all descendent file systems.

`zfs rename` *filesystem*|*volume*|*snapshot*
filesystem|*volume*|*snapshot*
`zfs rename` [`-p`] *filesystem*|*volume filesystem*|*volume*
Renames the given dataset. The new target can be located anywhere in the ZFS hierarchy, with the exception of snapshots. Snapshots can only be renamed within the parent file system or volume. When renaming a snapshot, the parent file system of the snapshot does not need to be specified as part of the second argument. Renamed file systems can inherit new mount points, in which case they are unmounted and remounted at the new mount point.

`-p`
Creates all the nonexistent parent datasets. Datasets created in this manner are automatically mounted according to the `mountpoint` property inherited from their parent.

`zfs rename -r` *snapshot snapshot*
Recursively renames the snapshots of all descendent datasets. Snapshots are the only dataset that can be renamed recursively.

`zfs rename` *share share*
Renames the specified share to a new share name.

`zfs rollback` [`-rRf`] *snapshot*
Rolls back the given dataset to a previous snapshot. When a dataset is rolled back, all data that has changed since the snapshot is discarded, and the dataset reverts to the state at the

time of the snapshot. By default, the command refuses to roll back to a snapshot other than the most recent one. In order to do so, all intermediate snapshots must be destroyed by specifying the `-r` option.

The `-rR` options do not recursively destroy the child snapshots of a recursive snapshot. Only the top-level recursive snapshot is destroyed by either of these options. To completely roll back a recursive snapshot, you must rollback the individual child snapshots.

`-r`
Recursively destroys any snapshots more recent than the one specified.

`-R`
Recursively destroys any more recent snapshots, as well as any clones of those snapshots.

`-f`
Used with the `-R` option to force an unmount of any clone file systems that are to be destroyed.

`zfs send` [`-DRbpv`] [`-[iI]` *snapshot*] *snapshot*
`zfs send -r` [`-Dbcpv`] [`-i` *snapshot*] *snapshot*
Creates a stream representation of the second *snapshot*, which is written to standard output. The output can be redirected to a file or to a different system (for example, using `ssh(1)`. By default, a full stream is generated.

`-b`
Sends only received property values whether or not they are overridden by local settings, but only if the dataset has ever been received. Use this option when you want `zfs receive` to restore received properties backed up on the sent dataset and to avoid sending local settings that may have nothing to do with the source dataset, but only with how the data is backed up.

`-c`
Creates a self-contained stream. A self-contained stream is one that is not dependent on any datasets not included in the stream package. Valid only with the `-r` option. If used with the `-i` option, the stream will be dependent on the snapshot specified as an argument to the that option.

See the "ZFS Streams" section of the *ZFS Administration Guide* for details.

`-D`
Performs `dedup` processing on the stream. Deduplicated streams cannot be received on systems that do not support the stream deduplication feature.

See the "ZFS Streams" section of the *ZFS Administration Guide* to understand how a replication stream package differs from a recursive stream package.

`-i` *snapshot*
Generates an incremental stream from the first *snapshot* to the second *snapshot*. The incremental source (the first *snapshot*) can be specified as the last component of the snapshot name (for example, the part after the `@`), and it is assumed to be from the same file system as the second *snapshot*.

If the destination is a clone, the source may be the origin snapshot, which must be fully specified (for example, `pool/fs@origin`, not just `@origin`).

`-I` *snapshot*
Generates a stream package that sends all intermediary snapshots from the first snapshot to the second snapshot. For example, `-I @a fs@d` is similar to `-i @a fs@b; -i @b fs@c; -i @c fs@d`. The incremental source snapshot may be specified as with the `-i` option.

`-R`
Generates a replication stream package that replicates the specified file system, and all descendent file systems, up to the named snapshot. When received, all properties, snapshots, descendent file systems, and clones are preserved.

If the `-i` or `-I` flags are used in conjunction with the `-R` flag, an incremental replication stream is generated. The current values of properties, and current snapshot and file system names are set when the stream is received. If the `-F` flag is specified when this stream is received, snapshots and file systems that do not exist on the sending side are destroyed.

`-r`
Generates a recursive stream package. A recursive stream package contains a series of full and/or incremental streams. When received, all properties and descendent file systems are preserved. Unlike with the replication stream packages generated with the `-R` flag, intermediate snapshots are not preserved unless the intermediate snapshot is the origin of a clone that is included in the stream.

If the `-i` option is used in conjunction with the `-r` option, an incremental recursive stream is generated. The current values of properties as well as current snapshot and file system names are set when the stream is received. If the `-F` option is specified when this stream is received, snapshots and file systems that do not exist on the sending side are destroyed. The `-I` option cannot be used in conjunction with the `-r` option.

When combined with the `-c` option, a self-contained recursive stream package is created. If both the `-c` and `-i` options are used, file systems and volumes that do not have the snapshot specified with the `-i` option are sent as self-contained streams.

See the “ZFS Streams” section of the *ZFS Administration Guide* to understand how a recursive stream package differs from a replication stream package.

`-p`
Sends properties.

`-v`
Displays verbose information about the stream package generated.

The format of the stream is committed. You will be able to receive your streams on future versions of ZFS.

`zfs set` [`-r`] *property=value filesystem|volume|snapshot ...*
Sets the property to the given value for each dataset. Only some properties can be edited. See the "Properties" section for more information on what properties can be set and acceptable values. Numeric values can be specified as exact values, or in a human-readable form with a suffix of `B`, `K`, `M`, `G`, `T`, `P`, `E`, `Z` (for bytes, kilobytes, megabytes, gigabytes, terabytes, petabytes, exabytes, or zettabytes, respectively). User properties can be set on snapshots. For more information, see the "User Properties" section.

`-r`
Recursively apply the effective value of the setting throughout the subtree of child datasets. The effective value may be set or inherited, depending on the property. Use the `zfs help -l properties` command to review whether a property is settable or inheritable.

`zfs share` [`-u`] `-o` *property=value ... filesystem%share*
`zfs share` *filesystem|mountpoint|filesystem%share*
`zfs share -a` | *filesystem*
For a full description of `zfs share` syntax and examples and setting the `share.nfs` or `share.smb` property, see `zfs_share(1M)`.

`zfs snapshot` [`-r`] [`-o` *property=value*] *... filesystem@snapname|volume@snapname*
Creates a snapshot with the given name. All previous modifications by successful system calls to the file system are part of the snapshot. `zfs snap` can be used as an alias for `zfs snapshot`. See the "Snapshots" section for details.

`-r`
Recursively creates snapshots of all descendent datasets. Snapshots are taken atomically, so that all recursive snapshots correspond to the same moment in time.

`-o` *property=value*
Sets the specified property; see `zfs create` for details.

`zfs unallow` [`-rldug`] `everyone|`*user|group*[,...] [*perm|@setname*[, ...]] *filesystem|volume*
`zfs unallow` [`-rld`] `-e` [*perm|@setname* [,...]] *filesystem|volume*
`zfs unallow` [`-r`] `-c` [*perm|@setname*[,...]]
filesystem|volume
`zfs unallow` [`-r`] `-s` *@setname* [*perm|@setname*[,...]]
filesystem|volume
For a full description of the `zfs unallow` syntax and examples, see `zfs_allow(1M)`.

`zfs unmount` [`-f`] `-a` | *filesystem|mountpoint*
Unmounts currently mounted ZFS file systems. Invoked automatically as part of the shutdown process.

`-f`
Forcefully unmount the file system, even if it is currently in use.

`-a`
Unmounts all available ZFS file systems. Invoked automatically as part of the boot process.

filesystem|*mountpoint*
Unmounts the specified filesystem. The command can also be given a path to a ZFS file system mount point on the system.

For an encrypted dataset, the key is not unloaded when the file system is unmounted. To unload the key, see `zfs key`.

`zfs unshare` *filesystem*|*mountpoint*|*filesystem%share*
`zfs unshare -a | -r` *filesystem*
For a full description of `zfs unshare` syntax and examples, see `zfs_share(1M)`.

`zfs upgrade`
Identifies a file system version, which determines available file system features in the currently running software release. You can continue to use older file system versions, but some features might not be available. A file system can be upgraded by using the `zfs upgrade -a` command. You will not be able to access a file system of a later version on a system that runs an earlier software version.

`zfs upgrade [-v]`
Displays ZFS file system versions that are supported by the current software. The current ZFS file system versions and all previously supported versions are displayed, along with an explanation of the features provided with each version.

`zfs upgrade [-r] [-V` *version*`] [-a |` *filesystem*`]`
Upgrades file systems to a new, on-disk version. Upgrading a file system means that it will no longer be accessible on a system running an older software version. A `zfs send` stream that is generated from a new file system snapshot cannot be accessed on a system that runs an older software version.

In general, the file system version is independent of the pool version. See `zpool(1M)` for information on the `zpool upgrade` command.

In some cases, the file system version and the pool version are interrelated and the pool version must be upgraded before the file system version can be upgraded.

`-a`
Upgrades all file systems on all imported pools.

filesystem
Upgrades the specified file system.

`-r`
Upgrades the specified file system and all descendent file systems.

`-V` *version*
Upgrades to the specified *version*. If the `-V` flag is not specified, this command upgrades to the most recent version. This option can only be used to increase the version number, and only up to the most recent version supported by this software.

`zfs userspace` [`-hniHp`] [`-o` *field*[,...]] [`-sS` *field*]... [`-t` *type* [,...]] *filesystem* | *snapshot*
Displays space consumed by, and quotas on, each user in the specified filesystem or snapshot. This corresponds to the `userused@`*user* and `userquota@`*user* properties.

`-h`
Displays syntax help message and exit.

`-n`
Prints numeric ID instead of user/group name.

`-H`
Does not print headers, use tab-delimited output.

`-p`
Uses exact (parseable) numeric output.

`-o` *field*[,...]
Displays only the specified fields from the following set, `type,name,used,quota`. The default is to display all fields.

`-s` *field*
Sorts output by this field. The *s* and *S* flags may be specified multiple times to sort first by one field, then by another. The default is `-s type -s name`.

`-S` *field*
Sorts by this field in reverse order. See `-s`.

`-t` *type*[,...]
Prints only the specified types from the following set, `all,posixuser,smbuser,posixgroup,smbgroup`.

The default is `-t posixuser,smbuser`

The default can be changed to include group types.

`-i`
Translates SID to POSIX ID. The POSIX ID may be ephemeral if no mapping exists. Normal POSIX interfaces (for example, `stat(2)`, `ls -l`) perform this translation, so the `-i` option allows the output from `zfs userspace` to be compared directly with those utilities. However, `-i` may lead to confusion if some files were created by an SMB user before a SMB-to-POSIX name mapping was established. In such a case, some files are owned by the SMB entity and some by the POSIX entity. However, the `-i` option will report that the POSIX entity has the total usage and quota for both.

Examples

EXAMPLE 1 Creating a ZFS File System Hierarchy

The following commands create a file system named `pool/home` and a file system named `pool/home/bob`. The mount point `/export/home` is set for the parent file system, and is automatically inherited by the child file system.

```
# zfs create pool/home
# zfs set mountpoint=/export/home pool/home
# zfs create pool/home/bob
```

EXAMPLE 2 Creating a ZFS Snapshot

The following command creates a snapshot named `yesterday`. This snapshot is mounted on demand in the `.zfs/snapshot` directory at the root of the `pool/home/bob` file system.

```
# zfs snapshot pool/home/bob@yesterday
```

EXAMPLE 3 Creating and Destroying Multiple Snapshots

The following command creates snapshots named `yesterday` of `pool/home` and all of its descendent file systems. Each snapshot is mounted on demand in the `.zfs/snapshot` directory at the root of its file system. The second command destroys the newly created snapshots.

```
# zfs snapshot -r pool/home@yesterday
# zfs destroy -r pool/home@yesterday
```

EXAMPLE 4 Disabling and Enabling File System Compression

The following command disables the `compression` property for all file systems under `pool/home`. The next command explicitly enables `compression` for `pool/home/anne`.

```
# zfs set compression=off pool/home
# zfs set compression=on pool/home/anne
```

EXAMPLE 5 Listing ZFS Datasets

The following command lists all active file systems and volumes in the system. Snapshots are displayed if the `listsnaps` property is `on`. The default is `off`. See `zpool(1M)` for more information on pool properties.

```
# zfs list
   NAME                     USED  AVAIL  REFER  MOUNTPOINT
   pool                     450K   457G    18K  /pool
   pool/home                315K   457G    21K  /export/home
   pool/home/anne            18K   457G    18K  /export/home/anne
   pool/home/bob            276K   457G   276K  /export/home/bob
```

EXAMPLE 6 Setting a Quota on a ZFS File System

The following command sets a quota of 30 GB for `pool/home/bob`.

```
# zfs set quota=30G pool/home/bob
```

EXAMPLE 7 Listing ZFS Properties

The following command lists all properties for `pool/home/bob`.

```
# zfs get all pool/home/bob
NAME           PROPERTY              VALUE                  SOURCE
pool/home/bob  aclinherit            restricted             default
pool/home/bob  aclmode               discard                default
pool/home/bob  atime                 on                     default
pool/home/bob  available             30.0G                  -
pool/home/bob  canmount              on                     default
pool/home/bob  casesensitivity       mixed                  -
pool/home/bob  checksum              on                     default
pool/home/bob  compression           on                     local
pool/home/bob  compressratio         1.00x                  -
pool/home/bob  copies                1                      default
pool/home/bob  creation              Tue Jul  3 10:39 2012  -
pool/home/bob  dedup                 off                    default
pool/home/bob  devices               on                     default
pool/home/bob  encryption            off                    -
pool/home/bob  exec                  on                     default
pool/home/bob  keychangedate         -                      default
pool/home/bob  keysource             none                   default
pool/home/bob  keystatus             none                   -
pool/home/bob  logbias               latency                default
pool/home/bob  mlslabel              none                   -
pool/home/bob  mounted               yes                    -
pool/home/bob  mountpoint            /pool/home/bob         default
pool/home/bob  multilevel            off                    -
pool/home/bob  nbmand                off                    default
pool/home/bob  normalization         none                   -
pool/home/bob  primarycache          all                    default
pool/home/bob  quota                 30G                    local
pool/home/bob  readonly              off                    default
pool/home/bob  recordsize            128K                   default
pool/home/bob  referenced            31K                    -
pool/home/bob  refquota              none                   default
pool/home/bob  refreservation        none                   default
pool/home/bob  rekeydate             -                      default
pool/home/bob  reservation           none                   default
pool/home/bob  rstchown              on                     default
pool/home/bob  secondarycache        all                    default
pool/home/bob  setuid                on                     default
pool/home/bob  shadow                none                   -
pool/home/bob  share.*               ...                    inherited
pool/home/bob  snapdir               hidden                 default
pool/home/bob  sync                  standard               default
pool/home/bob  type                  filesystem             -
pool/home/bob  used                  31K                    -
```

EXAMPLE 7 Listing ZFS Properties *(Continued)*

```
pool/home/bob  usedbychildren        0                    -
pool/home/bob  usedbydataset         31K                  -
pool/home/bob  usedbyrefreservation  0                    -
pool/home/bob  usedbysnapshots       0                    -
pool/home/bob  utf8only              off                  -
pool/home/bob  version               6                    -
pool/home/bob  vscan                 off                  default
pool/home/bob  xattr                 on                   default
pool/home/bob  zoned                 off                  default
```

The following command gets a single property value.

```
# zfs get -H -o value compression pool/home/bob
on
```

The following command lists all properties with local settings for `pool/home/bob`.

```
# zfs get -r -s local -o name,property,value all pool/home/bob
NAME            PROPERTY  VALUE
pool/home/bob   compression  on
pool/home/bob   quota     30G
```

EXAMPLE 8 Rolling Back a ZFS File System

The following command reverts the contents of `pool/home/anne` to the snapshot named `yesterday`, deleting all intermediate snapshots.

```
# zfs rollback -r pool/home/anne@yesterday
```

EXAMPLE 9 Creating a ZFS Clone

The following command creates a writable file system whose initial contents are the same as `pool/home/bob@yesterday`.

```
# zfs clone pool/home/bob@yesterday pool/clone
```

EXAMPLE 10 Promoting a ZFS Clone

The following commands illustrate how to test out changes to a file system, and then replace the original file system with the changed one, using clones, clone promotion, and renaming:

```
# zfs create pool/project/production
  populate /pool/project/production with data
# zfs snapshot pool/project/production@today
# zfs clone pool/project/production@today pool/project/beta
make changes to /pool/project/beta and test them
# zfs promote pool/project/beta
# zfs rename pool/project/production pool/project/legacy
# zfs rename pool/project/beta pool/project/production
```

EXAMPLE 10 Promoting a ZFS Clone *(Continued)*

```
once the legacy version is no longer needed, it can be destroyed
# zfs destroy pool/project/legacy
```

EXAMPLE 11 Inheriting ZFS Properties

The following command causes `pool/home/bob` and `pool/home/anne` to inherit the `checksum` property from their parent.

```
# zfs inherit checksum pool/home/bob pool/home/anne
```

EXAMPLE 12 Remotely Replicating ZFS Data

The following commands send a full stream and then an incremental stream to a remote machine, restoring them into `poolB/received/fs@a`and `poolB/received/fs@b`, respectively. `poolB` must contain the file system `poolB/received`, and must not initially contain `poolB/received/fs`.

```
# zfs send pool/fs@a | \
   ssh host zfs receive poolB/received/fs@a
# zfs send -i a pool/fs@b | ssh host \
   zfs receive poolB/received/fs
```

The above syntax assumes that `sshd` has been configured to allow remote root access.

EXAMPLE 13 Using the `zfs receive -d` Option

The following command sends a full stream of `poolA/fsA/fsB@snap` to a remote machine, receiving it into `poolB/received/fsA/fsB@snap`. The `fsA/fsB@snap` portion of the received snapshot's name is determined from the name of the sent snapshot. `poolB` must contain the file system `poolB/received`. If `poolB/received/fsA` does not exist, it is created as an empty file system.

```
# zfs send poolA/fsA/fsB@snap | \
   ssh host zfs receive -d poolB/received
```

EXAMPLE 14 Setting User Properties

The following example sets the user–defined `com.example:department` property for a dataset.

```
# zfs set com.example:department=12345 tank/accounting
```

EXAMPLE 15 Performing a Rolling Snapshot

The following example shows how to maintain a history of snapshots with a consistent naming scheme. To keep a week's worth of snapshots, the user destroys the oldest snapshot, renames the remaining snapshots, and then creates a new snapshot, as follows:

```
# zfs destroy -r pool/users@7daysago
# zfs rename -r pool/users@6daysago @7daysago
# zfs rename -r pool/users@5daysago @6daysago
```

EXAMPLE 15 Performing a Rolling Snapshot *(Continued)*

```
# zfs rename -r pool/users@4daysago @5daysago
# zfs rename -r pool/users@3daysago @4daysago
# zfs rename -r pool/users@2daysago @3daysago
# zfs rename -r pool/users@yesterday @2daysago
# zfs rename -r pool/users@today @yesterday
# zfs snapshot -r pool/users@today
```

EXAMPLE 16 Displaying ZFS Snapshot Differences

The following example is output of the `zfs diff -F` and `-t` options specified:

```
# zfs diff -Ft myfiles@snap1
1269962501.206726811   M   /   /myfiles/
1269962444.207369955   M   F   /myfiles/link_to_me      (+1)
1269962499.207519034   R       /myfiles/rename_me -> /myfiles/renamed
1269962431.813566720   -   F   /myfiles/delete_me
1269962518.666905544   +   F   /myfiles/new_file
1269962501.393099817   +   |   /myfiles/new_pipe
```

Exit Status

The following exit values are returned:

`0`
Successful completion.

`1`
An error occurred.

`2`
Invalid command line options were specified.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/file-system/zfs
Interface Stability	Committed

See Also

`chmod(1)`, `chown(1)`, `pktool(1)`, `setlabel(1)`, `ssh(1)`, `mount(1M)`, `shadowd(1M)`, `share(1M)`, `share_nfs(1M)`, `share_smb(1M)`, `unshare(1M)`, `zonecfg(1M)`, `zpool(1M)`, `chmod(2)`, `chown(2)`, `stat(2)`, `write(2)`, `fsync(3C)`, `setflabel(3TSOL)`, `dfstab(4)`, `vfstab(4)`, `attributes(5)`

See the `gzip`(1) man page, which is not part of the SunOS man page collection.

For information about other ZFS features, see `zfs_allow(1M)`, `zfs_encrypt(1M)`, `zfs_share(1M)`, and the *Oracle Solaris 11.1 Administration: ZFS File Systems*.

Notes A file described as modified by the `diff` subcommand might have been modified in multiple ways. Any action that causes a change in the `st_ctim` (see `stat(2)`) is a basis for reporting a modification.

Name zfs_allow – delegates ZFS file system administration permission to non-privileged users

Synopsis

```
zfs help subcommand | help | property property-name | permission

zfs help -l properties

zfs allow filesystem|volume

zfs allow [-ldug] everyone|user|group[,...] perm|@setname[,...]
     filesystem|volume

zfs allow [-ld] -e perm|@setname[,...] filesystem|volume

zfs allow -c perm|@setname[,...] filesystem|volume

zfs allow -s @setname perm|@setname[,...] filesystem|volume

zfs unallow [-rldug] everyone|user|group[,...] [perm|@setname[,... ]]
     filesystem|volume

zfs unallow [-rld] -e [perm|@setname[,... ]] filesystem|volume

zfs unallow [-r] -c [perm|@setname[ ... ]] filesystem|volume

zfs unallow [-r] -s @setname [perm|@setname[,... ]] filesystem|volume
```

Description The `zfs allow` command can be used to delegate permissions to non-privileged users for administering ZFS file systems in a ZFS storage pool, as described in `zpool(1M)`. You can use the `zfs unallow` command to revoke administrative permissions.

Permissions are generally the ability to use a ZFS subcommand or change a ZFS property. The following permissions are available:

```
# zfs help permissions
The following delegated permissions are supported:

NAME            TYPE           NOTES
allow           subcommand     Must also have the permission that is being
                               allowed
clone           subcommand     Must also have the 'create' ability and 'mount'
                               ability in the origin file system
create          subcommand     Must also have the 'mount' ability
destroy         subcommand     Must also have the 'mount' ability
diff            subcommand     Allows lookup of paths within a dataset,
                               given an object number. Ordinary users need this
                               in order to use zfs diff
hold            subcommand     Allows adding a user hold to a snapshot
mount           subcommand     Allows mount/umount of ZFS datasets
promote         subcommand     Must also have the 'mount'
                               and 'promote' ability in the origin file system
receive         subcommand     Must also have the 'mount' and 'create' ability
release         subcommand     Allows releasing a user hold which
                               might destroy the snapshot
rename          subcommand     Must also have the 'mount' and 'create'
```

```
                                ability in the new parent
rollback        subcommand      Allows rolling back datasets to previously-taken snapshots
send            subcommand      Allows sending of snapshots
share           subcommand      Allows sharing file systems over NFS or SMB
                                protocols
snapshot        subcommand      Allows taking of snapshots
groupquota      other           Allows accessing any groupquota@... property
groupused       other           Allows reading any groupused@... property
key             other           Allows load/unload of dataset key
keychange       other           Allows key change operations
userprop        other           Allows changing any user property
userquota       other           Allows accessing any userquota@... property
userused        other           Allows reading any userused@... property

The following properties can have delegated permissions applied:
aclinherit      aclmode         atime           canmount
casesensitivity checksum        compression     copies
dedup           devices         encryption      exec
keysource       logbias         mountpoint      multilevel
nbmand          normalization   primarycache    quota
readonly        recordsize      refquota        refreservation
reservation     rstchown        secondarycache  setuid
shadow          sharenfs        sharesmb        snapdir
sync            utf8only        version         volblocksize
volsize         vscan           xattr           zoned
```

Subcommands All subcommands that modify state are logged persistently to the pool in their original form.

`zfs ?`
Displays a help message.

`zfs help` *command* | help | *property property-name* | *permission*
Displays `zfs` command usage information. You can display help for a specific command, property, or delegated permission. If you display help for a specific command or property, the command syntax or property value is displayed. Using `zfs help` without any arguments displays a complete list of `zfs` commands.

`zfs help -l properties`
Displays `zfs` property information, including whether the property value is editable and inheritable, and their possible values.

`zfs allow` *filesystem* | *volume*
Displays permissions that have been delegated on the specified filesystem or volume. See the other forms of `zfs allow` for more information.

`zfs allow` [`-ldug`] `everyone`|*user*|*group*[,...] *perm*|*@setname*[,...] *filesystem*| *volume*
`zfs allow` [`-ld`] `-e` *perm*|*@setname*[,...] *filesystem* | *volume*
Delegates ZFS administration permission for the file systems to non-privileged users.

[-ug] everyone|*user*|*group*[,...]
Specifies to whom the permissions are delegated. Multiple entities can be specified as a comma-separated list. If neither of the -ug options are specified, then the argument is interpreted preferentially as the keyword everyone, then as a user name, and lastly as a group name. To specify a user or group named "everyone", use the -u or -g options. To specify a group with the same name as a user, use the -g options.

[-e] *perm*|*@setname*[,...]
Specifies that the permissions be delegated to everyone. Multiple permissions may be specified as a comma-separated list. Permission names are the same as ZFS subcommand and property names. See the property list below. Property set names, which begin with an at sign (@) , may be specified. See the -s form below for details.

[-ld] *filesystem*|*volume*
Specifies where the permissions are delegated. If neither of the -ld options are specified, or both are, then the permissions are allowed for the file system or volume, and all of its descendents. If only the -l option is used, then is allowed "locally" only for the specified file system. If only the -d option is used, then is allowed only for the descendent file systems.

zfs allow -c *perm*|*@setname*[,...] *filesystem*|*volume*
Sets "create time" permissions. These permissions are granted (locally) to the creator of any newly-created descendent file system.

zfs allow -s *@setname perm*|*@setname*[,...] *filesystem*|*volume*
Defines or adds permissions to a permission set. The set can be used by other zfs allow commands for the specified file system and its descendents. Sets are evaluated dynamically, so changes to a set are immediately reflected. Permission sets follow the same naming restrictions as ZFS file systems, but the name must begin with an "at sign" (@), and can be no more than 64 characters long.

zfs unallow [-rldug] everyone|*user*|*group*[,...] [*perm*|*@setname*[, ...]] *filesystem*|*volume*
zfs unallow [-rld] -e [*perm*|*@setname* [,...]] *filesystem*|*volume*
zfs unallow [-r] -c [*perm*|*@setname*[,...]]
filesystem|*volume*
Removes permissions that were granted with the zfs allow command. No permissions are explicitly denied, so other permissions granted are still in effect. For example, if the permission is granted by an ancestor. If no permissions are specified, then all permissions for the specified *user*, *group*, or everyone are removed. Specifying everyone (or using the -e option) only removes the permissions that were granted to everyone, not all permissions for every user and group. See the zfs allow command for a description of the -ldugec options.

