Using Virtual Networks in Oracle® Solaris 11.1

Part No: E28992–01
October 2012

121203@25097

Contents

Preface 5

1 **Network Virtualization and Resource Management in Oracle Solaris** 7

Overview of Network Virtualization 7

Components of Network Virtualization 8

Who Should Implement Virtual Networks? 10

Commands for Configuring Virtualization Components 10

Overview of Network Resource Management 11

Datalink Properties for Resource Control 11

Network Resource Management by Using Flows 12

Commands for Network Resource Management 13

2 **Creating and Administering Virtual Networks in Oracle Solaris** 15

Configuring the Components of Network Virtualization 15

▼ How to Configure VNICs and Etherstubs 15

▼ How to Configure VNICs With VLAN IDs 17

Building Virtual Networks 18

▼ How to Configure a Zone for the Virtual Network 19

▼ How to Reconfigure a Zone to Use a VNIC 21

▼ How to Create a Private Virtual Network 23

Other Administrative Tasks for VNICs 25

Modifying the VLAN ID of a VNIC 26

Modifying VNIC MAC Addresses 26

Migrating VNICs 27

Displaying VNIC Information 29

▼ How to Delete a VNIC 29

3 Managing Network Resources in Oracle Solaris 31
Working With Clients, Transmit Rings, and Receive Rings 31
MAC Clients and Ring Allocation 31
Datalink Properties for Ring Allocation 33
Commands for Working With Receive and Transmit Rings 33
Obtaining and Interpreting Ring Information 34
▼ How to Configure Clients and Allocate Rings 38
Working With Pools and CPUs 40
▼ How to Configure a CPU Pool for a Datalink 42
▼ How to Allocate CPUs to a Link 43
Managing Resources on Flows 44
▼ How to Configure Flows 45

4 Monitoring Network Traffic and Resource Usage in Oracle Solaris 51
Overview of Network Traffic Flow 51
Commands for Monitoring Traffic Statistics 54
Gathering Statistics About Network Traffic on Links 54
Obtaining Network Traffic Statistics on Network Devices 55
Obtaining Network Traffic Statistics on Lanes 56
Obtaining Network Traffic Statistics on Link Aggregations 58
Gathering Statistics About Network Traffic on Flows 58
Configuring Network Accounting for Network Traffic 60
▼ How to Set Up Network Accounting 60
▼ How to Obtain Historical Statistics on Network Traffic 62

Index 65

Preface

Welcome to *Using Virtual Networks in Oracle Solaris 11.1*. This book is part of the series *Establishing an Oracle Solaris 11.1 Network*, which covers basic topics and procedures to configure Oracle Solaris networks. This book assumes that you have already installed Oracle Solaris.

Who Should Use This Book

This book is intended for anyone responsible for administering systems that run Oracle Solaris and which are configured in a network. To use this book, you should have at least two years of UNIX system administration experience. Attending UNIX system administration training courses might be helpful.

Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info` or visit `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs` if you are hearing impaired.

Typographic Conventions

The following table describes the typographic conventions that are used in this book.

TABLE P–1 Typographic Conventions

Typeface	Description	Example
`AaBbCc123`	The names of commands, files, and directories, and onscreen computer output	Edit your `.login` file. Use `ls -a` to list all files. `machine_name% you have mail.`

`AaBbCc123`	What you type, contrasted with onscreen computer output	`machine_name%` **`su`**
		`Password:`

TABLE P–1 Typographic Conventions *(Continued)*

Typeface	Description	Example
aabbcc123	Placeholder: replace with a real name or value	The command to remove a file is `rm` *filename*.
AaBbCc123	Book titles, new terms, and terms to be emphasized	Read Chapter 6 in the *User's Guide*. A *cache* is a copy that is stored locally. Do *not* save the file. **Note:** Some emphasized items appear bold online.

Shell Prompts in Command Examples

The following table shows the default UNIX system prompt and superuser prompt for shells that are included in the Oracle Solaris OS. Note that the default system prompt that is displayed in command examples varies, depending on the Oracle Solaris release.