-r
Recursively remove the permissions from this file system and all descendents.

zfs unallow [-r] -s *@setname* [*perm*|*@setname*[,...]]
filesystem|*volume*

Removes permissions from a permission set. If no permissions are specified, then all permissions are removed, thus removing the set entirely.

Examples

EXAMPLE 1 Delegating ZFS Administration Permissions on a ZFS Dataset

The following example shows how to set permissions so that user anne can create, destroy, mount, and take snapshots on pool/home/anne. The permissions on pool/home/anne are also displayed.

```
# zfs allow anne create,destroy,mount,snapshot pool/home/anne
# zfs allow pool/home/anne
---- Permissions on pool/home/anne --------------------------------------
Local+Descendent permissions:
        user anne create,destroy,mount,snapshot
```

Because the pool/home/anne mount point permission is set to 755 by default, user anne will be unable to mount file systems under pool/home/anne. Set an ACL similar to the following syntax to provide mount point access:

```
# chmod A+user:anne:add_subdirectory:allow /pool/home/anne
```

EXAMPLE 2 Delegating Create Time Permissions on a ZFS Dataset

The following example shows how to grant anyone in the group staff to create file systems in pool/home. This syntax also allows staff members to destroy their own file systems, but not destroy anyone else's file system. The permissions on pool/home are also displayed.

```
# zfs allow staff create,mount pool/home
# zfs allow -c destroy pool/home
# zfs allow pool/home
---- Permissions on pool/home --------------------------------------------
Create time permissions:
        destroy
Local+Descendent permissions:
        group staff create,mount
```

EXAMPLE 3 Defining and Granting a Permission Set on a ZFS Dataset

The following example shows how to define and grant a permission set on the pool/home file system. The permissions on pool/home are also displayed.

```
# zfs allow -s @pset create,destroy,snapshot,mount pool/home
# zfs allow staff @pset pool/home
# zfs allow pool/home
---- Permissions on pool/home --------------------------------------------
Permission sets:
        @pset create,destroy,mount,snapshot
Create time permissions:
        destroy
Local+Descendent permissions:
        group staff @pset,create,mount
```

EXAMPLE 4 Delegating Property Permissions on a ZFS Dataset

The following example shows to grant the ability to set quotas and reservations on the `tank/users` file system. The permissions on `tank/users` are also displayed.

```
# zfs allow mark quota,reservation tank/users
# zfs allow tank/users
---- Permissions on tank/users --------------------------------------
Local+Descendent permissions:
        user mark quota,reservation
mark% zfs set quota=10G tank/users/tim
mark% zfs get quota tank/users/tim
NAME            PROPERTY  VALUE  SOURCE
tank/users/tim  quota     10G    local
```

EXAMPLE 5 Removing ZFS Delegated Permissions on a ZFS Dataset

The following example shows how to remove the snapshot permission from the `@pset` permission set for the `staff` group on the `pool/home` file system. The permissions on `pool/home` are also displayed.

```
# zfs unallow -s @pset snapshot pool/home
# zfs allow pool/home
---- Permissions on pool/home ---------------------------------------
Permission sets:
        @pset create,destroy,mount
Create time permissions:
        destroy
Local+Descendent permissions:
        group staff @pset,create,mount
```

Exit Status

The following exit values are returned:

`0`
: Successful completion.

`1`
: An error occurred.

`2`
: Invalid command line options were specified.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/file-system/zfs
Interface Stability	Committed

See Also `zfs`(1M), `zpool`(1M), `chmod`(2), `chown`(2), `attributes`(5)

For information about using other ZFS features, see zfs_encrypt.1m, zfs_share.1m, `zfs`(1M) and the *Oracle Solaris 11.1 Administration: ZFS File Systems*.

Name zfs_encrypt – encrypting ZFS file systems

Synopsis

```
zfs [-?]

zfs help subcommand | help | property property-name | permission

zfs help -l properties

zfs create -o encryption=on [-o keysource=raw | hex |
     passphrase,prompt | file://|pkcs11:|https://] ... dataset

zfs clone [-p] [-K] [-o property=value] ... snapshot filesystem|volume

zfs get [-r|-d depth][-Hp][-o all | field[,...]] [-s source[,...]]
     all | property[,...] filesystem|volume|snapshot ...

zfs key -l | {-a | [-r] filesystem|volume}

zfs key -u [-f] {-a | [-r] filesystem|volume}

zfs key -c [-o keysource=value] {-a | [-r] filesystem|volume}

zfs key -K {-a | [-r] filesystem|volume}

zfs mount

zfs mount [-vO] [-o options] -a | filesystem

zfs unmount [-f] -a | filesystem|mountpoint
```

Description The `zfs create -o encryption` command encrypts a newly created ZFS dataset within a ZFS storage pool, as described in zpool(1M).

Encryption

Encryption is the process in which data is encoded for privacy and a key is needed by the data owner to access the encoded data. You can set an encryption policy when a ZFS dataset is created, but the policy cannot be changed. See the `encryption` and `keysource` property descriptions in the "Native Properties" section for details.

Dataset encryption is inherited permanently and cannot be removed during dataset cloning. When receiving a replicated dataset stream, the destination dataset must have encryption enabled if encryption is desired. Otherwise, the data is stored as clear text. A fully replicated stream of an encrypted dataset results in an encrypted dataset but under a newly generated key. The stream itself is not encrypted.

Native ZFS Encryption Properties

The following native properties related to ZFS encryption consist of read-only statistics about the dataset. These properties cannot be set nor inherited. Native properties apply to all dataset types unless otherwise noted. For a full description and list of ZFS native properties, see zfs(1M).

`keychangedate`
: The date of the last wrapping key change from a `zfs key -c` operation on a given dataset. If no key change operation has been performed, `keychangedate` is the same as the creation date.

`keystatus`
Identifies the encryption key status for the dataset. The availability of a dataset's key is indicated by showing the status of `available` or `unavailable`. For datasets that do not have encryption enabled, `none` is displayed.

`mounted`
For file systems, indicates whether the file system is currently mounted. This property can be either `yes` or `no`.

`rekeydate`
The date of the last data encryption key change from a `zfs key -K` or `zfs clone -K` operation on this dataset. If no rekey operation has been performed, `rekeydate` is the same as `creation` date.

The following properties cannot be changed after the file system is created and, therefore, should be set when the file system is created. If the properties are not set with the `zfs create` or `zpool create` commands, these properties are inherited from the parent dataset. If the parent dataset lacks these properties due to having been created prior to these features being supported, the new file system will have the default values for these properties.

`encryption=off` | `on` | `aes-128-ccm` | `aes-192-ccm` | `aes-256-ccm` | `aes-128-gcm` | `aes-192-gcm` | `aes-256-gcm`
Defines the encryption algorithm and key length that is used for the encrypted dataset. The `on` value is equal to `aes-128-ccm`. The default value is `off`. When encryption is set to a value other than `off`, the `checksum` property is set to `sha256+mac` and becomes `readonly`.

You must grant the `checksum`, `encryption`, and `keysource` permissions to delegate the `encryption` property.

The following properties must be specified at creation time and can modified by using special commands:

`keysource=`*raw* | *hex* | *passphrase*,*prompt* | *file://* | `pkcs11:` | `https://`
Defines the format and location of the key that wraps the dataset keys. The key must be present when the dataset is created, mounted, or loaded by using the `zfs key -l` command.

The `keysource` property accepts two values: `format` determines how the key is presented; `locator` identifies where the key is coming from.

`format` accepts three values:

- *raw*: the raw key bytes
- *hex*: a hexadecimal key string
- *passphrase*: a character string that generates a key

`locator` accepts two values:

- `prompt`: You are prompted for a key or a passphrase when the dataset is created or mounted

- `file:///`*filename*: the key or a passphrase file location in a file system
- `pkcs11`: A URI describing the location of a key or a passphrase in a PKCS#11 token
- `https://`*location*: The key or a passphrase file location on a secure server. Transporting key information in the clear using this method is not recommended. A `GET` on the URL returns just the key value or the passphrase, according to what was requested in the format part of the `keysource`.

  When using an `https://` locator for the `keysource`, the certificate that the server presents must be one that is trusted by libcurl and OpenSSL. Add your own trust anchor or self signed certificate to the certificate store in `/etc/openssl/certs`. Place the PEM format certificate into the `/etc/certs/CA` directory and run the `svcadm refresh ca-certificates` command.

See "Examples" for examples of creating a key by using the `https://` locator.

To change the wrapping key value or the key, you must run the `zfs key -c` command. If only the key location needs to changed, for example, a filename change, then use the `zfs set` command with the `keysource` property. Note that no checking is performed by ZFS when only the key location is changed with the `zfs set` command, such as whether the new location has a valid wrapping key.

If `keysource` is not specified and not inherited, then the default `keysource` is set to `passphrase,prompt` for a dataset that has encryption on and is set to `none` for a dataset that has encryption off.

Subcommands All subcommands that modify state are logged persistently to the pool in their original form.

`zfs ?`
Displays a help message.

`zfs help` *command* | help | *property property-name* | *permission*
Displays `zfs` command usage information. You can display help for a specific command, property, or delegated permission. If you display help for a specific command or property, the command syntax or property value is displayed. Using `zfs help` without any arguments displays a complete list of `zfs` commands.

`zfs help -l properties`
Displays `zfs` property information, including whether the property value is editable and inheritable, and their possible values.

`zfs create` [`-p`] [`-o` *encryption*`=on`] [`-o keysource=`*raw* | *hex* | *passphrase,prompt* | *file://* | `pkcs11:` | `https://`] ... *filesystem*
Creates a new ZFS file system with encryption enabled, which uses `aes-128-ccm` See the encryption property description for a list of supported encryption algorithms.

`-p`
Creates all the non-existing parent datasets. Datasets created in this manner are automatically mounted according to the `mountpoint` property inherited from their

parent. Any property specified on the command line using the `-o` option is ignored. If the target filesystem already exists, the operation completes successfully.

`-o` *encryption=value*
Sets the encryption property to *value*. Multiple `-o` options can be specified. An error results if the same property is specified in multiple `-o` options.

`zfs clone` [`-p`] [`-K`] [`-o` *property=value*] ... *snapshot filesystem|volume*
Creates a clone of the given snapshot. See the "Clones" section for details. The target dataset can be located anywhere in the ZFS hierarchy, and is created as the same type as the original.

`-p`
Creates all the non-existing parent datasets. Datasets created in this manner are automatically mounted according to the `mountpoint` property inherited from their parent. If the target filesystem or volume already exists, the operation completes successfully.

`-o` *property=value*
Sets the specified property; see `zfs create` for details.

`-K`
Creates a new data encryption key in the keychain for this dataset. Data written in the clone uses the new data encryption key, which is distinct from its original snapshot.

`zfs set keysource=`*value filesystem|volume*| ...
Sets the `keysource` property to the given value for each dataset. You can only change the `keysource` location. If you want to change the wrapping key value, use the `zfs key -c` command.

`-r`
Recursively apply the effective value of the setting throughout the subtree of child datasets. The effective value may be set or inherited, depending on the property.

`zfs get encryption | keysource | keystatus | rekeydate` *filesystem|volume*| ...
Displays properties for the given datasets.

`-r`
Recursively display properties for the given datasets.

`-d` *depth*
Recursively display any descendent datasets, limiting the recursion to *depth*. A depth of `1` will display only the dataset and its direct children.

`-H`
Display output in a form more easily parsed by scripts. Any headers are omitted, and fields are explicitly separated by a single tab instead of an arbitrary amount of space.

`zfs key -l | {-a | [-r] filesystem|volume}`
Loads the encryption key for a dataset and any datasets that inherit the key. The key that is provided with this command is not the actual key that is used to encrypt the dataset. It is a wrapping key for the set of data encryption keys for the dataset.

`-l`
Loads the wrapping key to unlock the encrypted dataset and datasets that inherit the key. This command loads the key based on what is defined by the dataset's `keysource` property.

During a pool import, a key load operation is performed when a dataset is mounted. During boot, if the wrapping key is available and the `keysource` is not set to `prompt`, the key load operation is performed.

`-a`
Apply to all datasets in all pools on the system.

`-r`
Apply the operation recursively to all datasets below the named file system or volume.

`zfs key -u [-f] | {-a | [-r] filesystem|volume}`
Unloads the encryption key for a dataset and any datasets that inherit the key.

`-u`
Unmounts the dataset and then attempts to unload the wrapping key for an encrypted dataset and datasets that inherit the key. If successful, the dataset is not accessible and is unmounted.

`-f`
Attempts to force unmount the dataset before attempting to unload the key. If not specified, a normal unmount is attempted.

`-a`
Apply to all datasets in all pools on the system.

`-r`
Apply the operation recursively to all datasets below the named file system or volume.

`zfs key -c [-o keysource=value] | {-a | [-r] filesystem|volume}`
Changes the wrapping key. If the new key has a different format or locator, the `keysource` property must be included as part of the command. Only the `keysource` property can be changed as part of the `zfs key -c` command.

`-c`
Changes the wrapping key for the key of an encrypted dataset and the datasets that inherit it. The existing key must already have been loaded before the key change operation can occur. ZFS does not prompt you for the existing passphrase.

`-o` *property=value*
Property to be changed as part of the key change operation. The `keysource` property is the only option that can be changed as part of a key change operation.

You must have permission to change the `keysource` properties.

`-a`
Apply to all datasets in all pools on the system.

`-r`
Apply the operation recursively to all datasets below the named file system or volume.

`zfs key -K` {`-a` | [`-r`] *filesystem*|*volume*}
Creates a new data encryption key. The new data encryption key is wrapped by the same wrapping key as any existing data encryption keys for this dataset.

`-K`
Creates a new data encryption key for this dataset. Data written after this operation will use the new data encryption key.

`-a`
Apply to all datasets in all pools on the system.

`-r`
Apply the operation recursively to all datasets below the named file system or volume.

`zfs mount`
`zfs mount` [`-vO`] [`-o` *options*] `-a` | *filesystem*
Mounts ZFS file systems. Invoked automatically as part of the boot process. For a full description of `zfs mount` syntax, see `zfs`(1M).

filesystem
Mount the specified filesystem.

A `zfs mount` operation of an encrypted dataset might prompt you for a key, depending on the `keysource` property value. This might occur, for example, if the `keysource` locator is set to `prompt`.

`zfs unmount` [`-f`] `-a` | *filesystem*|*mountpoint*
Unmounts currently mounted ZFS file systems. Invoked automatically as part of the shutdown process. For a full description of `zfs unmount` syntax, see `zfs`(1M).

filesystem|*mountpoint*
Unmount the specified filesystem. The command can also be given a path to a ZFS file system mount point on the system.

For an encrypted dataset, the key is not unloaded when the file system is unmounted. To unload the key, see `zfs key`.

Examples

EXAMPLE 1 Creating an Encrypted Dataset

The following example shows how to create an encrypted dataset by using a `passphrase` prompt, which is the default value of the `keysource` property. This example assumes that the `tank/home` dataset is not encrypted.

```
# zfs create -o encryption=on tank/home/bob
Enter passphrase for 'tank/home/bob/': **********
Enter again: **********
```

In the following example, the `pktool(1)` command is used to generate a raw key to a file. Next, an encrypted dataset (`tank/home/anne`) is created with the `aes-256-ccm` algorithm and the raw key file that was generated by `pktool`.

```
# pktool genkey keystore=file outkey=/media/stick/mykey \
keytype=aes keylen=256
# zfs create -o encryption=aes-256-ccm \
-o keysource=raw,file:///rmdisk/stick/mykey tank/home/anne
```

This example shows how to create an encrypted ZFS file system that retrieves the passphrase that is stored at an `https` location.

```
# zfs create -o encryption=on \
-o keysource=passphrase,https://keys.example.com/keys/42 tank/home/fs1
```

This example shows how to generate a raw key in a PKCS#11 token. Then, an encrypted dataset is created with the raw PKCS#11 key that was generated from `pktool`.

```
# pktool genkey keystore=pkcs11 keytype=aes keylen=128 label=fs2
Enter PIN for Sun Software PKCS#11 softtoken: xxxxx
# zfs create -o encryption=on -o keysource=raw,pkcs11:object=fs2 \
tank/home/fs2
Enter PKCS#11 token PIN for 'tank/home/fs2': xxxxx
```

This example shows how to generate a raw key in a KMS token. Then, an encrypted dataset is created with the raw KMS key that was generated from `pktool`.

```
# pktool genkey keystore=pkcs11 keytype=aes keylen=256 token=KMS \
label=fs3
Enter PIN for KMS: xxxxx
# zfs create -o encryption=aes-256-ccm \
-o keysource="raw,pkcs11:token=KMS;object=fs3" tank/home/fs3
Enter 'KMS' PKCS#11 token PIN for 'tank/home/fs3': xxxxx
```

EXAMPLE 2 Creating an Encrypted Dataset with a Different Encryption Algorithm

In this example, any `tank/home` datasets inherit the `keysource` properties, but the `tank/home/bob` dataset is created by using a different encryption algorithm.

```
# zpool create tank ....
# zfs create -o encryption=on tank/home
# zfs get keystatus tank/home
```

EXAMPLE 2 Creating an Encrypted Dataset with a Different Encryption Algorithm *(Continued)*

```
NAME       PROPERTY   VALUE        SOURCE
tank/home  keystatus  available    -

# zfs create -o encryption=aes-256-ccm tank/home/bob
```

EXAMPLE 3 Inheriting Encryption and Keysource Properties

In this example, all of the `tank/home` datasets inherit the `encryption` and `keysource` properties.

```
# zpool create -O encryption=on -o keysource=raw,file:///... tank ...
# zfs create tank/home
```

EXAMPLE 4 Changing an Encrypted Dataset's Wrapping Key and Keysource

This example shows how to change a dataset's wrapping key to a new key defined by the dataset's `keysource` property.

```
# zfs create -o encryption=aes-256-ccm -o keysource=raw,file:///etc/keyfile \
tank/home/rory
# zfs get keysource tank/home/rory
NAME            PROPERTY   VALUE                   SOURCE
tank/home/rory  keysource  raw,file:///etc/keyfile  local
# zfs key -c -o keysource=passphrase,prompt  tank/home/rory
Enter passphrase for 'tank/home/rory/': **********
Enter again: **********
# zfs get keychangedate tank/home/rory
NAME            PROPERTY       VALUE                  SOURCE
tank/home/rory  keychangedate  Thu Jun 28 14:32 2012  local
```

The following example shows how to change the `http` location of dataset's wrapping key.

```
# zfs get keysource tank/home/bob
NAME           PROPERTY   VALUE              SOURCE
tank/home/bob  keysource  passphrase,prompt  local

# zfs set keysource=passphrase,https://internal.example.com/keys/bob/zfs \
tank/home/bob
```

You must have the delegated `key` and `keychange` permissions to change the `keysource` property.

EXAMPLE 5 Rekeying the Dataset's Encryption Key

This example shows how to change a dataset's encryption key, which is neither visible nor managed by you or an administrator. The dataset's encryption key is wrapped (encrypted) by the key specified in the `keysource` property.

EXAMPLE 5 Rekeying the Dataset's Encryption Key *(Continued)*

```
# zfs key -K tank/project42
# zfs get rekeydate,creation tank/project42
```

You must have the delegated `keychange` permission to perform a key change operation.

EXAMPLE 6 Customizing the Encrypted Dataset's Wrapping Key

The following examples illustrate that the wrapping key size does not have to match the key size specified for the `encryption` property.

```
# zfs create -o encryption=aes-128-gcm -o keysource=raw,file:///k256 \
/tank/home/amy
```

In the above example, the encryption key size is 128 and the wrapping key size is 256.

```
# zfs create -o encryption=aes-256-gcm -o keysource=raw,file:///k192 \
/tank/home/rose
```

In the above example, the encryption key size is 256 and the wrapping key size is 192.

Exit Status

The following exit values are returned:

`0`
: Successful completion.

`1`
: An error occurred.

`2`
: Invalid command line options were specified.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/file-system/zfs
Interface Stability	Committed

See Also

`chmod(1)`, `chown(1)`, `pktool(1)`, `ssh(1)`, `mount(1M)`, `zfs(1M)`, `zpool(1M)`, `chmod(2)`, `chown(2)`, `stat(2)`, `write(2)`, `attributes(5)`

For information about using other ZFS features, see `zfs_allow(1M)`, `zfs_share(1M)`, `zfs(1M)`, and the *Oracle Solaris 11.1 Administration: ZFS File Systems*.

Name zfs_share – share and unshare a ZFS file system

Synopsis

```
zfs help subcommand | help | property property-name | permission

zfs help -l properties

zfs destroy share

zfs get [-rHpe|-d max][-o all | field[,...]] [-s source[,...]]
     all | property[,...] filesystem|volume|snapshot|share ...

zfs get share [filesystem]

zfs [-r] set [-r | -c] filesystem|volume|snapshot|share ...

zfs [-r] set share.nfs=on | off filesystem

zfs [-r] set share.smb=on | off filesystem

zfs share -u [-o property=value]... filesystem%share

zfs share filesystem|mountpoint|filesystem%share

zfs share -a | -r filesystem

zfs unshare filesystem|mountpoint|filesystem%share

zfs unshare -a | -r filesystem
```

Description You can create an NFS share or an SMB share of a ZFS file system by setting `share.nfs` or `share.smb` property. You can also use the `zfs share` and `zfs unshare` commands to publish or unpublish a ZFS share.

Sharing ZFS File Systems

A file system can be shared by setting or inheriting the `share.nfs=on` or `share.smb=on` property value. For example:

```
# zfs set share.nfs=on tank/home
# zfs set share.smb=on tank/data
```

The above simple syntax creates and publishes the file system shares automatically. This method is referred to as an *automatic* share. For more information, see the EXAMPLES section.

The automatic share is read-only and inherits all of its properties from the parent file system. This method allows sharing to be enabled by inheritance alone, if needed, without having to create a share for each descendent file system. The published share name, `share.name`, of an automatic share is generated from the dataset mount point.

For example, the *share.name* of `tank/home` is `tank_home`.

A file system's automatic share name displays as `filesystem%`. For example, `tank/home%`.

You can also create and publish a share by using the `zfs share` command as follows:

```
# zfs share -o share.smb=on sandbox/myfs%myshare
```

The above syntax creates and publishes a *named* share, which provides more flexibility when you need to share subdirectories within a file system over NFS or SMB protocols. For more information, see the EXAMPLES section.

The `listshares` pool property is used to determine whether share information is displayed when using the `zfs list` command. For more information, see `zpool(1M)`.

Native Share File System Properties

File system properties are divided into two types, native properties and user-defined (or *user*) properties. Native properties either display information or control ZFS behavior. In addition, native properties are either editable or read-only.

Properties are inherited from the parent unless overridden by the child. Some properties apply only to certain types of datasets (file systems, volumes, or snapshots).

The following native properties can be used to change the behavior of a ZFS file system and are generally used when a file system is shared.

`nbmand=on | off`
Controls whether the file system should be mounted with `nbmand` (Non Blocking mandatory locks). This is used for SMB clients. Changes to this property only take effect when the file system is unmounted and remounted. See `mount(1M)` for more information on `nbmand` mounts.

`readonly=on | off`
Controls whether this dataset can be modified. The default value is `off`.

This property can also be referred to by its shortened column name, `rdonly`.

`share.nfs=on | off`
Controls whether a ZFS dataset is created and published as an NFS share. You can also publish and unpublish an named NFS share of a ZFS dataset by using the `zfs share` and `zfs unshare` commands. Both methods of publishing an NFS share require that the NFS share properties are already set. For information about setting NFS share properties, see the `zfs set` command syntax below.

When the `share.nfs` property is changed, the file system share and any children inheriting the property are re-published with any new options that have been set with the `zfs set` command only if the property was previously `off`, or if the shares were published before the property was changed. If the new property value is `off`, the file system shares are unpublished.

`share.smb=on | off`
Controls whether a ZFS dataset is created and published as an SMB share. You can also publish and unpublish an named SMB share of a ZFS dataset by using the `zfs share` and `zfs unshare` commands. Both methods of publishing an SMB share require that the SMB share properties are also set. For information about setting SMB share properties, see the `zfs set` command syntax below.

When SMB shares are created, the SMB share name appears as an entry in the `.zfs/shares` directory. You can use the `ls` or `chmod` command to display the share-level ACLs on the entries in this directory.

When the property is changed from off to on, any shares that inherit the property are re-shared with their current options. When the property is set to off, the shares that inherit the property are unshared.

`vscan=on | off`
Controls whether regular files should be scanned for viruses when a file is opened and closed. In addition to enabling this property, the virus scan service must also be enabled for virus scanning to occur. The default value is `off`.

The following properties cannot be changed after the file system is created and, therefore, should be set when the file system is created. If the properties are not set with the `zfs create` or `zpool create` commands, these properties are inherited from the parent dataset. If the parent dataset lacks these properties due to having been created prior to these features being supported, the new file system will have the default values for these properties.

`casesensitivity=sensitive | insensitive | mixed`
Indicates whether the file name matching algorithm used by the file system should be case-sensitive, case-insensitive, or allow a combination of both styles of matching. The default value for the `casesensitivity` property is `mixed`. Traditionally, UNIX and POSIX file systems have case-sensitive file names.

The `mixed` value for the `casesensitivity` property indicates that the file system can support requests for both case-sensitive and case-insensitive matching behavior. Currently, case-insensitive matching behavior on a file system that supports mixed behavior is limited to the Solaris SMB server product. For more information about the `mixed` value behavior, see the *Oracle Solaris 11.1 Administration: ZFS File Systems*.

`normalization = none | formC | formD | formKC | formKD`
Indicates whether the file system should perform a `unicode` normalization of file names whenever two file names are compared, and which normalization algorithm should be used. File names are always stored unmodified, names are normalized as part of any comparison process. If this property is set to a legal value other than `none`, and the `utf8only` property was left unspecified, the `utf8only` property is automatically set to `on`. The default value of the `normalization` property is `none`. This property cannot be changed after the file system is created.

`utf8only=on | off`
Indicates whether the file system should reject file names that include characters that are not present in the UTF-8 character code set. If this property is explicitly set to `off`, the normalization property must either not be explicitly set or be set to `none`. The default value for the `utf8only` property is `off`. This property cannot be changed after the file system is created.

Specific NFS or SMB Properties

In addition to native properties and user properties, you can also designate properties that control the way a file system is shared. The following set of share-related properties fall into 3 categories: global properties that apply to both NFS and SMB sharing, NFS-specific properties, and SMB-specific properties.

Global share properties are mostly read-only with a few exceptions. The following global share properties apply to either a NFS or SMB share or on the shared or to be shared file system:

TABLE 1 Global Share Property Descriptions

Property	Description	Inheritable	Value
`share.desc`	Editable property that provides a user-defined description and can be set on the file system or a share. The default value is no description.	Yes	*string*
`share.fs`	Read-only property that identifies the file system name for a share.	No	*filesystem*
`share.name`	Read-only property that identifies the share name for a share.	No	*share-name*
`share.noauto`	Editable property that disables automatic sharing and can only be set on the file system to be shared only.	No	on or off
`share.path`	Editable property that sets the share path on a share.	No	*mountpoint-relative-path*
`share.point`	Read-only property that identifies the absolute path of an existing share that is derived from the current value of the `share.path` property relative to the dataset mount point.	No	*path*
`share.protocols`	Read-only property that identifies the protocols established for the file system or the share.	No	*protocol-list*
`share.state`	Read-only property that identifies the current state of the share.	No	unshared, shared, or failed

The following share properties are specific to the NFS protocol. All NFS share specific properties are editable and inheritable. The default value for most of these properties is off unless stated otherwise.

The following are the NFS share property descriptions.

`share.nfs`
Determines whether a file system is shared over the NFS protocol. Value: `on` or `off`

`share.nfs.aclok`
Determines NFSv2 client access control so that when this property is set on the server, maximum access is given to all clients. If this property is not set, minimum access is given to all clients. Value: `on` or `off`

`share.nfs.aclfab`
Determines whether ACL permissions are fabricated. Value: `on` or `off`

`share.nfs.anon`
Sets UID to the effective user ID of unknown users. By default, unknown users are given the effective UID `nobody`. If UID is set to -1, access is denied. Value: *uid*

`share.nfs.charset.euc-cn`
Sets NFS character encoding to `euc-cn` (Chinese). Value: *access-list*

`share.nfs.charset.euc-jpms`
Sets NFS character encoding to `euc-jpms` (MicroSoft-compatible Japanese). Value: *access-list*

`share.nfs.charset.euc-kr`
Sets NFS character encoding to `euc-kr` (Korean). Value: *access-list*

`share.nfs.charset.euc-tw`
Sets NFS character encoding to `euc-tw` (Taiwanese). Value: *access-list*

`share.nfs.charset.iso8859-1`
Sets NFS character encoding to ISO 8859–1 (Latin 1). Value: *access-list*

`share.nfs.charset.iso8859-2`
Sets NFS character encoding to ISO 8859–2 (Latin 2). Value: *access-list*

`share.nfs.charset.iso8859-5`
Sets NFS character encoding to ISO 8859–5 (Latin/Cyrillic). Value: *access-list*

`share.nfs.charset.iso8859-6`
Sets NFS character encoding to ISO 8859–6 (Arabic). Value: *access-list*

`share.nfs.charset.iso8859-7`
Sets NFS character encoding to ISO 8859–7 (Greek). Value: *access-list*

`share.nfs.charset.iso8859-8`
Sets NFS character encoding to ISO 8859–8 (Hebrew). Value: *access-list*

`share.nfs.charset.iso8859-9`
Sets NFS character encoding to ISO 8859–9 (Turkish). Value: *access-list*

`share.nfs.charset.iso8859-13`
Sets NFS character encoding to ISO 8859–13 (Baltic). Value: *access-list*

`share.nfs.charset.iso8859-15`
Sets NFS character encoding to ISO 8859–15 (Western European). Value: *access-list*

`share.nfs.charset.koi8-r`
Sets NFS character encoding to ISO KOI8–R (Russian/Cyrillic). Value: *access-list*

`share.nfs.cksum`
Not yet implemented. Value: *string*

`share.nfs.index`
Determines whether a file is loaded rather than a directory listing that contains this file when the directory is referenced by an NFS URL. Value: *filename*

`share.nfs.log`
Enables NFSv2 or NFSv3 server logging for the specified file system. The tag is defined in the `/etc/nfs/nfslog.conf` file. If no tag is specified, the default values associated with the global tag in the `/etc/nfs/nfslog.conf` file is used. Value: *tag*

`share.nfs.nosub`
Prevents NFSv2 or NFSv3 clients from mounting subdirectories of shared directories. Value: `on` or `off`

`share.nfs.nosuid`
Prevents the NFS client from creating files with `setuid` or `setguid` permissions. If enabled, the NFS server silently ignores any attempt to enable the `setuid` or `setgid` permissions. Value: `on` or `off`

`share.nfs.public`
Changes the location of the public file handle from root to the shared directory for NFS-enabled browsers and clients. Value: `on` or `off`

`share.nfs.sec`
Sets the default security mode to `SYS`. The `SYS` security mode uses `AUTH_SYS` authentication, which means the user's UID and GID are passed in clear text on the network, unauthenticated by the NFS server. Value: *security-mode-list*

`share.nfs.sec.default.none`
Sets the default security mode to none for *access-list*. Value: *access-list*

`share.nfs.sec.default.ro`
Sets the default security mode to read-only access for *access-list*. Value: *access-list*

`share.nfs.sec.default.root`
Sets the default security mode to root access for *access-list*. By default, no system has root access. Value: *access-list*

`share.nfs.sec.default.root_mapping`
Sets the default security mode to root access to a specific *UID*. By default, no user has root access. Value: *UID*

`share.nfs.sec.default.rw`
Sets the default security mode to read-write access for *access-list*. Value: *access-list*

`share.nfs.sec.default.window`
Sets a maximum life time in seconds for the requestor's credential that the NFS server allows for the default security mode. The default value is 30000 seconds (8.3 hours). Value: *seconds*

`share.nfs.sec.dh.none`
Sets the Diffie Helman (dh) security mode to none for *access-list*. Value: *access-list*

`share.nfs.sec.dh.ro`
Sets the dh security mode to read-only access for *access-list.* Value: *access-list.*

`share.nfs.sec.dh.root`
Sets the dh security mode to root access for *access-list*. By default, no system has root access. Value: *access-list.*

`share.nfs.sec.dh.root_mapping`
Sets the dh security mode to root access to a specific *UID*. By default, no user has root access. Value: *UID*

`share.nfs.sec.dh.rw`
Sets the default security mode to read-write access for *access-list*. Value: *access-list*

`share.nfs.sec.dh.window`
Sets a maximum life time in seconds for the requestor's credential that the NFS server allows for the dh security mode. The default value is 30000 seconds (8.3 hours). Value: *seconds*

`share.nfs.sec.krb5.none`
Sets the Kerberos V5 (krb5) security mode to none for *access-list*. Value: *access-list*

`share.nfs.sec.krb5.ro`
Sets the krb5 security mode to read-only access for *access-list*. Value: *access-list*

`share.nfs.sec.krb5.root`
Sets the krb5 security mode to root access for *access-list*. By default, no system has root access. Value: *access-list*

`share.nfs.sec.krb5.root_mapping`
Sets the krb5 security mode to root access to a specific *UID*. By default, no user has root access. Value: *UID*

`share.nfs.sec.krb5.rw`
Sets the krb5 security mode to read-write access for *access-list*. Value: *access-list*

`share.nfs.sec.krb5.window`
This property is not implemented for the krb5 security mode. Value: N/A

`share.nfs.sec.krb5i.none`
Sets the Kerberos V5 (krb5i) security mode to none. Value: *access-list*

`share.nfs.sec.krb5i.ro`
Sets the krb5i security mode to read-only access for *access-list*. Value: *access-list*

`share.nfs.sec.krb5i.root`
Sets the krb5i security mode to root access for *access-list*. By default, no system has root access. Value: *access-list*

`share.nfs.sec.krb5i.root_mapping`
Sets the krb5i security mode to root access to a specific *UID*. By default, no user has root access. Value: *UID*

`share.nfs.sec.krb5i.rw`
Sets the krb5i security mode to read-write access for *access-list*. Value: *access-list*

`share.nfs.sec.krb5i.window`
This property is not available for the krb5i security mode. Value: N/A

`share.nfs.sec.krb5p.none`
Sets the Kerberos V5 (krb5i) security mode to none for *access-list*. Value: *access-list*

`share.nfs.sec.krb5p.ro`
Sets the krb5p security mode to read-only access for *access-list*. Value: *access-list*

`share.nfs.sec.krb5p.root`
Sets the krb5p security mode to root access for *access-list*. By default, no system has root access. Value: *access-list*

`share.nfs.sec.krb5p.root_mapping`
Sets the krb5p security mode to root access to a specific *UID*. By default, no user has root access. Value: *UID*

`share.nfs.sec.krb5p.rw`
Sets the krb5i security mode to read-write access for *access-list*. Value: *access-list*

`share.nfs.sec.krb5p.window`
This property is not implemented for the krb5p security mode. Value: N/A

`share.nfs.sec.none.none`
Sets the security mode to none for *access-list*. Value: *access-list*

`share.nfs.sec.none.ro`
Sets the security mode to read-only access for *access-list*. Value: *access-list*.

`share.nfs.sec.none.root`
Sets the security mode to root access for *access-list*. By default, no system has root access. Value: *access-list*

`share.nfs.sec.none.root_mapping`
Sets the security mode to root access to a specific *UID*. By default, no user has root access. Value: *UID*

`share.nfs.sec.none.rw`
Sets the security mode to read-write access for *access-list*. Value: *access-list*

`share.nfs.sec.none.window`
This property is not implemented. Value: *seconds*

`share.nfs.sec.sys.none`
Sets the `SYS` security mode to none for *access-list*. Value: *access-list*

`share.nfs.sec.sys.ro`
Sets the SYS security mode to read-only access for *access-list*. Value: *access-list*.

`share.nfs.sec.sys.root`
Sets the SYS security mode to root access for *access-list*. By default, no system has root access. Value: *access-list*

`share.nfs.sec.sys.root_mapping`
Sets the security mode to root access to a specific *UID*. By default, no user has root access. Value: *UID*

`share.nfs.sec.sys.rw`
Sets the security mode to read-write access for *access-list*. Value: *access-list*

`share.nfs.sec.sys.window`
This property is not implemented for the `SYS` security mode. Value: *seconds*

The following share properties are specific to the SMB protocol. All SMB share specific properties are editable and inheritable.

TABLE 2 SMB Share Property Descriptions

Property	Description	Value
`share.smb`	Determines whether a file system is shared over the SMB protocol. The default value it `off`.	`on` or `off`
`share.smb.ad-container`	Enables SMB share to be published in an AD container. The default value is `off`.	*string*
`share.smb.abe`	Enables Access-Based Enumeration (abe) support. The default is `off`.	*share-name*
`share.smb.csc`	Enables client side caching support. The default value is disabled.	`disabled`, `manual`, `auto`, or `vdo`
`share.smb.catia`	Enables CATIA translation support. The default value is `off`.	`on` or `off`
`share.smb.dfsroot`	Enables a DFS root support. The default value is `off`.	`on` or `off`
`share.smb.guestok`	Enables guest access. The default value is `off`.	`on` or `off`

TABLE 2 SMB Share Property Descriptions *(Continued)*

Property	Description	Value
`share.smb.ro`	Sets the SMB share to read-only. You can specify `on`, `off`, or list of names (*access-list*). The default value is `off`.	*access-list*
`share.smb.rw`	Sets the SMB share to read-write. You can specify `on`, `off`, or list of names (*access-list*). The default value is `on`.	*access-list*
`share.smb.none`	Sets the SMB share to `off` for the specified users in the *access-list*.	*access-list*

Subcommands

All subcommands that modify state are logged persistently to the pool in their original form.

`zfs ?`
Displays a help message.

`zfs help` *command* | help | *property property-name* | *permission*
Displays `zfs` command usage information. You can display help for a specific command, property, or delegated permission. If you display help for a specific command or property, the command syntax or property value is displayed. Using `zfs help` without any arguments displays a complete list of `zfs` commands.

`zfs help -l properties`
Displays `zfs` property information, including whether the property value is editable and inheritable, and their possible values.

`zfs create` [`-p`] [`-o share.nfs=on | share.smb=on -o` ... *filesystem*
Creates a new ZFS file system. The file system is automatically mounted according to the `mountpoint` property inherited from the parent.

`-o` *property=value*
Sets the specified property as if the command `zfs set` *property=value* was invoked at the same time the dataset was created. Any editable ZFS property can also be set at creation time. Multiple `-o` options can be specified. An error results if the same property is specified in multiple `-o` options.

`zfs destroy` [*share*
The specified file system share is destroyed.

`zfs get` [`-r`|`-d` *depth*] [`-Hp`] [`-o all` | *field*[,...] [`-s` *source*[,...]] `all` | *property*[,...] *dataset*| *dataset%namedshare* ...
Displays properties for the given datasets. If no datasets are specified, then the command displays properties for all datasets on the system. For each property, the following columns are displayed:

```
name      Dataset name
property  Property name
value     Property value
```

```
source    Property source. Can either be local, default,
          temporary, inherited, or none (-).
```

All columns except the `RECEIVED` column are displayed by default; specify particular or all columns, using the `-o` option. This command takes a comma-separated list of properties as described in the "Native Properties" and "User Properties" sections.

The special value `all` can be used to display all properties that apply to the given dataset's type (filesystem, volume, or snapshot).

`-r`
Recursively display properties for any children.

`-d` *depth*
Recursively display any children of the dataset, limiting the recursion to *depth*. A depth of `1` will display only the dataset and its direct children.

`-H`
Display output in a form more easily parsed by scripts. Any headers are omitted, and fields are explicitly separated by a single tab instead of an arbitrary amount of space.

`-o` *field*
Set of fields to display. One or more of:

```
name,property,value,received,source
```

Present multiple fields as a comma-separated list. The default value is:

```
name,property,value,source
```

The keyword `all` specifies all sources.

`-s` *source*
A comma-separated list of sources to display. Those properties coming from a source other than those in this list are ignored. Each source must be one of the following:

```
local,default,inherited,temporary,received,none
```

The default value is all sources.

`-p`
Display numbers in parseable (exact) values.

`zfs get share` [*filesystem*]
Displays all defined shares or the defined shares for a specified file system.

`zfs set share.nfs=`on | `share.smb=on` [`desc=`*description*], *filesystem* | *filesystem%share*
Defines an NFS or SMB file sharing properties for a ZFS dataset by setting the `share.nfs` or `share.smb` property to `on`.

`zfs set` [`-r`] *property=value filesystem*|*volume*|*snapshot*|*share* ...
Sets the property to the given value for each file system or file system share. Only some properties can be edited. See the section for more information on what properties can be

set and acceptable values. For more information, see NFS Share Property Descriptions section or the SMB Share Property Descriptions section.

`-r`
Recursively applies the effective value of the setting throughout the subtree of child datasets. The effective value may be set or inherited, depending on the property.

`zfs share` [`-u`] `-o` *property=value ... filesystem%share*
`zfs share` *filesystem*|*mountpoint*|*filesystem%share*
`zfs share -a -r` | *filesystem*
Creates and publishes an NFS or SMB share of a ZFS dataset according to the share properties values.

Sharing a dataset with the NFS or SMB protocol means that the dataset data is available over the network. ZFS datasets that have the `share.nfs` or `share.smb` property set are automatically shared when a system is booted.

`-u`
Creates a share without immediately sharing it.

`-o` *property=value*
Shares the ZFS file system with the specified share property value.

`-a`
Shares all ZFS file systems according to their share property values and to the settings of the `share.nfs` and `share.smb` properties.

`-r`
Applies the share operation recursively to all file systems and shares below the specified file system.

filesystem | *filesystem%share*
Shares the specified file system or named file system share.

`zfs unshare` *filesystem*|*mountpoint*|*filesystem%share*
`zfs unshare -a` | `-r` |*filesystem*
Unshares all ZFS datasets that have the `share.nfs` or `share.smb` property set.

`-a`
Unshare all shared ZFS file systems. Invoked automatically as part of the boot process.

`-r`
Applies the unshare operation recursively to all file systems and shares below the specified file system

filesystem|*mountpoint*|*filesystem%share*
Unshare the specified file system. The command can also be given a path to a ZFS file system shared on the system.

Examples

EXAMPLE 1 Creating an NFS or SMB Share of a ZFS File System

The following examples show how to share ZFS file systems in different ways.

A ZFS file system can be shared when it is created. For example:

```
# zfs create -o share.nfs=on tank/workspace
```

You can also apply a share property.

```
# zfs set share.nfs.nosuid=on tank/workspace
```

Confirm that the file system is shared. For example:

```
# grep workspace /etc/dfs/sharetab
/tank/workspace tank_workspace  nfs     nosuid,sec=sys,rw
```

A descendent file system is automatically shared. For example:

```
# zfs create tank/workspace/fs1
```

A file system can be shared after it is created. For example:

```
# zfs set share.smb=on tank/data
```

EXAMPLE 2 Creating a More Complex SMB Share of ZFS File System

The following example shows how to create an SMB that uses ABE (access-enabled enumeration) to determine which users can see files for which they have access. A share called `%shareabe` is created with the `share.smb.abe` property set to on. A new share called `%sharenoabe` is created on the same file system with `share.smb.abe` set to off.

```
# zfs share -o share.smb=on -o share.smb.abe=on tank/users/files%shareabe
# zfs share -o share.smb=off tank/users/files%sharenoabe
```

To allow specific users to see all files in the `sharenoabe` share and other users to see only files for which they have access in the `shareabe` share, you would need to modify the share permissions that are accessible in `/tank/users/files/.zfs/shares` directory.

EXAMPLE 3 Adding or Changing Share Properties on a ZFS File System

You can share a file system over both NFS and SMB protocols. For example:

```
# zfs set share.nfs=on tank/data
# zfs set share.smb=on tank/data
```

You can further add or change share properties. For example:

```
# zfs set share.nfs.ro=on tank/data
```

EXAMPLE 4 Displaying NFS or SMB Share Information

Display NFS or SMB share information.

EXAMPLE 4 Displaying NFS or SMB Share Information *(Continued)*

Confirm that descendent file systems are shared when the parent file system is shared. For example:

```
# zfs get -r share.nfs tank/workspace
NAME                 PROPERTY   VALUE  SOURCE
tank/workspace       share.nfs  on     local
tank/workspace%      share.nfs  on     inherited from tank/workspace
tank/workspace/fs1   share.nfs  on     inherited from tank/workspace
tank/workspace/fs1%  share.nfs  on     inherited from tank/workspace
```

Exit Status

The following exit values are returned:

`0`
Successful completion.

`1`
An error occurred.

`2`
Invalid command line options were specified.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/file-system/zfs
Interface Stability	Committed

See Also

`share(1M)`, `share_nfs(1M)`, `share_smb(1M)`, `unshare(1M)`, `zfs(1M)`, `zpool(1M)`, `chmod(2)`, `chown(2)`, `stat(2)`, `write(2)`, `fsync(3C)`, `dfstab(4)`, `vfstab(4)`, `attributes(5)`

For information about using other ZFS features, see `zfs_allow(1M)`, `zfs_encrypt(1M)`, and `zfs(1M)`, and the *Oracle Solaris 11.1 Administration: ZFS File Systems*.

Name zic – time zone compiler

Synopsis

```
zic [--version] [-s] [-v] [-d directory | -l localtime]
    [-p posixrules] [-y yearistype] [filename]...
```

Description `zic` reads text from the file(s) named on the command line and creates the time conversion information files specified in this input. If a *filename* is '–', the standard input is read.

Input lines are made up of fields. Fields are separated by any number of white space characters. Leading and trailing white space on input lines is ignored. A pound sign (#) indicates a comment that extends to the end of the line. White space characters and pound signs can be enclosed within double quotes (" ") if they are to be used as part of a field. Any line that is blank (after comment stripping) is ignored. Non-blank lines are expected to be of one of three types: rule lines, zone lines, or link lines.

Rule A rule line has the form:

For example:

```
Rule   NAME  FROM  TO  TYPE  IN   ON     AT   SAVE  LETTER/S
```

The fields that make up a rule line are:

```
Rule   USA   1969  1973   -  Apr lastSun 2:00  1:00   D
```

`NAME` Gives the (arbitrary) name of the set of rules this rule is part of.

`FROM` Gives the first year in which the rule applies. The word `minimum` (or an abbreviation) means the minimum year with a representable time value. The word `maximum` (or an abbreviation) means the maximum year with a representable time value.

`TO` Gives the final year in which the rule applies. In addition to `minimum` and `maximum` (as above), the word `only` (or an abbreviation) can be used to repeat the value of the `FROM` field.

`TYPE` Gives the type of year in which the rule applies. If `TYPE` is:

- '–' The rule applies in all years between `FROM` and `TO`, inclusive.
- `uspres` The rule applies in U.S. Presidential election years.
- `nonpres` The rule applies in years other than U.S. Presidential election years.
- `even` The rule applies to even-numbered years.
- `odd` The rule applies to odd-numbered years.

If `TYPE` is something else, then `zic` will attempt to execute the command

`yearistype` *year* `type`

to check the type of a year: an exit status of `0` means that the year is of the given type; an exit status of `1` means that the year is not of the given type. The `yearistype` command is not currently provided in the Solaris environment.

`IN` Names the month in which the rule takes effect. Month names can be abbreviated.

`ON` Gives the day on which the rule takes effect. Recognized forms include:

`5`	the fifth day of the month
`lastSun`	The last Sunday in the month
`lastMon`	The last Monday in the month
`Sun>=8`	First Sunday on or after the eighth
`Sun<=25`	Last Sunday on or before the 25th

Names of days of the week can be abbreviated or spelled out in full. Note: There cannot be spaces within the `ON` field.

`AT` Gives the time of day at which the rule takes effect. Recognized forms include:

`2`	Time in hours
`2:00`	Time in hours and minutes
`15:00`	24-hour format time (for times after noon)
`1:28:14`	Time in hours, minutes, and seconds, where hour 0 is midnight at the start of the day and hour 24 is midnight at the end of the day.

Any of these forms can be followed by the letter `w` if the given time is local "wall clock" time; `s` if the given time is local "standard" time; or `u` (or `g` or `z`) if the given time is universal time. In the absence of an indicator, wall clock time is assumed.

`SAVE` Gives the amount of time to be added to local standard time when the rule is in effect. This field has the same format as the `AT` field (without the `w` and `s` suffixes).

`LETTER/S` Gives the "variable part" (for example, the "S" or "D" in "EST" or "EDT" of time zone abbreviations to be used when this rule is in effect. If this field is '–', the variable part is null.

`Zone` A zone line has the form:

```
Zone  NAME                GMTOFF  RULES/SAVE   FORMAT  [UNTIL]
```

For example:

```
Zone Australia/SouthWest   9:30        -         CST    1992 Mar 15 12:00
                           8:30      Aus         CST
```

The fields that make up a zone line are:

NAME — The name of the time zone. This is the name used in creating the time conversion information file for the zone.

GMTOFF — The amount of time to add to UTC to get standard time in this zone. This field has the same format as the AT and SAVE fields of rule lines; begin the field with a minus sign to subtract time from UTC.

RULES/SAVE — The name of the rule(s) that apply in the time zone or, alternately, an amount of time to add to local standard time. If this field is '–', then standard time always applies in the time zone.

FORMAT — The format for time zone abbreviations in this time zone. The pair of characters %s is used to show where the "variable part" of the time zone abbreviation goes. Alternately, a slash (/) separates standard and daylight abbreviations.

UNTIL — The time at which the UTC offset or the rule(s) change for a location. It is specified as a year, a month, a day, and a time of day. The time of day has the same format as the AT field of rule lines. If this is specified, the time zone information is generated from the given UTC offset and rule change until the time specified.

The month, day, and time of day have the same format as the IN, ON, and AT columns of a rule; trailing columns can be omitted, and default to the earliest possible value for the missing columns.

The next line must be a "continuation" line. This line has the same form as a zone line except that the string "Zone" and the name are omitted. The continuation line places information starting at the time specified as the UNTIL field in the previous line in the file used by the previous line. Continuation lines can contain an UNTIL field, just as zone lines do, indicating that the next line is a further continuation.

Link — A link line has the form:

```
Link   LINK-FROM   LINK-TO
```

For example:

```
Link   Europe/Istanbul Asia/Istanbul
```

The LINK-FROM field should appear as the NAME field in some zone line; the LINK-TO field is used as an alternate name for that zone.

Except for continuation lines, lines can appear in any order in the input.

Options

`--`*version* Outputs version information and exits.

`-d` *directory* Creates time conversion information files in the directory *directory* rather than in the standard directory `/usr/share/lib/zoneinfo`.

The `-d` and `-l` options are mutually exclusive.

`-l` *localtime* Uses the given time zone as local time *localtime*. `zic` acts as if the file contained a link line of the form:

```
Link localtime localtime
```

The `-d` and `-l` options are mutually exclusive.

`-p` *posixrules* Uses the rules of the given time zone *posixrules* when handling POSIX-format time zone environment variables. `zic` acts as if the input contained a link line of the form:

```
Link    posixrules posixrules
```

This option is not used by `ctime(3C)` and `mktime(3C)` in the Solaris environment.

`-s` Limits time values stored in output files to values that are the same whether they are taken to be signed or unsigned. You can use this option to generate SVVS-compatible files.

This option is obsolete and may be removed in a future release.

`-v` Complains if a year that appears in a data file is outside the range of years representable by system time values (`0:00:00 a.m. UTC, January 1, 1970`, to `3:14:07 a.m. UTC, January 19, 2038`). This option also complains if a time of 24:00 (which cannot be handled by pre-1998 versions of `zic`) appears in the input.

`-y` *yearistype* Uses the given command *yearistype* rather than `yearistype` when checking year types (see `Rules` under `DESCRIPTION`).

Operands

filename A file containing input lines that specify the time conversion information files to be created. If a *filename* is '–', the standard input is read.

Files

`/usr/share/lib/zoneinfo`	Standard directory used for created files
`/usr/share/lib/zoneinfo/src`	Directory containing source files

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os
Interface Stability	Committed*

* The `-s` option is obsolete.

See Also `time(1)`, `zdump(1M)`, `ctime(3C)`, `mktime(3C)`, `attributes(5)`

Notes For areas with more than two types of local time, you might need to use local standard time in the `AT` field of the earliest transition time's rule to ensure that the earliest transition time recorded in the compiled file is correct.

If the current *timezone* file is edited and compiled using the "`zic`" command, the changes will only be reflected in any new processes that are running. The most accurate way to reflect the changes for the whole system would be a reboot.

Name zoneadm – administer zones

Synopsis

```
zoneadm -z zonename [-u uuid-match] subcommand
     [subcommand_options]

zoneadm [-R root] [-z zonename] [-u uuid-match] list
     [list_options]

zoneadm [-R root] -z zonename [-u uuid-match] mark incomplete
```

Description The `zoneadm` utility is used to administer system zones. A zone is an application container that is maintained by the operating system runtime.

Security Once a process has been placed in a zone other than zone `0`, the process or any of its children cannot change zones.

Security Except for simple listing and help functions, only a user operating in the global system zone can use `zoneadm`, and it must be executed with an effective user ID of root. In addition, the user must be authorized to execute specific subcommands.

`zoneadm` checks for authorization strings that optionally include the specified *zonename* as a suffix, preceded by the slash character. When omitted, the authorization matches any zone.

Subcommands that only provide information, for example, `help` or `list`, do not require any authorizations. All other subcommands require the authorization `solaris.zone.manage/`*zonename*.

Once a process has been placed in a zone other than zone 0, neither that process nor any of its children can change zones.

Options The following options are supported:

`-R` *root*
: Specify an alternate root (boot environment). This option can only be used in conjunction with the "`list`" and "`mark`" subcommands.

`-u` *uuid-match*
: Unique identifier for a zone, as assigned by `libuuid(3LIB)`. If this option is present and the argument is a non-empty string, then the zone matching the UUID is selected instead of the one named by the `-z` option, if such a zone is present.

`-z` *zonename*
: String identifier for a zone.

Subcommands Subcommands which can result in destructive actions or loss of work have a `-F` flag to force the action. If input is from a terminal device, the user is prompted if such a command is given without the `-F` flag; otherwise, if such a command is given without the `-F` flag, the action is disallowed, with a diagnostic message written to standard error. If a zone installation or uninstallation is interrupted, the zone is left in the incomplete state. Use uninstall to reset such a zone back to the configured state.

The following subcommands are supported:

`attach` [`-u`] [`-F`] [`-x` *extended_options*] [`-n` *path*] [*brand-specific options*]
The `attach` subcommand takes a zone that has been detached from one system and attaches the zone onto a new system. Therefore, it is advised (though not required) that the `detach` subcommand should be run before the "attach" takes place. Once you have the new zone in the configured state, use the `attach` subcommand to set up the zone root instead of installing the zone as a new zone.

The `attach` subcommand is also used to transition a zone from the `unavailable` state to the `installed` state. If the `attach` subcommand is unable to perform such a transition, the zone will remain in the unavailable state.

The `-F` option can be used to force the zone into the "installed" state with no validation. This option should be used with care since it can leave the zone in an unsupportable state if it was moved from a source system to a target system that is unable to properly host the zone. The `-n` option can be used to perform a "dry run" of the `attach` subcommand. It uses the output of the "`detach -n`" subcommand as input and is useful to identify any conflicting issues, such as the network device being incompatible, and can also determine whether the host is capable of supporting the zone. The path can be "`-`", to read the input from standard input.

The zone's brand may include additional options that govern how the zone will be attached. See `brands(5)` for specific brand information.

The zone being attached must first be configured using the `zonecfg` (see `zonecfg(1M)`) command. This does not apply when running "`attach -n`".

Use the following command to attach a zone:

```
# zoneadm -z my-zone attach
```

Use the following command to attach and update a zone:

```
# zoneadm -z my-zone attach -u
```

In the absence of `-n` (as above), the source zone must be halted before this subcommand can be used.

`-n` *path*
Read the zone manifest and verify that the target machine has the correct configuration to host the zone without actually performing an attach. The zone on the target system does not have to be configured on the new host before doing a trial-run attach.

`-u`
Update the attached zone.

`-x force-zpool-import`
Specify this option to forcibly reuse existing `zpool` resources that may appear to be in in-use.

`-x force-zpool-create`
`-x force-zpool-create-all`
Specify `-x` with `force-zpool-create-all` to forcibly create all `zpool` resources.

Use `-x` with `force-zpool-create`, with the syntax:

`-x force-zpool-create=`*zpoolname*`{,`*zpoolname*`,`*zpoolname*`,...}`

...to limit this option to a specific set of `zpool` resources. To name a `rootzpool` `zpool` resource, use `rpool`. For a `zpool` resource, use the name specified in the corresponding zone config `name` property.

These options are only available for archive-based attach usage. See `brands(5)` for the information which brands support archive-based attach.

`boot` [`-w`|`-W`] [`--` *boot_options*]
Boot (or activate) the specified zones.

The `boot` subcommand has the following mutually exclusive options:

`-w`
Boots the zone with a writable root, effectively overriding the `file-mac-profile` setting in the zone's configuration. This option is in effect for this boot-cycle only: a subsequent reboot will boot the zone with a `file-mac-profile` in effect again.

`-W`
Boots the zone in transient r/w mode; when the zone completes self-assembly, the zone will reboot in read-only mode. Has no effect non-read only root zones.

The following *boot_options* are supported:

`-i` *altinit*
Select an alternative executable to be the primordial Process. *altinit* is a valid path to an executable. The default primordial process is `init(1M)`.

`-m` *smf_options*
The *smf_options* include two categories of options to control booting behavior of the service management facility: recovery options and messages options.

Message options determine the type and amount of messages that `smf(5)` displays during boot. Service options determine the services which are used to boot the system. See `kernel(1M)` for a listing of the `-m` suboptions.

`-s`
Boots only to milestone `svc:/milestone/single-user:default`. This milestone is equivalent to init level `s`. See `svc.startd(1M)` and `init(1M)`.

`clone` [`-m` *copy*] [`-s` *zfs_snapshot*] [-x *extended_options*] [*brand-specific options*] *source_zone*
Install a zone by copying an existing installed zone. This subcommand is an alternative way to install the zone.

`-m` *copy*
Force the clone to be a copy, even if a "ZFS clone" is possible.

Note, this is the default (and only supported) clone method for zones that have a `rootzpool` resource configured.

`-s` *zfs_snapshot*
Specify the name of a ZFS snapshot to use as the source of the clone. The *snapshot* must be a *snapshot* of the source zone taken from a previous "`zoneadm` clone" installation.

`-x force-zpool-import`
Specify the `-x` option with `force-zpool-import` to forcibly reuse existing `zpool` resources that may appear to be in use.

`-x force-zpool-create`
`-x force-zpool-create-all`
Specify the `-x` option with `force-zpool-create-all` to forcibly create all `zpool` resources.

Use `-x` with `force-zpool-create`, with the syntax:

`-x force-zpool-create=`*zpoolname*`{,`*zpoolname*`,`*zpoolname*`,...}`

...to limit this option to a specific set of `zpool` resources. To name a `rootzpool` `zpool` resource, use `rpool`. For a `zpool` resource, use the name specified in the corresponding zone config `name` property.

The source zone must be halted before this subcommand can be used.

`detach` [`-F` | `-n`]
Detach the specified zone. Detaching a zone is the first step in moving a zone from one system to another. The full procedure to migrate a zone is that the zone is detached, the *zonepath* directory is moved to the new host, and then the zone is attached on the new host. Once the zone is detached, it is left in the configured state. If you try to install or clone to a configured zone that has been detached, you will receive an error message and the `install` or `clone` subcommand will not be allowed to proceed. The `-n` option can be used to perform a "dry run" of the `detach` subcommand. This generates the information needed for running the "`attach -n`" subcommand, which is useful to identify any conflicting issues, such as the network device being incompatible or if the host is capable of supporting the zone. The information is sent to standard output and can be saved to a file or piped to the "`attach -n`" subcommand. The `-F` option can be used to forcefully detach the zone without performing verification checks on the existing *zonepath*.

Use the following command to detach a zone:

```
# zoneadm -z my-zone detach
```

Unless the `-n` option is used, the source zone must be halted before this subcommand can be used.

`-F`
Forcefully detach the zone without performing verification checks on the zone's storage. This option is typically used if the storage for the *zonepath* is no longer accessible to this host. Such a scenario is normally encountered when the zone's storage has been failed over to an alternate host, either manually or as part of a cluster.

`-n`
Generate a zone manifest on a running zone without actually detaching the zone. The state of the zone on the originating system is not changed. The zone manifest is sent to `stdout`. The global administrator can direct this output to a file or pipe it to a remote command to be immediately validated on the target host.

`halt`
Halt the specified zones. `halt` bypasses running the shutdown scripts inside the zone. It also removes run time resources of the zone.

See also the `shutdown` subcommand, below.

`help` [*subcommand*]
Display general help. If you specify *subcommand*, displays help on *subcommand*.

`install` [`-x` *extended_options*] [*brand-specific options*]
Install the specified zone on the system. This subcommand automatically attempts to verify first. It refuses to install if the verify step fails. See the `verify` subcommand.

`-x force-zpool-import`
Specify the `-x` option with `force-zpool-import` to forcibly reuse existing `zpool` resources that may appear to be in use.

`-x force-zpool-create`
`-x force-zpool-create-all`
Specify the `-x` option with `force-zpool-create-all` to forcibly create all `zpool` resources.

Use `-x` with `force-zpool-create`, with the syntax:

`-x force-zpool-create=`*zpoolname*`{,`*zpoolname*`,`*zpoolname*`,...}`

...to limit this option to a specific set of `zpool` resources. To name a `rootzpool` `zpool` resource, use `rpool`. For a `zpool` resource, use the name specified in the corresponding zone config `name` property.

The zone's brand may include additional options that govern how the software will be installed in the zone. See `brands(5)` for specific brand information.

`list` [*list_options*]
Display the name of the current zones, or the specified zone if indicated.

By default, all running zones are listed. If you use this subcommand with the `zoneadm -z` *zonename* option, it lists only the specified zone, regardless of its state. In this case, the `-i` and `-c` options are disallowed.

If neither the `-i` or `-c` options are given, all running zones are listed.

The following *list_options* are supported:

`-c`
Display all configured zones. This option overides the `-i` option.

`-i`
Expand the display to all installed zones.

`-p`
Request machine parsable output. The output format is a list of lines, one per zone, with colon- delimited fields. These fields are:

```
zoneid:zonename:state:zonepath:uuid:brand:ip-type:\
r/w:file-mac-profile
```

If the *zonepath* contains embedded colons, they can be escaped by a backslash ("\:"), which is parsable by using the shell `read(1)` function with the environmental variable IFS. The *uuid* value is assigned by `libuuid(3LIB)` when the zone is installed, and is useful for identifying the same zone when present (or renamed) on alternate boot environments. Any software that parses the output of the "`zoneadm list -p`" command must be able to handle any fields that may be added in the future.

The `-v` and `-p` options are mutually exclusive. If neither `-v` nor `-p` is used, just the zone name is listed.

`-v`
Display verbose information, including zone name, id, current state, root directory, brand type, ip-type, and options.

The `-v` and `-p` options are mutually exclusive. If neither `-v` nor `-p` is used, just the zone name is listed.

`mark` *state*
Change the state of a zone. Only a subset of the zone states are supported, as described below.

`incomplete`
Change the state of an installed zone to `incomplete`. This command can be useful in cases where administrative changes on the system have rendered a zone permanently unusable or inconsistent. This change cannot be undone, except by uninstalling the zone.

`unavailable`
Change the state of an installed zone to `unavailable`. This command can be useful in cases where administrative changes or failures on the system have rendered a zone temporarily unusable. This change can be undone with the attach subcommand.

`move` *new_zonepath*
Move the *zonepath* to *new_zonepath*. The zone must be halted before this subcommand can be used. The *new_zonepath* must be a local file system and normal restrictions for *zonepath* apply.

Note, for zones configured with a `rootzpool` resource only the *zonepath* will be renamed but the zone itself will not move out of its containing `zpool`.

`ready`
Prepares a zone for running applications but does not start any user processes in the zone.

`reboot` [`--` *boot_options*]
Restart the zone. This is equivalent to a `halt` `boot` sequence (shutdown scripts are not run).

See the `boot` subcommand for supported boot options.

`shutdown` [`-r` [`--` *boot_options*]]
Cleanly shut down the zone (equivalent to running `/usr/sbin/init 0` in the zone). The `shutdown` subcommand waits until the zone is successfully shut down; a `zoneadm halt` can be used to forcibly halt the zone, if the shutdown process takes a long time.

If `-r` is specified, reboot the zone. See the `boot` subcommand for supported boot options.

`uninstall [-F]` [`-x` *extended_options*]
Uninstall the specified zone from the system. Use this subcommand with caution. It removes all of the files under the *zonepath* of the zone in question. You can use the `-F` flag to force the action.

`-x force-zpool-destroy`
`-x force-zpool-destroy-all`
The `-x` option with `force-zpool-destroy-all` option can be used to destroy all zpools.

Use `-x` with `force-zpool-destroy`, with the syntax:

`-x force-zpool-destroy=`*zpoolname*`{,`*zpoolname*`,`*zpoolname*`,...}`

...to limit this option to a specific set of `zpool` resources. To name a `rootzpool` `zpool` resource, use `rpool`. For a `zpool` resource, use the name specified in the corresponding zone config `name` property.

`verify`
Check to make sure the configuration of the specified zone can safely be installed on the machine. Following is a breakdown of the checks by `resource/property` type:

zonepath
zonepath and its parent directory exist and are owned by root with appropriate modes . The appropriate modes are that *zonepath* is `700`, its parent is not `group` or `world-writable` and so forth. *zonepath* is not over an NFS mount. A sub-directory of the *zonepath* named “root” does not exist.

If *zonepath* does not exist, the `verify` does not fail, but merely warns that a subsequent install will attempt to create it with proper permissions. A `verify` subsequent to that might fail should anything go wrong.

zonepath cannot be a symbolic link.

`fs`
Any `fs` resources have their *type* value checked. An error is reported if the value is one of `proc`, `mntfs`, `autofs`, or `nfs` or the filesystem does not have an associated mount binary at `/usr/lib/fs/`<*fstype*>`/mount`.

It is an error for the *directory* to be a relative path.

It is an error for the path specified by `raw` to be a relative path or if there is no `fsck` binary for a given filesystem type at `/usr/lib/fs/`<*fstype*>`/fsck`. It is also an error if a corresponding `fsck` binary exists but a `raw` path is not specified.

`net`
All physical network interfaces exist. All network address resources are one of:

- a valid IPv4 address, optionally followed by "`/`" and a prefix length;
- a valid IPv6 address, which must be followed by "`/`" and a prefix length;
- a host name which resolves to an IPv4 address.

Note that hostnames that resolve to IPv6 addresses are not supported.

The physical interface name is the network interface name.

A zone can be configured to be either exclusive-IP or shared-IP. For a shared-IP zone, both the physical and address properties must be set. For an exclusive-IP zone, the physical property must be set and the address property cannot be set.

`anet`
It verifies that the lower-link, over which the VNIC will be automatically created, exists.

`rctl`
It also verifies that any defined resource control values are valid on the current machine. This means that the privilege level is `privileged`, the limit is lower than the currently defined system value, and that the defined action agrees with the actions that are valid for the given resource control.

`rootzpool`, `zpool`
All `zpool`s configured are online on the system for a zone in the `installed` state.

For a zone in configured state, it verifies that none of the configured `zpool` resources are already online on the system.

Examples

EXAMPLE 1 Using the -m Option

The following command illustrates the use of the -m option.

```
# zoneadm boot -- -m verbose
```

EXAMPLE 2 Using the -i Option

The following command illustrates the use of the -i option.

```
# zoneadm boot -- -i /usr/sbin/init
```

EXAMPLE 3 Using the -s Option

The following command illustrates the use of the -s option.

```
# zoneadm boot -- -s
```

Exit Status

The following exit values are returned:

0
: Successful completion.

1
: An error occurred.

2
: Invalid usage.

Attributes

See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/zones
Interface Stability	Committed

See Also

read(1), svcs(1), zlogin(1), zonename(1), init(1M), kernel(1M), svcadm(1M), zpool(1M), svc.startd(1M), svc.startd(1M), zonecfg(1M), libuuid(3LIB), attributes(5), brands(5), mwac(5), smf(5), zones(5), zfs(7FS)

Notes

The zones(5) service is managed by the service management facility, smf(5), under the service identifier:

```
svc:/system/zones:default
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using svcadm(1M). The service's status can be queried using the svcs(1) command.

For the first boot after a read-only zone is installed or upgraded, or when the zone is booted with -w/-W, the write-only protection is disabled. Care must be taken that the zone is otherwise protected.

Name zoneadmd – zone administration daemon

Synopsis

```
/usr/lib/zones/zoneadmd
```

Description `zoneadmd` is a system daemon that is started when the system needs to manage a particular zone. Because each instance of the `zoneadmd` daemon manages a particular zone, it is not unexpected to see multiple `zoneadmd` daemons running.

This daemon is started automatically by the zone management software and should not be invoked directly. The daemon shuts down automatically when no longer in use. It does not constitute a programming interface, but is classified as a private interface.

Addresses configured by `zoneadmd` on behalf of shared-ip zones are assigned automatically generated `ipadm(1M)` address object names prefixed with the string `zoneadmd`. For example, in the following output, the address 10.2.2.1/24, configured on the `net0` interface, was configured by `zoneadmd` with an automatically generated address object name of `net0/zoneadmd.v4`.

```
# ipadm show-addr
ADDROBJ           TYPE     STATE        ADDR
lo0/v4            static   ok           127.0.0.1/8
net0/v4           static   ok           10.1.2.3/24
net0/zoneadmd.v4  static   ok           10.2.2.1/24
lo0/v6            static   ok           ::1/128
```

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/zones
Interface Stability	Private

See Also `svcs(1)`, `zlogin(1)`, `ipadm(1M)`, `svcadm(1M)`, `zoneadm(1M)`, `attributes(5)`, `smf(5)`, `zones(5)`

Notes The `zones(5)` service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/system/zones:default
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. The service's status can be queried using the `svcs(1)` command.

Name zonecfg – set up zone configuration

Synopsis

```
zonecfg -z zonename
zonecfg -z zonename subcommand
zonecfg -z zonename -f command_file
zonecfg help
```

Description The `zonecfg` utility creates, modifies, and lists the configuration of a zone. The creation and modification functions are only available to authorized users and require that the process is executed with an effective user ID of root. Otherwise it runs in read-only mode.

A zone's configuration consists of a number of resources and properties.

To simplify the user interface, `zonecfg` uses the concept of a scope. The default scope is global.

The following synopsis of the `zonecfg` command is for interactive usage:

```
zonecfg -z zonename subcommand
```

Parameters changed through `zonecfg` do not affect a running zone. The zone must be rebooted for the changes to take effect.

In addition to creating and modifying a zone, the `zonecfg` utility can also be used to persistently specify the resource management settings for the global zone.

In the following text, “rctl” is used as an abbreviation for “resource control”. See `resource_controls(5)`.

Every zone is configured with an associated brand. The brand determines the user-level environment used within the zone, as well as various behaviors for the zone when it is installed, boots, or is shutdown. Once a zone has been installed the brand cannot be changed. The default brand is determined by the installed distribution in the global zone. Some brands do not support all of the `zonecfg` properties and resources. See the brand-specific man page for more details on each brand. For an overview of brands, see the `brands(5)` man page.

Resources The following resource types are supported:

`attr`
: Generic attribute.

`capped-cpu`
: Limits for CPU usage.

`capped-memory`
: Limits for physical, swap, and locked memory.

`dataset`
: ZFS dataset.

`dedicated-cpu`
Subset of the system's processors dedicated to this zone while it is running.

`device`
Device.

`fs`
file-system

`net`
Network interface.

`anet`
Automatic network interface.

`admin`
Delegated administrator.

`rctl`
Resource control.

`rootzpool`
Dedicated ZFS zpool for zone installation.

`zpool`
ZFS zpool delegated to the zone.

Sparse and Whole Root Non-Global Zones

Previous releases of Solaris offered the notion of *sparse root zones*. This functionality was intimately associated with the SVr4 packaging system and intended to save disk space and reduce administrative effort.

The new packaging system, IPS, provides more flexibility when choosing which packages to install in a zone. This, along with advances in file system technology (notable among which is ZFS *deduplication*), means that it was most sensible to remove sparse root zones. The benefits of sparse root zones are provided for all zones by means of the combination of IPS packaging and file system advances.

Properties

Each resource type has one or more properties. There are also some global properties, that is, properties of the configuration as a whole, rather than of some particular resource.

The following properties are supported:

(global)
`zonename`

(global)
`zonepath`

(global)
`autoboot`

(global)
bootargs

(global)
pool

(global)
limitpriv

(global)
brand

(global)
cpu-shares

(global)
hostid

(global)
max-lwps

(global)
max-msg-ids

(global)
max-processes

(global)
max-sem-ids

(global)
max-shm-ids

(global)
max-shm-memory

(global)
scheduling-class

(global)
fs-allowed

(global)
file-mac-profile

fs
dir, special, raw, type, options

net
address, allowed-address, configure-allowed-address, physical, defrouter

anet
linkname, lower-link, allowed-address, auto-mac-address,
configure-allowed-address, defrouter, mac-address, mac-slot, mac-prefix, mtu,

maxbw, priority, vlan-id, vsi-typeid, vsi-vers, vsi-mgrid, rxfanout, rxrings, txrings, link-protection, allowed-dhcp-cids, pkey, linkmode, etsbw-lcl, cos

device
match, allow-partition, allow-raw-io

rctl
name, value

attr
name, type, value

dataset
name, alias

dedicated-cpu
ncpus, importance

capped-memory
physical, swap, locked

capped-cpu
ncpus

admin
user, auths

rootzpool
storage

zpool
storage, name

As for the property values that are paired with these names, they are either simple, complex, or lists. The type allowed is property-specific. Simple values are strings, optionally enclosed within quotation marks. Complex values have the syntax:

(<*name*>=<*value*>,<*name*>=<*value*>,...)

where each <*value*> is simple, and the <*name*> strings are unique within a given property. Lists have the syntax:

[<*value*>,...]

where each <*value*> is either simple or complex. A list of a single value (either simple or complex) is equivalent to specifying that value without the list syntax. That is, "foo" is equivalent to "[foo]". A list can be empty (denoted by "[]").

In interpreting property values, `zonecfg` accepts regular expressions as specified in `fnmatch(5)`. See `EXAMPLES`.

The property types are described as follows:

global: `zonename`
The name of the zone.

global: `zonepath`
Path to zone's file system.

global: `autoboot`
Boolean indicating that a zone should be booted automatically at system boot. Note that if the zones service is disabled, the zone will not autoboot, regardless of the setting of this property. You enable the zones service with a `svcadm` command, such as:

```
# svcadm enable svc:/system/zones:default
```

Replace `enable` with `disable` to disable the zones service. See `svcadm(1M)`.

global: `bootargs`
Arguments (options) to be passed to the zone bootup, unless options are supplied to the "`zoneadm boot`" command, in which case those take precedence. The valid arguments are described in `zoneadm(1M)`.

global: `pool`
Name of the resource pool that this zone must be bound to when booted. This property is incompatible with the `dedicated-cpu` resource.

global: `limitpriv`
The maximum set of privileges any process in this zone can obtain. The property should consist of a comma-separated privilege set specification as described in `priv_str_to_set(3C)`. Privileges can be excluded from the resulting set by preceding their names with a dash (-) or an exclamation point (!). The special privilege string "zone" is not supported in this context. If the special string "default" occurs as the first token in the property, it expands into a safe set of privileges that preserve the resource and security isolation described in `zones(5)`. A missing or empty property is equivalent to this same set of safe privileges.

The system administrator must take extreme care when configuring privileges for a zone. Some privileges cannot be excluded through this mechanism as they are required in order to boot a zone. In addition, there are certain privileges which cannot be given to a zone as doing so would allow processes inside a zone to unduly affect processes in other zones. `zoneadm(1M)` indicates when an invalid privilege has been added or removed from a zone's privilege set when an attempt is made to either "boot" or "ready" the zone.

See `privileges(5)` for a description of privileges. The command "`ppriv -l`" (see `ppriv(1)`) produces a list of all Solaris privileges. You can specify privileges as they are displayed by `ppriv`. In `privileges(5)`, privileges are listed in the form PRIV_*privilege_name*. For example, the privilege *sys_time*, as you would specify it in this property, is listed in `privileges(5)` as `PRIV_SYS_TIME`.

global: `brand`
The zone's brand type.

global: `ip-type`
A zone can either have its own exclusive instance of IP (the default) or share the IP instance with the global zone. In the default zone template, `SYSdefault`, `ip-type` is set to `exclusive`. In the also-supplied `SYSdefault-shared-ip` template, `ip-type` is set to shared.

This property takes the values `exclusive` and `shared`.

global: `hostid`
A zone can emulate a 32-bit host identifier to ease system consolidation. A zone's `hostid` property is empty by default, meaning that the zone does not emulate a host identifier. Zone host identifiers must be hexadecimal values between 0 and FFFFFFFE. A `0x` or `0X` prefix is optional. Both uppercase and lowercase hexadecimal digits are acceptable.

global: `fs-allowed`
A comma-separated list of additional file systems that can be mounted within the zone; for example, `ufs,pcfs`. By default, only `hsfs(7FS)` and network file systems can be mounted.

This property does not apply to file systems mounted into the zone by means of `add fs` or `add dataset`.

Caution – Allowing filesystem mounts other than the default might allow the zone administrator to compromise the system with a bogus filesystem image and is not supported.

global: `file-mac-profile`
Define which parts of the filesystem are exempted from the read-only policy, that is, which parts of the filesystem the zone is allowed to write to.

There are currently four supported values for this property: `none`, `strict`, `fixed-configuration`, and `flexible-configuration`.

`none` makes the zone exactly the same as a normal, r/w zone. `strict` allows no exceptions to the read-only policy. `fixed-configuration` allows the zone to write to files in and below `/var`, except directories containing configuration files:

```
/var/ld
/var/lib/postrun
/var/pkg
/var/spool/cron,
/var/spool/postrun
/var/svc/manifest
/var/svc/profiles
```

`flexible-configuration` is equal to `fixed-configuration`, but allows writing to files in `/etc` in addition.

`fs`: dir, special, raw, type, options
Values needed to determine how, where, and so forth to mount file systems. See `mount(1M)`, `mount(2)`, `fsck(1M)`, and `vfstab(4)`.

`net`: address, allowed-address, configure-allowed-address, physical, defrouter

The `net` resource represents the assignment of a physical network resource to a zone. The resource must exists in the global zone prior to the assignment.

The network address and physical interface name of the network interface. The network address is one of:

- a valid IPv4 address, optionally followed by "`/`" and a prefix length;
- a valid IPv6 address, which must be followed by "`/`" and a prefix length;
- a host name which resolves to an IPv4 address.

Note that host names that resolve to IPv6 addresses are not supported.

The physical interface name is the network interface name.

The value for the optional default router is specified similarly to the network address except that it must not be followed by a `/` (slash) and a network prefix length. To enable correct use of the `defrouter` functionality, the zones that use the property must be on a different subnet from the subnet on which the global zone resides. Also, each zone (or set of zones) that uses a different `defrouter` setting must be on a different subnet.

A zone can be configured to be either exclusive-IP or shared-IP. For a shared-IP zone, you must set both the physical and address properties; setting the default router is optional. The interface specified in the physical property must be plumbed in the global zone prior to booting the non-global zone. However, if the interface is not used by the global zone, it should be configured `down` in the global zone, and the default router for the interface should be specified here. The `allowed-address` property cannot be set for a shared-IP zone.

For an exclusive-IP zone, the `physical` property must be set and the `address` property must not be set. Optionally, the set of IP addresses that the exclusive-IP zone can use might be constrained by specifying the `allowed-address` property. If `allowed-address` has not been specified, then the exclusive-IP zone can use any IP address on the associated `physical` interface for the net resource. Otherwise, when `allowed-address` is specified, the exclusive-IP zone cannot use IP addresses that are not in the `allowed-address` list for the `physical` address. If `configure-allowed-address` is set to `true`, the addresses specified by `allowed-address` are automatically configured on the interface each time the zone boots. When it is set to `false`, the `allowed-address` will not be configured on zone boot. By default, `configure-allowed-address` is set to `true` when an `allowed-address` is specified. In addition, when the `allowed-address` list has been populated, the `defrouter` property can also be optionally specified. However, if the `defrouter` value is specified and `configure-allowed-address` is set to `false`, the `defrouter` value will be ignored and an appropriate warning message will be shown. The interface specified for the `physical` property must not be in use in the global zone. If an `allowed-address` and default router are specified by means of `zonecfg`, these will be applied to the interface when it is enabled by means of `ipadm(1M)` in the non-global, exclusive-IP zone, typically during zone boot. The non-global exclusive-IP zone will not be able to apply any other addresses to that

interface, nor will it be able to transmit packets with a different source address for the specified IP version. A default router set up by means of `zonecfg` cannot be persistently deleted from within the non-global exclusive-IP zone using the `-p` flag with `route(1M)`.

Note that a single datalink cannot be shared among multiple exclusive-IP zones.

`anet`: linkname, lower-link, allowed-address, auto-mac-address, configure-allowed-address, defrouter, mac-address, mac-slot, mac-prefix, mtu, maxbw, priority, vlan-id, vsi-typeid, vsi-vers, vsi-mgrid, rxfanout, rxrings, txrings, link-protection, allowed-dhcp-cids, pkey, linkmode, etsbw-lcl,cos

The `anet` resource represents the automatic creation of a network resource for an exclusive-IP zone. When `zonecfg` creates a zone using the default `SYSdefault` template, an `anet` resource with the following properties is automatically included in the zone configuration:

```
linkname=net0
lower-link=auto
mac-address=random
link-protection=mac-nospoof
```

When such a zone boots, a temporary VNIC or IPoIB datalink is automatically created for the zone. The VNIC or the IPoIB datalink is deleted when the zone halts.

The supported properties are described below. All these properties are optional. Only the global zone is allowed to modify the automatically created VNIC or IPoIB datalink or its properties. If a property set in `zonecfg` cannot be assigned to the VNIC or IPoIB datalink at its creation time, the zone will fail to boot.

`linkname`

Specify a name for the automatically created VNIC or IPoIB datalink. By default, this property will be automatically set to the first available name (for the zone) of the form `net`*N*, where *N* is a non-negative integer. For example: `net0`, `net1`, and so on. The `info` subcommand displays the automatically selected `linkname`.

Multiple zones, including the global zone, can have links with the same name at the same time.

`lower-link`

Specify the link over which the VNIC or IPoIB will be created. This property has a default value of `auto` for Ethernet links. If `pkey` is specified, `lower-link` must be specified with a valid IPoIB `phys` class datalink. The administrator may explicitly specify a value upon adding an `anet` resource. The link can be any link accepted as an argument to `dladm create-vnic`'s `-l` option or to `dladm create-part`'s `-l` option (see `dladm(1M)`). If this property is set to a `linkname` (other than `auto`) and that link does not exist, then the zone will fail to boot. When set to `auto`, the `zoneadmd(1M)` daemon will automatically choose the link over which the VNIC will be created each time the zone

boots. All IPoIB datalinks will will be skipped when selecting the default `lower-link` for creating the VNIC automatically during boot. A link will be chosen using the following heuristic:

1. A link aggregation that has a link state of `up`.
2. Of the physical Ethernet links that have a link state of `up`, the one with the alphabetically smallest link name.
3. If none is `up`, the datalink named `net0` is used if it exists.

If none of the above can be satisfied, the zone will fail to boot.

`allowed-address`
See the description of the `allowed-address` property for exclusive-IP zones in the `net` resource.

`auto-mac-address`
Holds the value of the randomly generated MAC address when the `mac-address` property (see below) is set to `random` or `auto`, so that the zone reacquires the same address on a persistent basis. To reset the randomly generated address, an administrator needs to clear this property.

`configure-allowed-address`
See the description of the `configure-allowed-address` property for exclusive-IP zones in the `net` resource.

`cos`
The 802.1p priority associated with the datalink. See `dladm(1M)` for details on this property.

`defrouter`
See the description of the `defrouter` property for exclusive-IP zones in the `net` resource.

`etsbw-lcl`
Indicates the ETS bandwidth on the TX side. See `dladm(1M)` for details on this property.

`mac-address`
Set the VNIC's MAC address based on the specified value or keyword. If the value is not a keyword, it is interpreted as a unicast MAC address. This property is not supported on IPoIB datalinks. The supported keywords are:

- `factory`: Assign a factory MAC address to the VNIC. When a factory MAC address is requested, the `mac-slot` property can be used to specify the MAC address slot identifier. Otherwise, the next available factory MAC address will be used.
- `random`: Assign a random MAC address to the VNIC. Use the `mac-prefix` property to specify a prefix. Otherwise, a default prefix consisting of a valid IEEE OUI with the local bit set will be used. This is the default value.

- `auto`: Try to use a factory MAC address first. If none is available, assign a random MAC address.

If a random MAC address is selected, then the address generated will be preserved across zone boots and zone detach/attach. This will allow zones to retain their DHCP leases by maintaining stable client IDs, and otherwise take advantage of other benefits of having stable MAC addresses.

`mac-prefix`
Specify the MAC address prefix if a random MAC address is requested. Otherwise this property is ignored. This property is not valid over IPoIB datalinks.

`mac-slot`
Specify the MAC address slot identifier if the factory MAC address is requested. Otherwise this property is ignored. This property is not valid over IPoIB datalinks.

`mtu`
The maximum transmission unit of the VNIC in bytes. See `mtu` property in `dladm(1M)`.

`maxbw`
Specify the full duplex bandwidth for the VNIC. See `maxbw` property in `dladm(1M)`. By default, the VNIC will use the `maxbw` set on the `lower-link` and if none is set then there is no bandwidth limit.

`priority`
Specify the relative priority for the VNIC. See the `priority` property in `dladm(1M)` for supported values and default.

`vlan-id`
Enable VLAN tagging for this VNIC and specify a id for the VLAN tag. There is no default value which means if this property is not set then the VNIC does not participate in any VLAN. This property is not supported on IPoIB datalinks.

`vsi-typeid`
Specify the VSI Type ID associated with a VNIC. See description in `dladm(1M)`.

`vsi-vers`
Specify the VSI Version associated with a VNIC. See description in `dladm(1M)`.

`vsi-mgrid`
Specify the VSI Manager ID associated with a VNIC. See description in `dladm(1M)`.

`rxfanout`
Specify the number of receive-side fanout threads. See description in `dladm(1M)`.

`rxrings`
Specify the receive rings for the VNIC. See `rxrings` property in `dladm(1M)` for supported values and default.

`txrings`
Specify the transmit rings for the VNIC. See `txrings` property in `dladm(1M)` for supported values and default.

`link-protection`
Enables one or more types of link protection using comma-separated values. See the `protection` property in `dladm(1M)` for supported values. It has a default value of `mac-nospoof`.

Note that adding `ip-nospoof` to this property will have no effect unless `allowed-address` is also set. Setting `allowed-address` will implicitly add `ip-nospoof` to the set of `link-protection`, and clearing `allowed-address` will remove it.

`allowed-dhcp-cids`
Setting this property will enable `dhcp-nospoof` on the VNIC. See `dladm(1M)` for details.

`pkey`
Specifies the InfiniBand Partition key value in hexadecimal. `pkey` is always treated as hexadecimal, whether it has the `0x` prefix or not. This property is only valid for IPoIB datalinks.

`linkmode`
Sets the link transport service type on an IB partition datalink. The default value is `cm`. This property is valid only for IPoIB datalinks. Valid values are:

`cm`
Connected Mode. This mode uses a default MTU of 65520 and supports a maximum MTU of 65535 bytes. If Connected Mode is not available for a remote node, Unreliable Datagram mode will automatically be used instead.

`ud`
Unreliable Datagram Mode. This mode uses a default MTU of 2044 and supports a maximum MTU of 4092 bytes.

`device`: match, allow-partition, allow-raw-io
Device name to match. This can be a pattern to match or an absolute pathname. Note that device resources and aliased datasets can have namespace conflicts in `/dev/zvol`. See `dev(7FS)`.

Both `allow-partition` and `allow-raw-io` can be set to `true` or `false`, and default to `false`. See `NOTES`.

Note – In general, adding devices to a zone can compromise the security of the system; see `NOTES`.

`rctl`: name, value
The name and *priv*/*limit*/*action* triple of a resource control. See `prctl(1)` and `rctladm(1M)`. The preferred way to set rctl values is to use the global property name associated with a specific rctl.

`attr`: name, type, value
The name, type and value of a generic attribute. The `type` must be one of `int`, `uint`, `boolean` or `string`, and the value must be of that type. `uint` means unsigned , that is, a non-negative integer.

`dataset`: name, alias
The name of a ZFS dataset to be accessed from within the zone. See `zfs(1M)`. Each dataset is aliased such that it appears as a virtual ZFS pool in the zone. The alias is the name of this virtual pool. See `zpool(1M)` for name restrictions that apply to ZFS pool names and as a result also apply to dataset alias values. The alias `rpool` is reserved from the zone's rpool dataset. Note that aliased datasets and device resources can have namespace conflicts in `/dev/zvol`. See `dev(7FS)`.

global: `cpu-shares`
The number of Fair Share Scheduler (FSS) shares to allocate to this zone. This property is incompatible with the `dedicated-cpu` resource. This property is the preferred way to set the `zone.cpu-shares` rctl.

global: `max-lwps`
The maximum number of LWPs simultaneously available to this zone. This property is the preferred way to set the `zone.max-lwps` rctl.

global: `max-msg-ids`
The maximum number of message queue IDs allowed for this zone. This property is the preferred way to set the `zone.max-msg-ids` rctl.

global: `max-processes`
The maximum number of process table slots simultaneously available to this zone. This property is the preferred way to set the `zone.max-processes` rctl. Setting this property will implicitly set the value of the `max-lwps` property to 10 times the number of process slots unless the `max-lwps` property has been set explicitly.

global: `max-sem-ids`
The maximum number of semaphore IDs allowed for this zone. This property is the preferred way to set the `zone.max-sem-ids` rctl.

global: `max-shm-ids`
The maximum number of shared memory IDs allowed for this zone. This property is the preferred way to set the `zone.max-shm-ids` rctl.

global: `max-shm-memory`
The maximum amount of shared memory allowed for this zone. This property is the preferred way to set the `zone.max-shm-memory` rctl. A scale (K, M, G, T) can be applied to the value for this number (for example, 1M is one megabyte).

global: `scheduling-class`
Specifies the scheduling class used for processes running in a zone. When this property is not specified, the scheduling class is established as follows:

- If the `cpu-shares` property or equivalent rctl is set, the scheduling class FSS is used.

- If neither `cpu-shares` nor the equivalent rctl is set and the zone's pool property references a pool that has a default scheduling class, that class is used.
- Under any other conditions, the system default scheduling class is used.

`dedicated-cpu`: ncpus, importance
The number of CPUs that should be assigned for this zone's exclusive use. The zone will create a pool and processor set when it boots. See `pooladm(1M)` and `poolcfg(1M)` for more information on resource pools. The `ncpu` property can specify a single value or a range (for example, 1-4) of processors. The `importance` property is optional; if set, it will specify the `pset.importance` value for use by `poold(1M)`. If this resource is used, there must be enough free processors to allocate to this zone when it boots or the zone will not boot. The processors assigned to this zone will not be available for the use of the global zone or other zones. This resource is incompatible with both the `pool` and `cpu-shares` properties. Only a single instance of this resource can be added to the zone.

`capped-memory`: physical, swap, locked
The caps on the memory that can be used by this zone. A scale (K, M, G, T) can be applied to the value for each of these numbers (for example, 1M is one megabyte). Each of these properties is optional but at least one property must be set when adding this resource. Only a single instance of this resource can be added to the zone. The `physical` property sets the `max-rss` for this zone. This will be enforced by `rcapd(1M)` running in the global zone. The `swap` property is the preferred way to set the `zone.max-swap` rctl. The `locked` property is the preferred way to set the `zone.max-locked-memory` rctl.

`capped-cpu`: ncpus
Sets a limit on the amount of CPU time that can be used by a zone. The unit used translates to the percentage of a single CPU that can be used by all user threads in a zone, expressed as a fraction (for example, `.75`) or a mixed number (whole number and fraction, for example, `1.25`). An `ncpu` value of `1` means 100% of a CPU, a value of `1.25` means 125%, `.75` mean 75%, and so forth. When projects within a capped zone have their own caps, the minimum value takes precedence.

The `capped-cpu` property is an alias for `zone.cpu-cap` resource control and is related to the `zone.cpu-cap` resource control. See `resource_controls(5)`.

`admin`: user, auths
Delegates zone administrative authorizations to the specified user or role. The user must correspond to a valid local account. The allowed values for auths are:

`login`
Allows authenticated use of `zlogin(1)` into this zone.

`manage`
Allows normal management of the configured zone.

`copyfrom`
Allows the use of the specified zone as a source from which to clone a new zone.

`rootzpool`: storage
Defines one or more storage resources to be used exclusively for a dedicated `zpool` containing the zone installation. The allowed values for storage are defined in `suri(5)`.

`zpool`: storage, name
Defines one or more storage resources to be used exclusively for a `zpool` delegated to the zone. The allowed values for storage are defined in `suri(5)`. The allowed values for name are defined in `zpool(1M)`. The name `rpool` is not permitted.

The following table summarizes resources, property-names, and types:

```
resource          property-name   type
(global)          zonename        simple
(global)          zonepath        simple
(global)          autoboot        simple
(global)          bootargs        simple
(global)          pool            simple
(global)          limitpriv       simple
(global)          brand           simple
(global)          ip-type         simple
(global)          hostid          simple
(global)          cpu-shares      simple
(global)          max-lwps        simple
(global)          max-msg-ids     simple
(global)          max-sem-ids     simple
(global)          max-shm-ids     simple
(global)          max-shm-memory  simple
(global)          scheduling-class simple
(global)          fs-allowed      list of simple
fs                dir             simple
                  special         simple
                  raw             simple
                  type            simple
                  options         list of simple
net               address         simple
                  allowed-address list of simple
                  configure-allowed-address simple
                  cos             simple
                  defrouter       list of simple
                  etsbw_lcl       simple
                  physical        simple
anet              linkname        simple
                  lower-link      simple
                  allowed-address list of simple
                  auto-mac-address simple
                  configure-allowed-address simple
                  defrouter       list of simple
                  mac-address     simple
                  mac-slot        simple
```

```
                    mac-prefix      simple
                    mtu             simple
                    maxbw           simple
                    priority        simple
                    vlan-id         simple
                    vsi-typeid      simple
                    vsi-vers        simple
                    vsi-mgrid       simple
                    rxfanout        simple
                    rxrings         simple
                    txrings         simple
                    link-protection list of simple
                    allowed-dhcp-cids list of simple
                    pkey            simple
                    linkmode        simple
device              match           simple
                    allow-partition simple
                    allow-raw-io    simple
rctl                name            simple
                    value           list of complex
attr                name            simple
                    type            simple
                    value           simple
dataset             name            simple
                    alias           simple
dedicated-cpu       ncpus           simple or range
                    importance      simple

capped-memory       physical        simple with scale
                    swap            simple with scale
                    locked          simple with scale

capped-cpu          ncpus           simple

admin               user            simple
                    auths           list of simple
rootzpool           storage         simple

zpool               storage         simple
                    name            simple
```

To further specify things, the breakdown of the complex property “value” of the “rctl” resource type, it consists of three name/value pairs, the names being “priv”, “limit” and “action”, each of which takes a simple value. The “name” property of an “attr” resource is syntactically restricted in a fashion similar but not identical to zone names: it must begin with an alphanumeric, and can contain alphanumerics plus the hyphen (-), underscore (_), and dot (.) characters. Attribute names beginning with “zone” are reserved for use by the system. Finally, the “autoboot” global property must have a value of “true“ or “false”.

Using Kernel Statistics to Monitor CPU Caps

Using the kernel statistics (kstat(3KSTAT)) module caps, the system maintains information for all capped projects and zones. You can access this information by reading kernel statistics (kstat(3KSTAT)), specifying caps as the kstat module name. The following command displays kernel statistics for all active CPU caps:

```
# kstat caps::'/cpucaps/'
```

A kstat(1M) command running in a zone displays only CPU caps relevant for that zone and for projects in that zone. See EXAMPLES.

The following are cap-related arguments for use with kstat(1M):

caps
: The kstat module.

project_caps or zone_caps
: kstat class, for use with the kstat -c option.

cpucaps_project_*id* or cpucaps_zone_*id*
: kstat name, for use with the kstat -n option. *id* is the project or zone identifier.

The following fields are displayed in response to a kstat(1M) command requesting statistics for all CPU caps.

module
: In this usage of kstat, this field will have the value caps.

name
: As described above, cpucaps_project_*id* or cpucaps_zone_*id*

above_sec
: Total time, in seconds, spent above the cap.

below_sec
: Total time, in seconds, spent below the cap.

maxusage
: Maximum observed CPU usage.

nwait
: Number of threads on cap wait queue.

usage
: Current aggregated CPU usage for all threads belonging to a capped project or zone, in terms of a percentage of a single CPU.

value
: The cap value, in terms of a percentage of a single CPU.

zonename
: Name of the zone for which statistics are displayed.

See EXAMPLES for sample output from a kstat command.

Options The following options are supported:

-f *command_file*
Specify the name of zonecfg command file. *command_file* is a text file of zonecfg subcommands, one per line.

-z *zonename*
Specify the name of a zone. Zone names are case sensitive. Zone names must begin with an alphanumeric character and can contain alphanumeric characters, the underscore (_) the hyphen (-), and the dot (.). The name global and all names beginning with SYS are reserved and cannot be used.

Subcommands You can use the add and select subcommands to select a specific resource, at which point the scope changes to that resource. The end and cancel subcommands are used to complete the resource specification, at which time the scope is reverted back to global. Certain subcommands, such as add, remove and set, have different semantics in each scope.

zonecfg supports a semicolon-separated list of subcommands. For example:

```
# zonecfg -z myzone "add net; set physical=myvnic; end"
```

Subcommands which can result in destructive actions or loss of work have an -F option to force the action. If input is from a terminal device, the user is prompted when appropriate if such a command is given without the -F option otherwise, if such a command is given without the -F option, the action is disallowed, with a diagnostic message written to standard error.

The following subcommands are supported:

add *resource-type* (global scope)
add *property-name property-value* (resource scope)
In the global scope, begin the specification for a given resource type. The scope is changed to that resource type.

In the resource scope, add a property of the given name with the given value. The syntax for property values varies with different property types. In general, it is a simple value or a list of simple values enclosed in square brackets, separated by commas ([foo,bar,baz]). See PROPERTIES.

cancel
End the resource specification and reset scope to global. Abandons any partially specified resources. cancel is only applicable in the resource scope.

clear *property-name*
Clear the value for the property.

commit
Commit the current configuration from memory to stable storage. The configuration must be committed to be used by zoneadm. Until the in-memory configuration is committed,

you can remove changes with the `revert` subcommand. The `commit` operation is attempted automatically upon completion of a `zonecfg` session. Since a configuration must be correct to be committed, this operation automatically does a verify.

`create [-F] [ -a` *path* `|-b | -t` *template*`]`
Create an in-memory configuration for the specified zone. Use `create` to begin to configure a new zone. See `commit` for saving this to stable storage.

If you are overwriting an existing configuration, specify the `-F` option to force the action. Specify the `-t` *template* option to create a configuration identical to *template*, where *template* is the name of a configured zone.

`create` uses a default template of `SYSdefault`. The default template can be changed on a system-wide basis using the `default_template` SMF property of the `svc:/system/zones:default` service. An administrator can override the default for this zone using `-t` (with a specific template) or `-b` (to use a blank template).

Use the `-a` *path* option to facilitate configuring a detached zone on a new host. The *path* parameter is the `zonepath` location of a detached zone that has been moved on to this new host. Once the detached zone is configured, it should be installed using the "`zoneadm attach`" command (see `zoneadm(1M)`). All validation of the new zone happens during the `attach` process, not during zone configuration.

Use the `-b` option to create a blank configuration. Without arguments, `create` applies the Oracle Sun default settings.

`delete [-F]`
Delete the specified configuration from memory and stable storage. This action is instantaneous, no commit is necessary. A deleted configuration cannot be reverted.

Specify the `-F` option to force the action.

`end`
End the resource specification. This subcommand is only applicable in the resource scope. `zonecfg` checks to make sure the current resource is completely specified. If so, it is added to the in-memory configuration (see `commit` for saving this to stable storage) and the scope reverts to global. If the specification is incomplete, it issues an appropriate error message.

`export [-f` *output-file*`]`
Print configuration to standard output. Use the `-f` option to print the configuration to *output-file*. This option produces output in a form suitable for use in a command file.

`help [usage] [`*subcommand*`] [syntax] [`*command-name*`]`
Print general help or help about given topic.

`info zonename | zonepath | autoboot | brand | pool | limitpriv`
`info [`*resource-type* `[`*property-name*`=`*property-value*`]*]`
Display information about the current configuration. If *resource-type* is specified, displays only information about resources of the relevant type. If any *property-name* value pairs are

specified, displays only information about resources meeting the given criteria. In the resource scope, any arguments are ignored, and `info` displays information about the resource which is currently being added or modified.

`remove` *resource-type*{*property-name*=*property-value*}(global scope)
In the global scope, removes the specified resource. The `[]` syntax means 0 or more of whatever is inside the square braces. If you want only to remove a single instance of the resource, you must specify enough property name-value pairs for the resource to be uniquely identified. If no property name-value pairs are specified, all instances will be removed. If there is more than one pair is specified, a confirmation is required, unless you use the `-F` option.

`select` *resource-type* {*property-name*=*property-value*}
Select the resource of the given type which matches the given *property-name property-value* pair criteria, for modification. This subcommand is applicable only in the global scope. The scope is changed to that resource type. The `{}` syntax means 1 or more of whatever is inside the curly braces. You must specify enough *property -name property-value* pairs for the resource to be uniquely identified.

`set` *property-name*=*property- value*
Set a given property name to the given value. Some properties (for example, `zonename` and `zonepath`) are global while others are resource-specific. This subcommand is applicable in both the global and resource scopes.

`verify` [`-v`]
Verify the current configuration for correctness:

- All resources have all of their required properties specified.
- A `zonepath` is specified.

If the `-v` option is specified, warnings will be issued if there is a potential for devices specified in device resources to conflict with and hide ZFS volumes created within aliased datasets. See dev(7FS).

`revert` [`-F`]
Revert the configuration back to the last committed state. The `-F` option can be used to force the action.

`exit` [`-F`]
Exit the `zonecfg` session. A commit is automatically attempted if needed. You can also use an EOF character to exit `zonecfg`. The `-F` option can be used to force the action.

Examples

EXAMPLE 1 Creating the Environment for a New Zone

In the following example, `zonecfg` creates the environment for a new zone. `/usr/local` is loopback mounted from the global zone into `/opt/local`. `/opt/sfw` is loopback mounted from the global zone, a VNIC over `nxge0` is added to the zone with three IP addresses, and a limit on the number of fair-share scheduler (FSS) CPU shares for a zone is set using the `rctl`

EXAMPLE 1 Creating the Environment for a New Zone *(Continued)*

resource type. The example also shows how to select a given resource for modification; in this case, by selecting the `anet` resource that is automatically created by `zonecfg`.

```
example# zonecfg -z myzone3
my-zone3: No such zone configured
Use 'create' to begin configuring a new zone.
zonecfg:myzone3> create
zonecfg:myzone3> set zonepath=/export/home/my-zone3
zonecfg:myzone3> set autoboot=true
zonecfg:myzone3> add fs
zonecfg:myzone3:fs> set dir=/opt/local
zonecfg:myzone3:fs> set special=/usr/local
zonecfg:myzone3:fs> set type=lofs
zonecfg:myzone3:fs> add options [ro,nodevices]
zonecfg:myzone3:fs> end
zonecfg:myzone3> add fs
zonecfg:myzone3:fs> set dir=/mnt
zonecfg:myzone3:fs> set special=/dev/dsk/c0t0d0s7
zonecfg:myzone3:fs> set raw=/dev/rdsk/c0t0d0s7
zonecfg:myzone3:fs> set type=ufs
zonecfg:myzone3:fs> end
zonecfg:myzone3> add fs
zonecfg:myzone3:fs> set dir=/opt/sfw
zonecfg:myzone3:fs> set special=/opt/sfw
zonecfg:myzone3:fs> set type=lofs
zonecfg:myzone3:fs> add options [ro,nodevices]
zonecfg:myzone3:fs> end
zonecfg:myzone3> select anet linkname=net0
zonecfg:myzone3:anet> set lower-link=nxge0
zonecfg:myzone3:anet> set allowed-address="192.168.0.1/24,192.168.1.2/\
          24,192.168.2.3/24"
zonecfg:myzone3:anet> end
zonecfg:my-zone3> set cpu-shares=5
zonecfg:my-zone3> add capped-memory
zonecfg:my-zone3:capped-memory> set physical=50m
zonecfg:my-zone3:capped-memory> set swap=100m
zonecfg:my-zone3:capped-memory> end
zonecfg:myzone3> exit
```

EXAMPLE 2 Creating an Exclusive-IP Zone

The following example creates a zone that is assigned a VNIC named `net0`. The link over which the VNIC is created is automatically determined. The IP addresses and routing are configured inside the new zone using `ipadm(1M)`.

```
example# zonecfg -z excl
zonecfg:excl> create
```

EXAMPLE 2 Creating an Exclusive-IP Zone *(Continued)*

```
zonecfg:excl> set zonepath=/export/zones/excl
zonecfg:excl> exit
```

EXAMPLE 3 Creating a Shared-IP Zone

The following example creates a zone that shares an IP stack with the global zone, and is assigned a single IP address and default router.

```
example# zonecfg -z shared
zonecfg:shared> create -b
zonecfg:shared> set zonepath=/export/zones/shared
zonecfg:shared> set ip-type=shared
zonecfg:shared> add net
zonecfg:shared:net> set physical=nge0
zonecfg:shared:net> set address=192.168.0.3/24
zonecfg:shared:net> set defrouter=192.168.0.1
zonecfg:shared:net> end
zonecfg:shared> exit
```

EXAMPLE 4 Associating a Zone with a Resource Pool

The following example shows how to associate an existing zone with an existing resource pool:

```
example# zonecfg -z myzone
zonecfg:myzone> set pool=mypool
zonecfg:myzone> exit
```

For more information about resource pools, see `pooladm(1M)` and `poolcfg(1M)`.

EXAMPLE 5 Changing the Name of a Zone

The following example shows how to change the name of an existing zone:

```
example# zonecfg -z myzone
zonecfg:myzone> set zonename=myzone2
zonecfg:myzone2> exit
```

EXAMPLE 6 Changing the Privilege Set of a Zone

The following example shows how to change the set of privileges an existing zone's processes will be limited to the next time the zone is booted. In this particular case, the privilege set will be the standard safe set of privileges a zone normally has along with the privilege to change the system date and time:

```
example# zonecfg -z myzone
zonecfg:myzone> set limitpriv="default,sys_time"
zonecfg:myzone2> exit
```

EXAMPLE 7 Setting the zone.cpu-shares Property for the Global Zone

The following command sets the `zone.cpu-shares` property for the global zone:

```
example# zonecfg -z global
zonecfg:global> set cpu-shares=5
zonecfg:global> exit
```

EXAMPLE 8 Using Pattern Matching

The following commands illustrate `zonecfg` support for pattern matching. In the zone `flexlm`, enter:

```
zonecfg:flexlm> add device
zonecfg:flexlm:device> set match="/dev/cua/a00[2-5]"
zonecfg:flexlm:device> end
```

In the global zone, enter:

```
global# ls /dev/cua
a     a000  a001  a002  a003  a004  a005  a006  a007  b
```

In the zone `flexlm`, enter:

```
flexlm# ls /dev/cua
a002  a003  a004  a005
```

EXAMPLE 9 Setting a Cap for a Zone to Three CPUs

The following sequence uses the `zonecfg` command to set the CPU cap for a zone to three CPUs.

```
zonecfg:myzone> add capped-cpu
zonecfg:myzone>capped-cpu> set ncpus=3
zonecfg:myzone>capped-cpu>capped-cpu> end
```

The preceding sequence, which uses the capped-cpu property, is equivalent to the following sequence, which makes use of the `zone.cpu-cap` resource control.

```
zonecfg:myzone> add rctl
zonecfg:myzone:rctl> set name=zone.cpu-cap
zonecfg:myzone:rctl> add value (priv=privileged,limit=300,action=none)
zonecfg:myzone:rctl> end
```

EXAMPLE 10 Using `kstat` to Monitor CPU Caps

The following command displays information about all CPU caps.

```
# kstat -n /cpucaps/
module: caps                            instance: 0
name:   cpucaps_project_0               class:    project_caps
        above_sec                       0
        below_sec                       2157
```

EXAMPLE 10 Using `kstat` to Monitor CPU Caps *(Continued)*

```
        crtime                          821.048183159
        maxusage                        2
        nwait                           0
        snaptime                        235885.637253027
        usage                           0
        value                           18446743151372347932
        zonename                        global

module: caps                            instance: 0
name:   cpucaps_project_1               class:    project_caps
        above_sec                       0
        below_sec                       0
        crtime                          225339.192787265
        maxusage                        5
        nwait                           0
        snaptime                        235885.637591677
        usage                           5
        value                           18446743151372347932
        zonename                        global

module: caps                            instance: 0
name:   cpucaps_project_201             class:    project_caps
        above_sec                       0
        below_sec                       235105
        crtime                          780.37961782
        maxusage                        100
        nwait                           0
        snaptime                        235885.637789687
        usage                           43
        value                           100
        zonename                        global

module: caps                            instance: 0
name:   cpucaps_project_202             class:    project_caps
        above_sec                       0
        below_sec                       235094
        crtime                          791.72983782
        maxusage                        100
        nwait                           0
        snaptime                        235885.637967512
        usage                           48
        value                           100
        zonename                        global

module: caps                            instance: 0
name:   cpucaps_project_203             class:    project_caps
```

EXAMPLE 10 Using `kstat` to Monitor CPU Caps *(Continued)*

```
        above_sec                       0
        below_sec                       235034
        crtime                          852.104401481
        maxusage                        75
        nwait                           0
        snaptime                        235885.638144304
        usage                           47
        value                           100
        zonename                        global

module: caps                            instance: 0
name:   cpucaps_project_86710           class:    project_caps
        above_sec                       22
        below_sec                       235166
        crtime                          698.441717859
        maxusage                        101
        nwait                           0
        snaptime                        235885.638319871
        usage                           54
        value                           100
        zonename                        global

module: caps                            instance: 0
name:   cpucaps_zone_0                  class:    zone_caps
        above_sec                       100733
        below_sec                       134332
        crtime                          821.048177123
        maxusage                        207
        nwait                           2
        snaptime                        235885.638497731
        usage                           199
        value                           200
        zonename                        global

module: caps                            instance: 1
name:   cpucaps_project_0               class:    project_caps
        above_sec                       0
        below_sec                       0
        crtime                          225360.256448422
        maxusage                        7
        nwait                           0
        snaptime                        235885.638714404
        usage                           7
        value                           18446743151372347932
        zonename                        test_001
```

EXAMPLE 10 Using `kstat` to Monitor CPU Caps *(Continued)*

```
module: caps                            instance: 1
name:   cpucaps_zone_1                  class:    zone_caps
        above_sec                       2
        below_sec                       10524
        crtime                          225360.256440278
        maxusage                        106
        nwait                           0
        snaptime                        235885.638896443
        usage                           7
        value                           100
        zonename                        test_001
```

EXAMPLE 11 Displaying CPU Caps for a Specific Zone or Project

Using the `kstat -c` and `-i` options, you can display CPU caps for a specific zone or project, as below. The first command produces a display for a specific project, the second for the same project within zone 1.

```
# kstat -c project_caps

# kstat -c project_caps -i 1
```

EXAMPLE 12 Delegating Zone Administrative Rights

The following example shows how to assign administrative rights for the current zone to a role.

```
example# zonecfg -z myzone
zonecfg:myzone> add admin
zonecfg:myzone:admin> set user=zadmin
zonecfg:myzone:admin> set auths=login,manage
zonecfg:myzone:admin> end
zonecfg:myzone> commit
```

The result of executing these commands would be an updated entry in the RBAC `user_attr(4)` database, similar to the following:

```
zadmin::::type=role;\
auths=solaris.zone.login/myzone,solaris.zone.manage/myzone;profiles=\
Zone Management
```

EXAMPLE 13 Creating an Exclusive-IP Zone with Non-Default Properties

The following example creates a zone with an automatically created VNIC over `mylink0` with the given MAC address, maximum bandwidth of 100 Mbps, high priority, dedicated hardware rings for RX side, no dedicated hardware rings for the TX side (that is, software-based) and with a VLAN id 2.

EXAMPLE 13 Creating an Exclusive-IP Zone with Non-Default Properties *(Continued)*

```
example# zonecfg -z excl
excl: No such zone configured
Use 'create' to begin configuring a new zone
zonecfg:excl> create -b
zonecfg:excl> set zonepath=/export/zones/excl
zonecfg:excl> add anet
zonecfg:excl:anet> set linkname=mynic0
zonecfg:excl:anet> set lower-link=mylink0
zonecfg:excl:anet> set mac-address=8:0:20:fe:4e:b8
zonecfg:excl:anet> set maxbw=100M
zonecfg:excl:anet> set priority=high
zonecfg:excl:anet> set vlan-id=2
zonecfg:excl:anet> set rxrings=hw
zonecfg:excl:anet> set txrings=sw
zonecfg:excl:anet> end
zonecfg:excl> exit
```

EXAMPLE 14 Creating a Read-Only Zone

The following example creates a new zone that has its root filesystem protected against modifications by the zone. Files in `/var` are writable by virtue of the `fixed-configuration` profile that is applied.

```
example# zonecfg -z rozone
rozone: No such zone configured
Use 'create' to begin configuring a new zone
zonecfg:rozone> create
zonecfg:rozone> set brand=solaris
zonecfg:rozone> set zonepath=/export/zones/rozone
zonecfg:rozone> set autoboot=true
zonecfg:rozone> set file-mac-profile=fixed-configuration
zonecfg:rozone> set ip-type=exclusive
zonecfg:rozone> add net
zonecfg:rozone:net> set physical=vnic0
zonecfg:rozone:net> end
zonecfg:rozone> exit
```

EXAMPLE 15 Creating an Exclusive-IP Zone with an IB Partition

The following example creates a zone with default properties. The zone will automatically create a IPoIB datalink when the zone boots and delete the datalink when the zone halts.

```
example# zonecfg -z excl
excl: No such zone configured
Use 'create' to begin configuring a new zone
zonecfg:excl> create
zonecfg:excl> set zonepath=/export/zones/excl
```

EXAMPLE 15 Creating an Exclusive-IP Zone with an IB Partition *(Continued)*

```
zonecfg:excl> set ip-type=exclusive
zonecfg:excl> add anet
zonecfg:excl> set linkname=part0
zonecfg:excl> set lower-link=net4
zonecfg:excl> set pkey=ffff
zonecfg:excl:anet> end
zonecfg:excl> exit
```

EXAMPLE 16 Creating a Zone Installed into a Dedicated Storage Resource and zpool

The following example creates a new zone with a `zpool` resource comprised of one storage resource containing the entire zone installation. The `zpool` will be automatically created or a pre-created `zpool` will be imported during zone installation. The name will be `zoss_rpool`.

```
example# zonecfg -z zoss
zoss: No such zone configured
Use 'create' to begin configuring a new zone
zonecfg:zoss> create
zonecfg:zoss> set zonepath=/zoss
zonecfg:zoss> add rootzpool
zonecfg:zoss:rootzpool> add storage iscsi://127.0.0.1/luname.naa.6001\
44f03d70c80000004ea57da10001
zonecfg:zoss:rootzpool> end
zonecfg:zoss> exit
```

EXAMPLE 17 Creating a Zone with a Delegated zpool Resource

The following example creates a new zone with a `zpool` resource delegated to the zone comprised of two storage resources. The `zpool` will be automatically created or a pre-created `zpool` will be imported during zone installation. The name will be `zoss_mypool`.

```
example# zonecfg -z zoss
zoss: No such zone configured
Use 'create' to begin configuring a new zone
zonecfg:zoss> create
zonecfg:zoss> set zonepath=/zoss
zonecfg:zoss> add zpool
zonecfg:zoss:zpool> set name=mypool
zonecfg:zoss:zpool> add storage dev:/dev/dsk/c0t1d0
zonecfg:zoss:zpool> add storage dev:/dev/dsk/c1t1d0
zonecfg:zoss:zpool> end
zonecfg:zoss> exit
```

Exit Status

The following exit values are returned:

0
Successful completion.

1
An error occurred.

2
Invalid usage.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/zones
Interface Stability	Volatile

See Also ppriv(1), prctl(1), zlogin(1), dladm(1M), format(1M), ipadm(1M), kstat(1M), mount(1M), pooladm(1M), poolcfg(1M), poold(1M), rcapd(1M), rctladm(1M), route(1M), suriadm(1M), svcadm(1M), zfs(1M), zoneadm(1M), zpool(1M), priv_str_to_set(3C), kstat(3KSTAT), user_attr(4), vfstab(4), attributes(5), brands(5), fnmatch(5), mwac(5), privileges(5), rbac(5), resource_controls(5), suri(5), zones(5), dev(7FS), hsfs(7FS), zfs(7FS), uscsi(7I)

Oracle Solaris Administration: Oracle Solaris Zones, Oracle Solaris 10 Zones, and Resource Management

Notes All character data used by `zonecfg` must be in US-ASCII encoding.

Adding a device to a zone, in general, can allow the zone to adversely affect the security and stability of the system, as not all devices have been audited for secure use inside a zone.

Storage devices using the `sd` or `ssd` target driver (this can be checked using `prtconf -D /dev/dsk/c2t40d3`, for example) can be safely delegated to a zone. This will allow a zone admin to label and partition such devices.

In order to allow disk labelling by means of format(1M), an entire disk/LUN should be delegated to a zone, and the `allow-partition` property set. For example:

```
zonecfg:myzone> add device
zonecfg:myzone> set match=/dev/*dsk/c2t40d3*
zonecfg:myzone> set allow-partition=true
zonecfg:myzone> end
```

While it is not recommended, it is also possible to delegate just a single slice (for example, `match=/dev/dsk/c2t40d3s0`) of a disk. In order for this to be safe, the `allow-partition` property must not be true, and the slice or partition must not overlap the disk header of disk labels (these are located within the first two or last two blocks of the partition or disk).

Raw access to storage devices can be enabled by setting the `allow-raw-io` property to `true`. This is unsafe, as it allows raw SCSI commands (see uscsi(7I)) to be performed by zone processes.

Inside a zone, device-in-use checking does not work, as the `/devices/` tree it relies upon is not present. A future project might address this limitation.

Name zonep2vchk – check a global zone's configuration for physical to virtual migration into non-global zone

Synopsis

```
zonep2vchk -V

zonep2vchk [-T release] -c

zonep2vchk [-T release] [-P] [-b] [ -s path[,path...] ] [-S file]
     [ -r {time}(h|m|s} ] [-x] [-e execname[,execname...] ]
     [-E file]
```

Description The `zonep2vchk` utility is used to evaluate a global zone's configuration before the process of physical-to-virtual (`p2v`) migration into a non-global zone.

The p2v process involves archiving a global zone (source), and then installing a non-global zone (target) using that archive. See the `install -a` documentation in the `solaris(5)` and `solaris10(5)` man pages.

`zonep2vchk` serves two functions. First, it can be used to report issues on the source which might prevent a successful `p2v` migration. Second, it can output a template `zonecfg`, which can be used to assist in configuring the non-global zone target.

`zonep2vchk` can be executed on a Solaris 10 or later global zone. To execute on Solaris 10, copy the `zonep2vchk` utility to the Solaris 10 source global zone.

Security The `zonep2vchk` utility must be run with an effective user id of zero. It interrogates the configuration state of a variety of Solaris subsystems.

Options The following options are supported:

`-V`
Display the command version and exit.

`-T` *release*
Specify the target release. The defaults are:

```
Global Zone         Default Target
Solaris 10          S10
Solaris 11          S11
```

Any configuration files generated by `zonep2chk` will be applicable to the target release. See `-c` below.

When run on Solaris 10, a target release of `S11` can be specified, which will check for `p2v` into a Solaris 10 Branded zone.

When the target is S10, it is assumed that a shared stack will be used. Any issues that will require an exclusive IP stack will be reported.

When the target is S11, it is assumed that an exclusive IP stack will be used.

If a particular feature in use by the global zone requires a particular patch/update level of the target to function, this information will be printed in the `zonep2vchk` output.

`-P`
Generate machine-parseable output. See the section "Parseable Output Format" below.

`-c`
Display a template zone configuration on stdout in the form of `zonecfg(1M)` `export` output. This configuration will contain resource limits and network configuration based on the source host's physical resources and networking configuration.

`-b`
Perform basic checks. This will check the global zone for issues that could prevent a successful `p2v`. This is the default behavior if none of `-b`, `-c`, `-s`, `-S`, `-r`, `-x` are specified.

`-r`{*time*} (h|m|s)
Perform runtime checks for the specified duration. This will analyze the currently executing processes in the global zone, and report issues that could prevent successful execution inside a non-global zone. Issues reported reflect actions made by the processes during the time in which `zonep2vchk` was executing.

`-x`
Perform runtime checks (as with `-r`) until `SIGINT` is received, such as is delivered by `Ctrl-c` from most shells.

`-e` *execname*[,*execname*...]
When performing runtime analysis (`-r`, `-x`), limit inspected programs to those matching the specified list of *execname*s. The *execname* is the name of process, as returned by `ps -o comm`. It is not necessary for named processes to exist when `zonep2vchk` is invoked. Any matching processes created while `zonep2vchk` is running will be inspected.

`-E` *file*
Similar to `-e`, but reads the list of *execname*s from *file*, one per line.

`-s` *path*[,*path*...]
Perform static binary analysis on the files or directories specified. This will inspect ELF binaries for system and library calls that might affect function inside a zone. Directories will be recursed, and non-ELF files will be ignored.

`-S` *file*
Similar to `-s`, but reads the path list from *file*, one per line.

Parseable Output Format

`zonep2vchk` will ouput a single line of parseable output for each issue detected. The line format is:

category:*issue*:*field1*:[*field2*:...]

Each field is delimited by a colon (`:`). Colon characters escaped with a backslash (`\:`) should not be treated as field delimiters.

Multiple instances of the same issue can be reported, each with fields describing the particular instance of the issue.

Below the existing categories and issues are defined. Future versions of `zonep2vchk` might include additional categories and issues. Existing issues might have new fields added after the existing fields for existing issues.

header Category — The `header` category lists information about the source, target, and `zonep2vchk` version. The issues in this category are:

`version`
The version of the `zonep2vchk` command.

Field1: The version of the `zonep2vchk` command.

`source`
Information about the source system.

Field1: The nodename of the source system.

Field2: The /etc/release version of the source system.

Field3: The kernel version of the source system.

Field4: The platform of the source system.

`target`
Information about the specified target of the `p2v` check.

Field1: The Solaris version of the target.

Field2: The brand type that would be used on the target.

Field3: The ip-type of the expected zone on the target.

footer Category — The `footer` category lists final summary information. The issues in this category are:

`issues`
A summary of the number of issues found.

Field1: The number of issues detected.

incompatible Category — The `incompatible` category represents issues that will not function in a non-global zone. The issues in this category are:

`etcsystem`
An `/etc/system` tunable exists. These tunables do not function inside a zone. The `/etc/system` tunable can be transferred to the target global zone, but it will affect the entire system, including all zones and the global zone. If there is an alternate tunable that can be configured from within the zone, this tunable is described.

Field1:
The `/etc/system` tunable setting.

Field2:
One of:

`noalternate` There is no alternate tunable from within a non-global zone.

`obsolete` The tunable is obsolete on the target. It no longer serves any function.

`replaced` The tunable has been replaced on the target. The replacement is configured in the global zone, and described by fields 3 and 4.

`alternate` An alternate tunable exists. This tunable can be configured from within a non-global zone. The tunable is described by fields 3 and 4.

`noinfo` `zonep2vchk` is not knowledgeable of the tunable. Tunable likely has no alternate inside a zone.

Field3:
Type of alternate/replacement tunable.

Field4:
Description of alternate/replacement tunable.

`be`
More than one boot environment exists. Only the active boot environment will be transferable to the non-global zone.

Field1: The name of the non-active boot environment.

`unsupported`
A feature is enabled that will not function in a zone.

Field1: `mobileip` The mobile IP agent, which does not function in a zone, is configured. See `mipagent(1M)` for details.

`nfs` (S10 sources only)
The system is sharing a filesystem by means of NFS. Native zones on Solaris 10 and Solaris 10 zones on Solaris 11 cannot share by means of NFS.

Field1: Path of file system being shared.

`smb`
The system is sharing a filesystem by means of in-kernel `smb/cifs`. Zones cannot share filesystems by means of SMB.

Field1: Path of file system being shared.

`pkg`
A package delivering software known not to work in a zone is installed.

Field1: Name of the package.

`iscsi-target`
The system is exporting an ISCSI target. Zones cannot export iSCSI targets.

Field1: Name of the iSCSI target.

`fcoe-target`
The system has configured an FCOE target. Zones cannot configure FCOE targets.

Field1: Ethernet device used.

Field2: WWN of the FCOE target.

`fc-target`
The system has configured an Fiberchannel target. Zones cannot configure Fiberchannel targets.

Field1: WWN of the Fiberchannel target.

`npiv`
The system has configured a virtual NPIV HBA. Zones cannot configure virtual HBAs.

Field1: Physical WWN hosting the virtual HBA.

Field2: Virtual WWN.

`scsi`
The system has configured an SCSI block device. Zones cannot configure scsi block devices.

Field1: Object configured as a SCSI device.

`svcnotallowed`
A service is enabled that will not function in a zone.

Field1: Name of the service.

`resourcepool`
A Solaris resource pool is configured. Zones cannot configure resource pools.

Field1: Name of the pool.

`pset`
A processor set is configured. Zones cannot configure processor sets.

Field1: Processor set ID.

Field2: List of CPU IDs in the processor set.

`zones`
Zones are configured. A zone cannot host zones. Any zones will not exist in the target non-global zone after `p2v`. Zones can be migrated separately using the `detach`/`attach` features in `zoneadm`(1M).

Field1: Name of the zone.

Field2: State of the zone.

`lofi` (Solaris 10 targets only)
A lofi device is configured. A zone cannot configure lofi devices.

Field1: Name of the lofi device.

Field2: Path of the file backing the device.

`syscall` (generated by `-s` and `-f`)
A binary makes a system or library call that cannot be made from a zone.

Field1: Name of the the binary file.

Field2: Name of the system or library call.

`syscallargs` (generated by `-s` and `-f`)
A binary makes a system or library call that cannot be made from a zone if called with certain arguments.

Field1: Name of the system or library call.

See regular output (no `-P`) for details on disallowed arguments.

`lib` (generated by `-s` and `-f`)
A binary links with a library that cannot be used inside a zone.

Field1: Name of the binary file.

Field2: Name of the disallowed library.

`privnotallowed` (generated by `-r` and `-x`)
A privilege is used by a process that cannot be added to a zone.

Field1: Name of the process.

Field2: Name of the privilege.

`devnotallowed` (generated by `-r` and `-x`)
A device is opened by a process that cannot be added to a zone.

Field1: Name of the process.

Field2: Name of the device.

configuration Category

The `configuration` category represents issues that will require a configuration setting to allow the issue to function inside the non-global zone. This could be a `zonecfg(1M)` configuration setting, a configuration change in the global zone, or both.

The issues in this category are:

`datalink`
A datalink feature is configured that cannot be configured from within a zone. The datalink feature must be configured in the global zone, and if necessary, delegated to the zone using `zonecfg add anet` (Solaris 11 only) or `zonecfg add net`.

Field1: Name of the datalink feature. One of:

`aggr`	Aggregation.
`ibiface`	Infiniband interface.
`ibpart`	Infiniband partition.
`vnic`	Virtual NIC.
`etherstub`	Ethernet stub.
`bridge`	A bridge instance.
`secobj`	A wireless WPA or WEB security object.

Field2: Datalink object name.

`dhcp-server` (Solaris 10 targets only)
The host is a DHCP server. To provide DHCP service, a zone must have `ip-type=exclusive`, or have the the privilege `net_rawaccess` and the device `/dev/ip`. Note that this will allow a shared stack zone to read and write raw IP packets on the network, similar to an exclusive stack zone or global zone.

Field1: FMRI of the DHCP server service.

`ntp-client`
An NTP client service is enabled. This service updates the system clock. Since all zones share the same system clock, this service is disabled automatically during `p2v`. If it is desired that the zone update the system clock on the target host, the zone will need the privilege `sys_time`, and the service will need to be enabled inside the zone after `p2v`.

Field1: FMRI of the client service.

`driverconf`
A networking device contains configuration settings in its `.conf` file. Zones cannot configure drivers. The driver must be configured in the global zone. Some network driver settings might be configurable using `dladm(1M)` instead of editing a driver configuration file.

Field1: Path of the configuration file.

`ifname` (Solaris 10 targets only)
An existing configuration file will be impacted by the change of a network device name. For example, an `/etc/hostname.bge0` file will be impacted if the network device given to the target non-global zone is not `bge0`.

Field1: Path of the impacted file.

`iscsi-initiator`
The system is accessing an iSCSI target as a client. Zones cannot access iSCSI targets. The global zone must be the iSCSI initiator. The device can then be added to the zone using `zonecfg add device`.

Field1: iSCSI target being accessed.

`fcoe-initiator`
The system has an FCOE initiator configured. A zone cannot configure an FCOE initiator. The global zone must configure the FCOE initiator, and make the SCSI target devices available to the zone using `zonecfg add fs` or `zonecfg add device`.

Field1: Ethernet network device.

Field2: WWN of the initiator.

`fc-initiator`
The system has an HBA Fiberchannel port online. A zone cannot access a Fiberchannel target. The target must be accessed from the global zone and made available to the zone.

Field1: Fiberchannel HBA port WWN.

`linkprop`
Datalink properties are configured. A zone cannot configure datalink properties. They must be configured from the global zone.

Field1: Name of the datalink.

Field2: Property name

Field3: Property value.

`ndd`
Tunables that cannot be configured by a zone have been configured using `ndd`. These tunables must be configured from the global zone.

Field1: File or script setting the tunable.

Field2: Driver being tuned.

Field3: Tunable parameter.

`dynaddr`
One or more dynamically assigned IP addresses are configured on a network interface. These addresses are not supported with shared-IP zones. These IP addresses could change as a result of MAC address changes. You may need to modify this system's address information on the DHCP server and on the DNS, LDAP, or NIS name servers.

Field1 can be one of:

`dhcp`	Configured DHCP address. In this case, *Field2* is the name of the interface configured for DHCP.
`v6autoconf`	IPv6 stateless address configuration is enabled. In this case, *Field2* is the name of the interface with IPv6 auto configuration.

`rarp` (Solaris 10 source only) — Reverse ARP assigned address is enabled. In this case, *Field2* is the name of the interface with reverse ARP enabled.

`patch` (Solaris 10 source with Solaris 11+ target only)
A patch is required before `p2v` into a non-global zone.

Field1: The patch required.

`physif` (Solaris 10 targets only)
A physical interface exists on the source system that will have to be replaced with a dedicated physical or VLAN interface on the destination system if migrating to an exclusive-IP zone.

Field1: Name of the interface on the source system.

`sched`
The system is configured with a default scheduling class. The default scheduling class of a non-global zone can be configured using the `zonecfg set scheduler` property. This will be provided in the `-c` output.

Field1: The configured default scheduling class.

`sharedip` (Solaris 10 targets only)
If migrating to a shared-IP zone, the following networking features will need to be configured from the global zone on behalf of the zone.

Field1 can be one of:

`ipmpgroup` — An IPMP group is configured. If IPMP is required, it must be configured from the global zone. In this case, *Field2* is the IPMP group name.

`vni` — A virtual network interface is configured. These must be configured from the global zone. In this case, *Field2* is the VNI interface name.

`v4forwarding`
`v6forwarding` — IP forwarding (v4 or v6) is configured on an interface. In this case, *Field2* is the interface with IP forwarding configured.

`staticroute` — Static routes are configured. Static routes must be configured from the global zone.

`exclusiveonly` (Solaris 10 targets only)
A networking feature is configured that is not supported for use with shared-IP zones. The feature will work without modification in exclusive-IP zones.

Field1:

`iptun` — A IPv4, IPv6, or 6to4 tunnel interface has been plumbed.

Field2: Name of the tunnel interface.

`sharedonly`
A networking feature is configured that is not supported in an exclusive-IP zone. When migrating to a shared-IP zone, the feature must be configured in the global zone to support communication.

Field1:

`cgtp` A Carrier Grade Transport Protocol interface has been plumbed.

Field2: Name of the CGTP interface.

`netdevalloc`
A networking feature requires its underlying device be allocated to the zone with the zonecfg(1M) add device command. This feature is not supported with shared-IP zones.

Field1: Can be :

`ppp` Point-to-Point Protocol (PPP). PPP configuration files exist under `/etc/ppp`. The underlying device that needs to be allocated to the zone is either a serial port or, in the case of `pppoe`, an Ethernet physical or VNIC interface.

`svcexlip` (Solaris 10 targets only)
A service is enabled that will require an exclusive-IP zone.

Field1: Name of the service FMRI.

`svcpriv`
A service is enabled that will require additional privileges be added to the zone using the `zonecfg(1M)` `limitpriv` property.

Field1: FMRI of the service.

Field2: List of the privileges required by the service.

`svm`
A Solaris Volume Manager metadevice is configured. Metadevices must be configured in the global zone, and made available to the non-global zone using `zonecfg(1M)` `add device`, `add fs`, or `add dataset`.

Field1: Name of the metadevice.

`ramdisk`
A ramdisk device is configured. A zone cannot configure ramdisk devices.

Field1: Ramdisk device path.

`vfstab`
A filesystem mount is configured by means of `/etc/vfstab`. The filesystem must be migrated to the target global zone and made available to the non-global zone.

Field1: Device being mounted.

Field2: Mountpoint.

`zpool`
The system has additional zpools configured. These zpools must be migrated to the target global zone, and made available to the zone using `zonecfg add dataset` or `zonecfg add fs`.

Field1: Name of the pool.

`privexclip` (Solaris 10 targets only)
A process used a privilege that requires and exclusive-IP stack. See `zonecfg(1M)` for a description of the `ip-type` property.

Field1: Name of the process.

Field2: Privilege used.

`devexclip` (Solaris 10 targets only)
A process opened a device that requires an exclusive IP stack. See `zonecfg(1M)` for a description of the `ip-type` property.

Field1: Name of the process.

Field2: Name of the device.

`privoptional`
A process used a privilege that requires additional privilege be added to the target non-global zone. See `zonecfg(1M)` for a description of the `limitpriv` property.

Field1: Name of the process.

Field2: Privilege used.

`devoptional`
A process opened a device that is not available in a zone by default. See `zonecfg(1M)` for a description of the `add device` resource.

Field1: Name of the process.

Field2: Path of the device.

`syscallpriv` (generated by `-s` and `-f`)
A binary makes a system or library call that might require additional privilege be added to the target non-global zone. See `zonecfg(1M)` for a description of the `limitpriv` property. See the non-parseable output for details concerning the system or library call.

Field1: Path of the binary

Field2: Name of the system call.

`syscallexclip` (generated by `-s` and `-f`)
A binary makes a system or library call that might require an exclusive-ip stack. See `zonecfg(1M)` for a description of the `ip-type` property. See the non-parseable output for details concerning the system or library call.

Field1: Path of the binary

Field2: Name of the system call.

Examples

EXAMPLE 1 Performing Static Binary Analysis

The following command performs static analysis on all ELF binaries in two application directory trees:

```
# zonep2vchk -s /opt/myapplication,/usr/local
```

EXAMPLE 2 Generating a Template for the Target Zone

The following command will generate a template zone configuration for Solaris 11 when run on a Solaris 10 global zone.

```
# zonep2vchk T S11 -c
```

EXAMPLE 3 Analyzing Running Applications for a Period

The following command will analyze the process named `myapplication` for one hour and report any activity that might not function in a zone.

```
# zonep2vchk -s 1h -e myapplication
```

EXAMPLE 4 Performing Basic Checks

The following command will analyze the global zone for configuration and Solaris features in use that might not function in a zone. Each discovered issue will be reported as a single line of parseable output.

```
# zonep2vchk  -bP
```

Exit Status

The following exit values are returned:

`0`
Successful completion, no issues detected.

`1`
An internal error occurred.

`2`
Invalid usage.

`3`
One or more issues were detected.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/zones
Interface Stability	See below

Command invocation and parseable output is Committed. Human readable output (default output) is Uncommitted.

See Also dladm(1M), mipagent(1M), zoneadm(1M), zonecfg(1M), attributes(5), solaris(5), solaris10(5), zones(5)

Notes The static (`-s` and `-f`) checks make use of the elfdump(1) utility, which is delivered by the following package:

Solaris 11 `developer/base-developer-utilities`

Solaris 10 `SUNWbtool`

The runtime (`-r`) checks make use of the dtrace(1M) utility, which is delivered by the following package:

Solaris 11 `system/dtrace`

Solaris 10 `SUNWdtrc`

Name zonestatd – zones monitoring daemon

Synopsis

```
/usr/lib/zones/zonestatd
```

Description `zonestatd` is a system daemon that is started during system boot. It monitors the utilization of system resources by zones, as well as zone and system configuration information such as `psrset` `psets`, pool `psets`, and resource control settings.

This daemon is started automatically by the zone management software and should not be invoked directly. It does not constitute a programming interface; it is classified as a private interface.

Security The `zonestat` service in the global zone must be online for the `zonestat` service in each non-global zone (NGZ) to function properly. The `zonestat` service in each NGZ does not directly read system configuration and utilization data, but rather reads from the `zonestat` service on the global zone.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/zones
Interface Stability	Private

See Also `prctl(1)`, `svcs(1)`, `zonestat(1)`, `acctadm(1M)`, `pooladm(1M)`, `poolcfg(1M)`, `rcapadm(1M)`, `svcadm(1M)`, `attributes(5)`, `smf(5)`, `zones(5)`

Notes The `zonestat` service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/system/zones-monitoring:default
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. The service's status can be queried using the `svcs(1)` command.

The `zonestat` service has the following SMF configuration property:

```
config/sample_interval
```

This property sets the `zonestatd` sample interval. This is the interval used by the zones monitoring daemon, `zonestatd` to sample resource utilization. This is also the interval used to determine configuration changes such as processor set changes, resource control changes, and zone state changes.

The default interval is 5 seconds.

The zonestat service makes use of extended accounting facility. If not already enabled, it enables the tracking of process accounting resources, and configures a process accounting file. The zonestat service will roll the process accounting log at its configured interval (see zonestat(1)).

If extended process accounting is enabled externally, the zonestat service will use the process accounting log as configured. It will not roll the accounting log, but will operate correctly if the accounting log is rolled externally.

Name zpool – configures ZFS storage pools

Synopsis

```
zpool [-?]

zpool help command | help | property property-name

zpool help -l properties

zpool add [-f] [-n [-l]] pool vdev ...

zpool attach [-f] pool device new_device

zpool clear [-nF [-f]] pool [device]

zpool create [-f] [-n [-l]] [-B] [-o property=value] ...
     [-O file-system-property=value] ... [-m mountpoint]
     [-R root] pool vdev ...

zpool destroy [-f] pool

zpool detach pool device

zpool export [-f] pool ...

zpool get all | property[,...] pool ...

zpool history [-il] [pool] ...

zpool import [-d path ...] [-D]

zpool import [-d path ... | -c cachefile][-F [-n <pool | id>

zpool import [-o mntopts] [-o property=value] ... [-d path ... |
     -c cachefile] [-D] [-f] [-m] [-N] [-R root] [-F [-n [-l]]] -a
     pool | id [newpool]

zpool import [-o mntopts] [-o property=value] ... [-d path ... |
     -c cachefile] [-D] [-f] [-m] [-N] [-R root] [-F [-n [-l]]]
     pool | id [newpool]

zpool iostat [-T d|u ] [-v [-l]] [pool] ... [interval[count]]

zpool list [-H] [-o property[,...]] [-T d|u ] [pool] ... [interval[count]]

zpool offline [-t] pool device ...

zpool online [-e] pool device ...

zpool remove pool device ...

zpool replace [-f] pool device [new_device]

zpool scrub [-s] pool ...

zpool set property=value pool

zpool split [-n [-l]] [-R altroot]  [-o mntopts] [-o property=value] pool
     newpool [device ...]

zpool status [-l] [-v] [-x] [-T d|u ] [pool] ... [interval[count]]
```

```
zpool upgrade

zpool upgrade -v

zpool upgrade [-V version] -a | pool ...
```

Description The `zpool` command configures ZFS storage pools. A storage pool is a collection of devices that provides physical storage and data replication for ZFS datasets.

All datasets within a storage pool share the same space. See `zfs(1M)` for information on managing datasets.

Virtual Devices (vdevs) A *virtual device* describes a single device or a collection of devices organized according to certain performance and fault characteristics. The following virtual devices are supported:

`disk`
A block device, typically located under `/dev/dsk`. ZFS can use individual slices or partitions, though the recommended mode of operation is to use whole disks. A disk can be specified by a full path, or it can be a shorthand name (the relative portion of the path under `/dev/dsk`). A whole disk can be specified by omitting the slice or partition designation. Alternatively, whole disks can be specified using the `/dev/chassis/.../disk` path that describes the disk's current location. When given a whole disk, ZFS automatically labels the disk, if necessary.

`file`
A regular file. The use of files as a backing store is strongly discouraged. It is designed primarily for experimental purposes, as the fault tolerance of a file is only as good as the file system of which it is a part. A file must be specified by a full path.

`mirror`
A mirror of two or more devices. Data is replicated in an identical fashion across all components of a mirror. A mirror with *N* disks of size *X* can hold *X* bytes and can withstand (*N-1*) devices failing before data integrity is compromised.

`raidz`
`raidz1`
`raidz2`
`raidz3`
A variation on RAID-5 that allows for better distribution of parity and eliminates the "RAID-5 write hole" (in which data and parity become inconsistent after a power loss). Data and parity is striped across all disks within a raidz group.

A raidz group can have single-, double- , or triple parity, meaning that the raidz group can sustain one, two, or three failures, respectively, without losing any data. The raidz1 `vdev` type specifies a single-parity raidz group; the raidz2 `vdev` type specifies a double-parity raidz group; and the raidz3 `vdev` type specifies a triple-parity raidz group. The raidz `vdev` type is an alias for raidz1.

A raidz group with *N* disks of size *X* with *P* parity disks can hold approximately (*N-P*)**X* bytes and can withstand *P* device(s) failing before data integrity is compromised. The minimum number of devices in a raidz group is one more than the number of parity disks. The recommended number is between 3 and 9 to help increase performance.

`spare`
A special pseudo-vdev which keeps track of available hot spares for a pool. For more information, see the "Hot Spares" section.

`log`
A separate-intent log device. If more than one log device is specified, then writes are load-balanced between devices. Log devices can be mirrored. However, raidz `vdev` types are not supported for the intent log. For more information, see the "Intent Log" section.

`cache`
A device used to cache storage pool data. A cache device cannot be configured as a mirror or `raidz` group. For more information, see the "Cache Devices" section.

Virtual devices cannot be nested, so a mirror or raidz virtual device can only contain files or disks. Mirrors of mirrors (or other combinations) are not allowed.

A pool can have any number of virtual devices at the top of the configuration (known as *root vdevs*). Data is dynamically distributed across all top-level devices to balance data among devices. As new virtual devices are added, ZFS automatically places data on the newly available devices.

Virtual devices are specified one at a time on the command line, separated by whitespace. The keywords `mirror` and `raidz` are used to distinguish where a group ends and another begins. For example, the following creates two root vdevs, each a mirror of two disks:

```
# zpool create mypool mirror c0t0d0 c0t1d0 mirror c1t0d0 c1t1d0
```

Alternatively, the following command could be used:

```
# zpool create tank \
mirror \
    /dev/chassis/RACK29.U01-04/DISK_00/disk \
    /dev/chassis/RACK29.U05-08/DISK_00/disk \
mirror \
    /dev/chassis/RACK29.U01-04/DISK_01/disk \
    /dev/chassis/RACK29.U05-08/DISK_01/disk
```

Pool or Device Failure and Recovery

ZFS supports a rich set of mechanisms for handling device failure and data corruption. All metadata and data is checksummed, and ZFS automatically repairs bad data from a good copy when corruption is detected.

In order to take advantage of these features, a pool must make use of some form of redundancy, using either mirrored or raidz groups. While ZFS supports running in a non-redundant configuration, where each root vdev is simply a disk or file, this is strongly discouraged. A single case of bit corruption can render some or all of your data unavailable.

A pool's health status is described by one of four states:

`DEGRADED`
A pool with one or more failed devices, but the data is still available due to a redundant configuration.

`ONLINE`
A pool that has all devices operating normally.

`SUSPENDED`
A pool that is waiting for device connectivity to be restored. A suspended pool remains in the wait state until the device issue is resolved.

`UNAVAIL`
A pool with corrupted metadata, or one or more unavailable devices and insufficient replicas to continue functioning.

The health of the top-level vdev, such as mirror or raidz device, is potentially impacted by the state of its associated vdevs, or component devices. A top-level vdev or component device is in one of the following states:

`DEGRADED`
One or more top-level vdevs is in the degraded state because one or more component devices are offline. Sufficient replicas exist to continue functioning.

One or more component devices is in the degraded or faulted state, but sufficient replicas exist to continue functioning. The underlying conditions are as follows:

- The number of checksum errors exceeds acceptable levels and the device is degraded as an indication that something may be wrong. ZFS continues to use the device as necessary.
- The number of I/O errors exceeds acceptable levels. The device could not be marked as faulted because there are insufficient replicas to continue functioning.

`OFFLINE`
The device was explicitly taken offline by the `zpool offline` command.

`ONLINE`
The device is online and functioning.

`REMOVED`
The device was physically removed while the system was running. Device removal detection is hardware-dependent and may not be supported on all platforms.

`UNAVAIL`
The device could not be opened. If a pool is imported when a device was unavailable, then the device will be identified by a unique identifier instead of its path since the path was never correct in the first place.

If a device is removed and later reattached to the system, ZFS attempts to put the device online automatically. Device attach detection is hardware-dependent and might not be supported on all platforms.

Hot Spares

ZFS allows devices to be associated with pools as *hot spares*. These devices are not actively used in the pool, but when an active device fails, it is automatically replaced by a hot spare. To create a pool with hot spares, specify a `spare` vdev with any number of devices. For example,

```
# zpool create pool mirror c0d0 c1d0 spare c2d0 c3d0
```

Spares can be added with the `zpool add` command and removed with the `zpool remove` command. Once a spare replacement is initiated, a new spare vdev is created within the configuration that will remain there until the original device is replaced. At this point, the hot spare becomes available again if another device fails.

An in-progress spare replacement can be cancelled by detaching the hot spare. If the original faulted device is detached, then the hot spare assumes its place in the configuration, and is removed from the spare list of all active pools.

If the original failed device is physically replaced, brought back online, or the errors are cleared, either through an FMA event or by using the `zpool online` or `zpool clear` commands, and the state of the original device becomes healthy, the `INUSE` spare device will become `AVAIL` again.

Spares cannot replace log devices.

Intent Log

The ZFS Intent Log (ZIL) satisfies POSIX requirements for synchronous transactions. For instance, databases often require their transactions to be on stable storage devices when returning from a system call. NFS and other applications can also use `fsync()` to ensure data stability. By default, the intent log is allocated from blocks within the main pool. However, it might be possible to get better performance using separate intent log devices such as NVRAM or a dedicated disk. For example:

```
# zpool create pool c0d0 c1d0 log c2d0
```

Multiple log devices can also be specified, and they can be mirrored. See the EXAMPLES section for an example of mirroring multiple log devices.

Log devices can be added, replaced, attached, detached, and imported, and exported as part of the larger pool. Mirrored log devices can be removed by specifying the top-level mirror for the log.

Cache Devices

Devices can be added to a storage pool as *cache devices*. These devices provide an additional layer of caching between main memory and disk. For read-heavy workloads, where the working set size is much larger than what can be cached in main memory, using cache devices allow much more of this working set to be served from low latency media. Using cache devices provides the greatest performance improvement for random read-workloads of mostly static content.

To create a pool with cache devices, specify a `cache` *vdev* with any number of devices. For example:

```
# zpool create pool c0d0 c1d0 cache c2d0 c3d0
```

Cache devices cannot be mirrored or part of a raidz configuration. If a read error is encountered on a cache device, that read I/O is reissued to the original storage pool device, which might be part of a mirrored or raidz configuration.

The content of the cache devices is considered volatile, as is the case with other system caches.

Processes

Each imported pool has an associated process, named `zpool-`*poolname*. The threads in this process are the pool's I/O processing threads, which handle the compression, checksumming, and other tasks for all I/O associated with the pool. This process exists to provides visibility into the CPU utilization of the system's storage pools. The existence of this process is an unstable interface.

Properties

Each pool has several properties associated with it. Some properties are read-only statistics while others are configurable and change the behavior of the pool. The following are read-only properties:

`allocated`
Amount of storage space within the pool that has been physically allocated. This property can also be referred to by its shortened column name, `alloc`.

`capacity`
Percentage of pool space used. This property can also be referred to by its shortened column name, `cap`.

`dedupratio`
The deduplication ratio specified for a pool, expressed as a multiplier. This value is expressed as a single decimal number. For example, a `dedupratio` value of `1.76` indicates that 1.76 units of data were stored but only 1 unit of disk space was actually consumed. This property can also be referred to by its shortened column name, `dedup`.

Deduplication can be enabled as follows:

 # zfs set dedup=on *pool/dataset*

The default value is `off`.

See `zfs(1M)` for a description of the deduplication feature.

`free`
Number of blocks within the pool that are not allocated.

`guid`
A unique identifier for the pool.

`health`
The current health of the pool. Health can be `ONLINE`, `DEGRADED`, `UNAVAIL`, or `SUSPENDED`.

`size`
Total size of the storage pool.

These space usage properties report actual physical space available to the storage pool. The physical space can be different from the total amount of space that any contained datasets can actually use. The amount of space used in a raidz configuration depends on the characteristics of the data being written. In addition, ZFS reserves some space for internal accounting that the `zfs(1M)` command takes into account, but the `zpool` command does not. For non-full pools of a reasonable size, these effects should be invisible. For small pools, or pools that are close to being completely full, these discrepancies may become more noticeable.

The following property can be set at creation time and import time:

`altroot`
Alternate root directory. If set, this directory is prepended to any mount points within the pool. This can be used when examining an unknown pool where the mount points cannot be trusted, or in an alternate boot environment, where the typical paths are not valid. `altroot` is not a persistent property. It is valid only while the system is up. Setting `altroot` defaults to using `cachefile=none`, though this may be overridden using an explicit setting.

The following property can be set at import time:

`readonly=on | off`
Controls whether the pool can be modified. When enabled, any synchronous data that exists only in the intent log is not accessible until the pool is imported in read-write mode.

Importing a pool in read-only mode has the following limitations:

- Attempts to set additional pool properties during the import are ignored.
- All file system mounts are converted to include the read-only (`ro`) mount option.

A pool that has been imported in read-only mode can be restored to read-write mode by exporting and importing the pool.

The following properties can be set at creation time and import time, and later changed with the `zpool set` command:

`autoexpand=on | off`
Controls automatic pool expansion when the underlying LUN is grown. If set to `on`, the pool will be resized according to the size of the expanded device. If the device is part of a mirror or `raidz` then all devices within that mirror/`raidz` group must be expanded before the new space is made available to the pool. The default behavior is `off`. This property can also be referred to by its shortened column name, `expand`.

`autoreplace=on | off`
Controls automatic device replacement. If set to `off`, device replacement must be initiated by the administrator by using the `zpool replace` command. If set to `on`, any new device, found in the same physical location as a device that previously belonged to the pool, is

automatically formatted and replaced. The default behavior is `off`. This property can also be referred to by its shortened column name, `replace`.

`bootfs=`*pool*`/`*dataset*
Identifies the default bootable dataset for the root pool. This property is expected to be set mainly by the installation and upgrade programs.

`cachefile=`*path* | `none`
Controls the location of where the pool configuration is cached. Discovering all pools on system startup requires a cached copy of the configuration data that is stored on the root file system. All pools in this cache are automatically imported when the system boots. Some environments, such as install and clustering, need to cache this information in a different location so that pools are not automatically imported. Setting this property caches the pool configuration in a different location that can later be imported with `zpool import -c`. Setting it to the special value `none` creates a temporary pool that is never cached, and the special value `''` (empty string) uses the default location.

Multiple pools can share the same cache file. Because the kernel destroys and recreates this file when pools are added and removed, care should be taken when attempting to access this file. When the last pool using a `cachefile` is exported or destroyed, the file is removed.

`dedupditto=`*number*
Sets a threshold for number of copies. If the reference count for a deduplicated block goes above this threshold, another ditto copy of the block is stored automatically. The default value is `0`.

`delegation=on` | `off`
Controls whether a non-privileged user is granted access based on the dataset permissions defined on the dataset. The default value is `on`. See `zfs(1M)` for more information on ZFS delegated administration.

`failmode=wait` | `continue` | `panic`
Controls the system behavior in the event of catastrophic pool failure. This condition is typically a result of a loss of connectivity to the underlying storage device(s) or a failure of all devices within the pool. The behavior of such an event is determined as follows:

`wait`
Blocks all I/O access to the pool until the device connectivity is recovered and the errors are cleared. A pool remains in the wait state until the device issue is resolved. This is the default behavior.

`continue`
Returns EIO to any new write I/O requests but allows reads to any of the remaining healthy devices. Any write requests that have yet to be committed to disk would be blocked.

`panic`
Prints out a message to the console and generates a system crash dump.

`listshares`=on | off
Controls whether share information in this pool is displayed with the `zfs list` command. The default value is `off`.

`listsnaps`=on | off
Controls whether information about snapshots associated with this pool is output when `zfs list` is run without the `-t` option. The default value is `off`.

`version`=*version*
The current on-disk version of the pool. This can be increased, but never decreased. The preferred method of updating pools is with the `zpool upgrade` command, though this property can be used when a specific version is needed for backwards compatibility. This property can be any number between 1 and the current version reported by `zpool upgrade -v`.

Subcommands

All subcommands that modify state are logged persistently to the pool in their original form.

The `zpool` command provides subcommands to create and destroy storage pools, add capacity to storage pools, and provide information about the storage pools. The following subcommands are supported:

`zpool -?`
Displays a help message.

`zpool help` *command* | `help` | *property property-name*
Displays `zpool` command usage. You can display help for a specific command or property. If you display help for a specific command or property, the command syntax or available property values are displayed. Using `zpool help` without any arguments displays a complete list of `zpool` commands.

`zpool help -l properties`
Displays `zpool` property information, including whether the property value is editable and their possible values. If you display help for a specific subcommand or property, the command syntax or property value is displayed. Using `zpool help` without any arguments displays a complete list of `zpool` subcommands.

`zpool add` [`-f`] [`-n` [`-l`]]*pool vdev* ...
Adds the specified virtual devices to the given pool. The *vdev* specification is described in the "Virtual Devices" section. The behavior of the `-f` option, and the device checks performed are described in the `zpool create` subcommand.

`-f`
Forces use of vdevs, even if they appear in use or specify a conflicting replication level. Not all devices can be overridden in this manner.

`-n`
Displays the configuration that would be used without actually adding the vdevs. The actual pool creation can still fail due to insufficient privileges or device sharing.

`-l`
If possible, have `-n` display the configuration in current `/dev/chassis` location form.

Do not add a disk that is currently configured as a quorum device to a ZFS storage pool. After a disk is in the pool, that disk can then be configured as a quorum device.

`zpool attach` [`-f`] *pool device new_device*
Attaches *new_device* to an existing `zpool` device. The existing device cannot be part of a raidz configuration. If *device* is not currently part of a mirrored configuration, *device* automatically transforms into a two-way mirror of *device* and *new_device*. If *device* is part of a two-way mirror, attaching *new_device* creates a three-way mirror, and so on. In either case, *new_device* begins to resilver immediately.

`-f`
Forces use of *new_device*, even if its appears to be in use. Not all devices can be overridden in this manner.

`zpool clear` [`-nF` [`-f`]] *pool* [*device*] ...
Clears device errors in a pool. If no arguments are specified, all device errors within the pool are cleared. If one or more devices is specified, only those errors associated with the specified device or devices are cleared.

`-F`
Initiates recovery mode for an unopenable pool. Attempts to discard the last few transactions in the pool to return it to an openable state. Not all damaged pools can be recovered by using this option. If successful, the data from the discarded transactions is irretrievably lost.

`-n`
Used in combination with the `-F` flag. Check whether discarding transactions would make the pool openable, but do not actually discard any transactions.

`-f`
This is a special pool recovery option that can be used if the `fmadm acquit` or `fmadm repair` commands fail to clear a pool's faults. If the system reboots, FMA replays the pool faults so you will need to resolve the FMA faults after the pool is recovered.

`zpool create` [`-f`] [`-n` [`-l`]] [`-B`] [`-o` *property=value*] ... [`-O` *file-system-property=value*] ... [`-m` *mountpoint*] [`-R` *root*] *pool vdev* ...
Creates a new storage pool containing the virtual devices specified on the command line. The pool name must begin with a letter, and can contain alphanumeric characters, as well as underscore (_), dash (-), colon (:), space ( ), and period (.). The pool names `mirror`, `raidz`, `spare`, and `log` are reserved, as are names beginning with the pattern `c[0-9]`. The vdev specification is described in the "Virtual Devices" section.

The command verifies that each device specified is accessible and not currently in use by another subsystem. There are some uses, such as being currently mounted, or specified as the dedicated dump device, that prevents a device from ever being used by ZFS. Other uses, such as having a preexisting UFS file system, can be overridden with the `-f` option.

The command also checks that the replication strategy for the pool is consistent. An attempt to combine redundant and non-redundant storage in a single pool, or to mix disks and files, results in an error unless `-f` is specified. The use of differently sized devices within a single raidz or mirror group is also flagged as an error unless `-f` is specified.

Unless the `-R` option is specified, the default mount point is */pool*. The mount point must not exist or must be empty, or else the root dataset cannot be mounted. This can be overridden with the `-m` option.

`-B`
When operating on a whole disk device, creates the boot partition, if one is required to boot from EFI (GPT) labeled disks on the platform. The `-B` option has no effect on devices that are not whole disks.

`-f`
Forces use of vdevs, even if they appear in use or specify a conflicting replication level. Not all devices can be overridden in this manner.

`-l`
If possible, have `-n` display the configuration in current `/dev/chassis` location form.

`-n`
Displays the configuration that would be used without actually creating the pool. The actual pool creation can still fail due to insufficient privileges or if a device is currently in use.

`-o` *property=value* [`-o` *property=value*] ...
Sets the given pool properties. See the "Properties" section for a list of valid properties that can be set.

`-O` *file-system-property=value*
[`-O` *file-system-property=value*] ...
Sets the given properties for the pool's top-level file system. See the "Properties" section of `zfs(1M)` for a list of valid properties that can be set.

`-R` *root*
Equivalent to `-o cachefile=none,altroot=`*root*.

`-m` *mountpoint*
Sets the mount point for the pool's top-level file system. The default mount point is */pool* or `altroot`*/pool* if `altroot` is specified. The mount point must be an absolute path, `legacy`, or `none`. For more information on dataset mount points, see `zfs(1M)`.

`zpool destroy` [`-f`] *pool*
Destroys the given pool, freeing up any devices for other use. This command tries to unmount any active datasets before destroying the pool.

`-f`
Forces any active datasets contained within the pool to be unmounted.

`zpool detach` *pool device*
Detaches a *device* or a spare from a mirrored storage pool. A spare can also be detached from a RAID-Z storage pool if an existing device was physically replaced. Or, you can detach an existing device in a RAID-Z storage pool if it was replaced by a spare. The operation is refused if there are no other valid replicas of the data.

`zpool export` [`-f`] *pool* ...
Exports the given pools from the system. All devices are marked as exported, but are still considered in use by other subsystems. The devices can be moved between systems (even those of different endianness) and imported as long as a sufficient number of devices are present.

Before exporting the pool, all datasets within the pool are unmounted.

For pools to be portable, you must give the `zpool` command whole disks, not just slices, so that ZFS can label the disks with portable EFI labels. Otherwise, disk drivers on platforms of different endianness will not recognize the disks.

`-f`
Forcefully unmount all datasets, using the `unmount -f` command.

This command will forcefully export the pool.

`zpool get` all | *property*[,...] *pool* ...
Retrieves the given list of properties (or all properties if `all` is used) for the specified storage pool(s). These properties are displayed with the following fields:

```
name          Name of storage pool
 property      Property name
 value         Property value
 source        Property source, either 'default' or 'local'.
```

See the "Properties" section for more information on the available pool properties.

`zpool history` [`-il`] [*pool*] ...
Displays the command history of the specified pools or all pools if no pool is specified.

`-i`
Displays internally logged ZFS events in addition to user initiated events.

`-l`
Displays log records in long format, which in addition to standard format includes, the user name, the hostname, and the zone in which the operation was performed.

`zpool import` [`-d` *path* ...] [`-D`]
`zpool import` [`-d` *path* ... | `-c` *cachefile*] [`-F` [`-n`]]*pool* | *id*
Lists pools available to import. If the `-d` option is not specified, this command searches for devices in `/dev/dsk`. The `-d` option can be specified multiple times, and all directories and device paths are searched. If the device appears to be part of an exported pool, this command displays a summary of the pool with the name of the pool, a numeric identifier,

as well as the *vdev* layout and current health of the device for each device or file. Pools that were previously destroyed with the `zpool destroy` command, are not listed unless the `-D` option is specified.

The numeric identifier is unique, and can be used instead of the pool name when multiple exported pools of the same name are available.

`-c` *cachefile*
Reads configuration from the given `cachefile` that was created with the "`cachefile`" pool property. This `cachefile` is used instead of searching for devices.

`-d` *path*
Searches for devices or files in *path*, where *path* where *path* can be a directory or a device path. The `-d` option can be specified multiple times.

`-D`
Lists destroyed pools only.

`zpool import` [`-o` *mntopts*] [`-o` *property=value*] ... [`-d` *path* ... | `-c` *cachefile*] [`-D`] [`-f`] [`-m`] [`-N`] [`-R` *root*] [`-F` [`-n` [`-l`]]] `-a`
Imports all pools found in the search directories or device paths. Identical to the previous command, except that all pools with a sufficient number of devices available are imported. Pools that were previously destroyed with the `zpool destroy` command, are not imported unless the `-D` option is specified.

`-o` *mntopts*
Comma-separated list of mount options to use when mounting datasets within the pool. See `zfs(1M)` for a description of dataset properties and mount options.

`-o` *property=value*
Sets the specified property on the imported pool. See the "Properties" section for more information on the available pool properties.

`-c` *cachefile*
Reads configuration from the given `cachefile` that was created with the "`cachefile`" pool property. This `cachefile` is used instead of searching for devices.

`-d` *path*
Searches for devices or files in *path*. The `-d` option can be specified multiple times. This option is incompatible with the `-c` option.

`-D`
Imports destroyed pools only. The `-f` option is also required.

`-f`
Forces import, even if the pool appears to be potentially active.

`-F`
Recovery mode for a non-importable pool. Attempt to return the pool to an importable state by discarding the last few transactions. Not all damaged pools can be recovered by

using this option. If successful, the data from the discarded transactions is irretrievably lost. This option is ignored if the pool is importable or already imported.

`-a`
Searches for and imports all pools found.

`-m`
Allows a pool to import when a log device is missing.

`-R` *root*
Sets the `cachefile` property to `none` and the *altroot* property to *root*.

`-N`
Imports the pool without mounting any file systems.

`-n`
Used with the `-F` recovery option. Determines whether a non-importable pool can be made importable again, but does not actually perform the pool recovery. For more details about pool recovery mode, see the `-F` option, above.

`-l`
If possible, have `-n` display information in current `/dev/chassis` location form.

`zpool import` [`-o` *mntopts*] [`-o` *property=value*] ... [`-d` *path* ... | `-c` *cachefile*] [`-D`] [`-f`] [`-m`] [`-N`] [`-R` *root*] [`-F` [`-n` [`-l`]]] *pool* | *id* [*newpool*]
Imports a specific pool. A pool can be identified by its name or the numeric identifier. If *newpool* is specified, the pool is imported using the persistent name *newpool*. Otherwise, it is imported with the same name as its exported name. Do not import a root pool with a new name. Otherwise, the system might not boot.

If a device is removed from a system without running `zpool export` first, the device appears as potentially active. It cannot be determined if this was a failed export, or whether the device is really in use from another host. To import a pool in this state, the `-f` option is required.

`-o` *mntopts*
Comma-separated list of mount options to use when mounting datasets within the pool. See `zfs(1M)` for a description of dataset properties and mount options.

`-o` *property=value*
Sets the specified property on the imported pool. See the “Properties” section for more information on the available pool properties.

`-c` *cachefile*
Reads configuration from the given `cachefile` that was created with the `cachefile` pool property. This `cachefile` is used instead of searching for devices.

`-d` *path*
Searches for devices or files in *path*. The `-d` option can be specified multiple times. This option is incompatible with the `-c` option.

`-D`
Imports destroyed pool. The `-f` option is also required.

`-f`
Forces import, even if the pool appears to be potentially active.

`-F`
Recovery mode for a non-importable pool. Attempt to return the pool to an importable state by discarding the last few transactions. Not all damaged pools can be recovered by using this option. If successful, the data from the discarded transactions is irretrievably lost. This option is ignored if the pool is importable or already imported.

`-R` *root*
Sets the `cachefile` property to `none` and the *altroot* property to *root*.

`-N`
Imports the pool without mounting any file systems.

`-n`
Used with the `-F` recovery option. Determines whether a non-importable pool can be made importable again, but does not actually perform the pool recovery. For more details about pool recovery mode, see the `-F` option, above.

`-l`
If possible, have `-n` display information in current `/dev/chassis` location form.

`-m`
Allows a pool to import when a log device is missing.

`zpool iostat` [`-T d|u`] [`-v` [`-l`]] [*pool*] ... [*interval*[*count*]]
Displays I/O statistics for the given pools. When given an interval, the statistics are printed every *interval* seconds until `Ctrl-C` is pressed. If no *pools* are specified, statistics for every pool in the system is shown. If *count* is specified, the command exits after *count* reports are printed.

`-T d|u`
Display a time stamp.

Specify `d` for standard date format. See `date(1)`. Specify `u` for a printed representation of the internal representation of time. See `time(2)`.

`-v`
Verbose statistics. Reports usage statistics for individual *vdevs* within the pool, in addition to the pool-wide statistics.

`-l`
If possible, have `-v` display `vdev` statistics in current `/dev/chassis` location form.

`zpool list` [`-H`] [`-o` *props*[,...]] [`-T d|u`] [*pool*] ...
Lists the given pools along with a health status and space usage. When given no arguments, all pools in the system are listed.

When given an interval, the status and space usage are displayed every *interval* seconds until Ctrl-C is entered. If *count* is specified, the command exits after *count* reports are displayed.

`-H`
Scripted mode. Do not display headers, and separate fields by a single tab instead of arbitrary space.

`-o` *props*
Comma-separated list of properties to display. See the "Properties" section for a list of valid properties. The default list is `name`, `size`, `allocated`, `free`, `capacity`, `health`, `altroot`.

`-T d|u`
Display a time stamp.

Specify `d` for standard date format. See `date(1)`. Specify `u` for a printed representation of the internal representation of time. See `time(2)`.

`zpool offline` [`-t`] *pool device* ...
Takes the specified physical device offline. While the *device* is offline, no attempt is made to read or write to the device.

This command is not applicable to spares or cache devices.

`-t`
Temporary. Upon reboot, the specified physical device reverts to its previous state.

`zpool online` [`-e`] *pool device*...
Brings the specified physical device online.

This command is not applicable to spares or cache devices.

`-e`
Expand the device to use all available space. If the device is part of a mirror or `raidz` then all devices must be expanded before the new space will become available to the pool.

`zpool remove` *pool device* ...
Removes the specified device from the pool. This command currently only supports removing hot spares, cache, and log devices. A mirrored log device can be removed by specifying the top-level mirror for the log. Non-log devices that are part of a mirrored configuration can be removed using the `zpool detach` command. Non-redundant and raidz devices cannot be removed from a pool.

`zpool replace` [`-f`] *pool old_device* [*new_device*]
Replaces *old_device* with *new_device*. This is equivalent to attaching *new_device*, waiting for it to resilver, and then detaching *old_device*.

The size of *new_device* must be greater than or equal to the minimum size of all the devices in a mirror or raidz configuration.

new_device is required if the pool is not redundant. If *new_device* is not specified, it defaults to *old_device*. This form of replacement is useful after an existing disk has failed and has been physically replaced. In this case, the new disk may have the same `/dev/dsk` path as the old device, even though it is actually a different disk. ZFS recognizes this.

In `zpool status` output, the *old_device* is shown under the word `replacing` with the string `/old` appended to it. Once the resilver completes, both the `replacing` and the *old_device* are automatically removed. If the new device fails before the resilver completes and a third device is installed in its place, then both failed devices will show up with `/old` appended, and the resilver starts over again. After the resilver completes, both `/old` devices are removed along with the word `replacing`.

`-f`
Forces use of *new_device*, even if its appears to be in use. Not all devices can be overridden in this manner.

`zpool scrub` [`-s`] *pool* ...
Begins a scrub. The scrub examines all data in the specified pools to verify that it checksums correctly. For replicated (mirror or raidz) devices, ZFS automatically repairs any damage discovered during the scrub. The `zpool status` command reports the progress of the scrub and summarizes the results of the scrub upon completion.

Scrubbing and resilvering are very similar operations. The difference is that resilvering only examines data that ZFS knows to be out of date (for example, when attaching a new device to a mirror or replacing an existing device), whereas scrubbing examines all data to discover silent errors due to hardware faults or disk failure.

Because scrubbing and resilvering are I/O-intensive operations, ZFS allows only one at a time. If a scrub is already in progress, a subsequent `zpool scrub` returns an error, with the advice to use `zpool scrub -s` to cancel the current scrub. If a resilver is in progress, ZFS does not allow a scrub to be started until the resilver completes.

`-s`
Stop scrubbing.

`zpool set` *property=value pool*
Sets the given property on the specified pool. See the “Properties” section for more information on what properties can be set and acceptable values.

`zpool split` [`-n` [`-l`]] [`-R` *altroot*] [`-o` *mntopts*] [`-o` *property=value*] *pool newpool* [*device* ...]
Splits off one disk from each mirrored top-level vdev in a pool and creates a new pool from the split-off disks. The original pool must be made up of one or more mirrors and must not be in the process of resilvering. The `split` subcommand chooses the last device in each mirror vdev unless overridden by a device specification on the command line.

When using a *device* argument, `split` includes the specified device(s) in a new pool and, should any devices remain unspecified, assigns the last device in each mirror vdev to that pool, as it does normally. If you are uncertain about the outcome of a `split` command, use the `-n` ("dry-run") option to ensure your command will have the effect you intend.

`-n`
Displays the configuration that would be created without actually splitting the pool. The actual pool split could still fail due to insufficient privileges or device status.

`-l`
If possible, have `-n` display the configuration in current `/dev/chassis` location form.

`-R` *altroot*
Automatically import the newly created pool after splitting, using the specified *altroot* parameter for the new pool's alternate root. See the `altroot` description in the "Properties" section, above.

`-o` *mntopts*
Comma-separated list of mount options to use when mounting datasets within the pool. See `zfs(1M)` for a description of dataset properties and mount options. Valid only in conjunction with the `-R` option.

`-o` *property=value*
Sets the specified property on the new pool. See the "Properties" section, above, for more information on the available pool properties.

`zpool status` [`-l`] [`-v`] [`-x`] [`-T d|u`] [*pool*] ... [*interval*[*count*]]
Displays the detailed health status for the given pools. If no *pool* is specified, then the status of each pool in the system is displayed. For more information on pool and device health, see the "Device Failure and Recovery" section.

When given an interval, the status and space usage are displayed every *interval* seconds until Ctrl-C is entered. If *count* is specified, the command exits after *count* reports are displayed.

If a scrub or resilver is in progress, this command reports the percentage done and the estimated time to completion. Both of these are only approximate, because the amount of data in the pool and the other workloads on the system can change.

`-l`
If possible, display vdev status in current `/dev/chassis` location form.

`-x`
Display status only for pools that are exhibiting errors or are otherwise unavailable.

`-v`
Displays verbose data error information, printing out a complete list of all data errors since the last complete pool scrub.

`-T d|u`
Display a time stamp.

Specify `d` for standard date format. See `date(1)`. Specify `u` for a printed representation of the internal representation of time. See `time(2)`.

`zpool upgrade`
Identifies a pool's on-disk version, which determines available pool features in the currently running software release. You can continue to use older pool versions, but some features might not be available. A pool can be upgraded by using the `zpool upgrade -a` command. You will not be able to access a pool of a later version on a system that runs an earlier software version.

`zpool upgrade -v`
Displays ZFS pool versions supported by the current software. The current ZFS pool versions and all previous supported versions are displayed, along with an explanation of the features provided with each version.

`zpool upgrade` [`-V` *version*] `-a` | *pool* ...
Upgrades the specified pool to the latest on-disk version. If this command reveals that a pool is out-of-date, the pool can subsequently be upgraded using the `zpool upgrade -a` command. A pool that is upgraded will not be accessible on a system that runs an earlier software release.

`-a`
Upgrades all pools.

`-V` *version*
Upgrade to the specified version, which must be higher than the current version. If the `-V` flag is not specified, the pool is upgraded to the most recent version.

Examples

EXAMPLE 1 Creating a RAID-Z Storage Pool

The following command creates a pool with a single raidz root *vdev* that consists of six disks.

```
# zpool create tank raidz c0t0d0 c0t1d0 c0t2d0 c0t3d0 c0t4d0 c0t5d0
```

EXAMPLE 2 Creating a Mirrored Storage Pool

The following command creates a pool with two mirrors, where each mirror contains two disks.

```
# zpool create tank mirror c0t0d0 c0t1d0 mirror c0t2d0 c0t3d0
```

Alternatively, whole disks can be specified using `/dev/chassis` paths describing the disk's current location.

```
# zpool create tank \
    mirror \
        /dev/chassis/RACK29.U01-04/DISK_00/disk \
        /dev/chassis/RACK29.U05-08/DISK_00/disk \
```

EXAMPLE 2 Creating a Mirrored Storage Pool *(Continued)*

```
    mirror \
        /dev/chassis/RACK29.U01-04/DISK_01/disk \
        /dev/chassis/RACK29.U05-08/DISK_01/disk
```

EXAMPLE 3 Adding a Mirror to a ZFS Storage Pool

The following command adds two mirrored disks to the pool `tank`, assuming the pool is already made up of two-way mirrors. The additional space is immediately available to any datasets within the pool.

```
# zpool add tank mirror c1t0d0 c1t1d0
```

EXAMPLE 4 Listing Available ZFS Storage Pools

The following command lists all available pools on the system.

```
# zpool list
NAME   SIZE  ALLOC  FREE  CAP  DEDUP  HEALTH  ALTROOT
pool   278G  4.19G  274G   1%  1.00x  ONLINE  -
rpool  278G  78.2G  200G  28%  1.00x  ONLINE  -
```

EXAMPLE 5 Listing All Properties for a Pool

The following command lists all the properties for a pool.

```
% zpool get all pool
NAME  PROPERTY       VALUE                SOURCE
pool  allocated      4.19G                -
pool  altroot        -                    default
pool  autoexpand     off                  default
pool  autoreplace    off                  default
pool  bootfs         -                    default
pool  cachefile      -                    default
pool  capacity       1%                   -
pool  dedupditto     0                    default
pool  dedupratio     1.00x                -
pool  delegation     on                   default
pool  failmode       wait                 default
pool  free           274G                 -
pool  guid           1907687796174423256  -
pool  health         ONLINE               -
pool  listshares     off                  local
pool  listsnapshots  off                  default
pool  readonly       off                  -
pool  size           278G                 -
pool  version        34                   default
```

EXAMPLE 6 Destroying a ZFS Storage Pool

The following command destroys the pool "*tank*" and any datasets contained within.

```
# zpool destroy -f tank
```

EXAMPLE 7 Exporting a ZFS Storage Pool

The following command exports the devices in pool *tank* so that they can be relocated or later imported.

```
# zpool export tank
```

EXAMPLE 8 Importing a ZFS Storage Pool

The following command displays available pools, and then imports the pool "tank" for use on the system.

The results from this command are similar to the following:

```
# zpool import
  pool: tank
    id: 7678868315469843843
 state: ONLINE
action: The pool can be imported using its name or numeric identifier.
config:

             tank  ONLINE
          mirror-0  ONLINE
            c1t2d0  ONLINE
            c1t3d0  ONLINE

# zpool import tank
```

EXAMPLE 9 Upgrading All ZFS Storage Pools to the Current Version

The following command upgrades all ZFS Storage pools to the current version of the software.

```
# zpool upgrade -a
This system is currently running ZFS pool version 22.

All pools are formatted using this version.
```

EXAMPLE 10 Managing Hot Spares

The following command creates a new pool with an available hot spare:

```
# zpool create tank mirror c0t0d0 c0t1d0 spare c0t2d0
```

If one of the disks were to fail, the pool would be reduced to the degraded state. The failed device can be replaced using the following command:

```
# zpool replace tank c0t0d0 c0t3d0
```

EXAMPLE 10 Managing Hot Spares *(Continued)*

After the device has been resilvered, the spare is automatically detached and is made available should another device fail. The hot spare can be permanently removed from the pool using the following command:

```
# zpool remove tank c0t2d0
```

EXAMPLE 11 Creating a ZFS Pool with Separate Mirrored Log Devices

The following command creates a ZFS storage pool consisting of two, two-way mirrors and mirrored log devices:

```
# zpool create pool mirror c0d0 c1d0 mirror c2d0 c3d0 log mirror \
   c4d0 c5d0
```

EXAMPLE 12 Adding Cache Devices to a ZFS Pool

The following command adds two disks for use as cache devices to a ZFS storage pool:

```
# zpool add pool cache c2d0 c3d0
```

Once added, the cache devices gradually fill with content from main memory. Depending on the size of your cache devices, it could take over an hour for them to fill. Capacity and reads can be monitored using the `iostat` option as follows:

```
# zpool iostat -v pool 5
```

EXAMPLE 13 Removing a Mirrored Log Device

Given the configuration shown immediately below, the following command removes the mirrored log device `mirror-2` in the pool `tank`.

```
   pool: tank
  state: ONLINE
  scrub: none requested
config:

         NAME        STATE     READ WRITE CKSUM
         tank        ONLINE       0     0     0
           mirror-0  ONLINE       0     0     0
             c6t0d0  ONLINE       0     0     0
             c6t1d0  ONLINE       0     0     0
           mirror-1  ONLINE       0     0     0
             c6t2d0  ONLINE       0     0     0
             c6t3d0  ONLINE       0     0     0
         logs
           mirror-2  ONLINE       0     0     0
             c4t0d0  ONLINE       0     0     0
             c4t1d0  ONLINE       0     0     0

# zpool remove tank mirror-2
```

EXAMPLE 14 Recovering a Faulted ZFS Pool

If a pool is faulted but recoverable, a message indicating this state is provided by `zpool status` if the pool was cached (see `cachefile` above), or as part of the error output from a failed `zpool import` of the pool.

Recover a cached pool with the `zpool clear` command:

```
# zpool clear -F data
Pool data returned to its state as of Thu Jun 07 10:50:35 2012.
Discarded approximately 29 seconds of transactions.
```

If the pool configuration was not cached, use `zpool import` with the recovery mode flag:

```
# zpool import -F data
Pool data returned to its state as of Thu Jun 07 10:50:35 2012.
Discarded approximately 29 seconds of transactions.
```

EXAMPLE 15 Importing a ZFS Pool with a Missing Log Device

The following examples illustrate attempts to import a pool with a missing log device. The `-m` option is used to complete the import operation.

Additional devices are known to be part of this pool, though their exact configuration cannot be determined.

```
# zpool import tank
The devices below are missing, use '-m' to import the pool anyway:
            c5t0d0 [log]

cannot import 'tank': one or more devices is currently unavailable

# zpool import -m tank
# zpool status tank
   pool: tank
  state: DEGRADED
status: One or more devices could not be opened.  Sufficient replicas
exist for
         the pool to continue functioning in a degraded state.
action: Attach the missing device and online it using 'zpool online'.
    see: http://www.support.oracle.com/msg/ZFS-8000-2Q
   scan: none requested
config:

         NAME                    STATE     READ WRITE CKSUM
         tank                    DEGRADED     0     0     0
           c7t0d0                ONLINE       0     0     0
         logs
           1693927398582730352   UNAVAIL      0     0     0  was
/dev/dsk/c5t0d0
```

EXAMPLE 15 Importing a ZFS Pool with a Missing Log Device *(Continued)*

```
errors: No known data errors
```

The following example shows how to import a pool with a missing *mirrored* log device.

```
# zpool import tank
The devices below are missing, use ?-m? to import the pool anyway:
mirror-1 [log]
c5t0d0
c5t1d0

# zpool import -m tank

# zpool status tank
   pool: tank
  state: DEGRADED
status: One or more devices could not be opened.  Sufficient replicas
exist for the pool to continue functioning in a degraded state.
action: Attach the missing device and online it using 'zpool online'.
    see: http://www.support.oracle.com/msg/ZFS-8000-2Q
    scan: none requested
config:

         NAME                      STATE     READ WRITE CKSUM
         tank                      DEGRADED     0     0     0
           c7t0d0                  ONLINE       0     0     0
         logs
           mirror-1                UNAVAIL      0     0     0
insufficient replicas
             46385995713041169     UNAVAIL      0     0     0  was
/dev/dsk/c5t0d0
             13821442324672734438  UNAVAIL      0     0     0  was
/dev/dsk/c5t1d0

errors: No known data errors
```

EXAMPLE 16 Importing a Pool By a Specific Path

The following command imports the pool `tank` by identifying the pool's specific device paths, `/dev/dsk/c9t9d9` and `/dev/dsk/c9t9d8`, in this example.

```
# zpool import -d /dev/dsk/c9t9d9s0 /dev/dsk/c9t9d8s0 tank
```

An existing limitation is that even though this pool is comprised of whole disks, the command must include the specific device's slice identifier.

Exit Status The following exit values are returned:

`0`
Successful completion.

`1`
An error occurred.

`2`
Invalid command line options were specified.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/file-system/zfs
Interface Stability	Committed

See Also `ps(1)`, `zfs(1M)`, `attributes(5)`, `SDC(7)`

Notes Each ZFS storage pool has an associated process, `zpool-`*poolname*, visible in such tools as `ps(1)`. A user has no interaction with these processes. See `SDC(7)`.

Name zstreamdump – filter data in zfs send stream

Synopsis

```
zstreamdump [-C] [-v]
```

Description The `zstreamdump` utility reads from the output of the `zfs send` command, then displays headers and some statistics from that output. See `zfs(1M)`.

Options The following options are supported:

`-C`
Suppress the validation of checksums.

`-v`
Verbose. Dump all headers, not only begin and end headers.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/file-system/zfs
Interface Stability	Uncommitted

See Also `zfs(1M)`, `attributes(5)`