TABLE P–2 Shell Prompts

Shell	Prompt
Bash shell, Korn shell, and Bourne shell	`$`
Bash shell, Korn shell, and Bourne shell for superuser	`#`
C shell	`machine_name%`
C shell for superuser	`machine_name#`

CHAPTER 1

Network Virtualization and Resource Management in Oracle Solaris

This chapter explains the basic concepts involved in network virtualization and resource control in Oracle Solaris. The following topics are covered:

- "Overview of Network Virtualization" on page 7
- "Overview of Network Resource Management" on page 11

These features help you to manage flow control, improve system performance, and configure the network utilization needed to achieve OS virtualization, utility computing, and server consolidation.

Overview of Network Virtualization

Network virtualization is the process of combining hardware network resources and software network resources into a single administrative unit. The goal of network virtualization is to provide systems and users with efficient, controlled, and secure sharing of the networking resources.

The end product of network virtualization is the *virtual network*. Virtual networks are classified into two broad types, external and internal. *External virtual networks* consist of several local networks that are administered by software as a single entity. The building blocks of classic external virtual networks are switch hardware and virtual local area network (VLAN) software technology. Examples of external virtual networks include large corporate networks and data centers.

This book focuses on the internal virtual network. An *internal virtual network* consists of one system using virtual machines or zones whose network interfaces are configured over at least one physical NIC. Those network interfaces are called *virtual network interface cards or virtual NICs (VNICs)*. These containers can communicate with each other as though they were on the same local network, effectively becoming a virtual network on a single host.

A special type of internal virtual network is the *private virtual network*. Private virtual networks are different from virtual private networks (VPNs). VPN software creates a secure point-to-point link between two endpoint systems. The private virtual network is a virtual network on a system that cannot be accessed by external systems. The isolation of this internal network from other external systems is achieved by configuring VNICs over etherstubs. Etherstubs are described in the following section.

You can combine networking resources to configure both internal and external virtual networks. For example, you can configure individual systems with internal virtual networks onto LANs that are part of a large, external virtual network.

Components of Network Virtualization

The following are the basic components of network virtualization in Oracle Solaris:

- Virtual network interface cards (VNICs)
- Virtual switches
- Etherstubs

VNICs are virtual network devices with the same datalink interfaces as a physical NIC. You configure VNICs over an underlying datalink. When VNICs are configured, they behave like physical NICs. In addition, the system's resources treat VNICs as if they were physical NICs. A VNIC has an automatically generated MAC address. Depending on the network interface in use, you can explicitly assign to a VNIC a MAC address other than this default address, as described in the `dladm`(1M) man page.

For the current list of physical interfaces that support VNICs, refer to the Network Virtualization and Resource Control FAQ (`http://hub.opensolaris.org/bin/view/Project+crossbow/faq`).

When you create a VNIC, a *virtual switch* is automatically created. In accordance with Ethernet design, if a switch port receives an outgoing packet from the host connected to that port, that packet cannot go to a destination on the same port. This design is a drawback for systems that are configured with virtual networks because the virtual networks share the same NIC. The outgoing packets go through a switch port out onto the external network. The incoming packets cannot reach their destination zone because the packets cannot return through the same port that they were sent through. Virtual switches provide these zones with a method to pass packets. The virtual switch opens a data path for the virtual networks to communicate with one another.

Etherstubs are pseudo Ethernet NICs. You can create VNICs over etherstubs instead of over physical links. VNICs over an etherstub become independent of the physical NICs on the system. With etherstubs, you can construct a private virtual network that is isolated both from the other virtual networks on the system and from the external network. For example, if you want to create a network environment whose access is limited only to your company developers and not to the network at large, etherstubs can be used to create such an environment.

Etherstubs and VNICs are only a part of the virtualization features of Oracle Solaris. You typically use these components with Oracle Solaris Zones. By assigning VNICs or etherstubs for use by zones, you can create a network within a single system. For information about zones, see *Oracle Solaris 11.1 Administration: Oracle Solaris Zones, Oracle Solaris 10 Zones, and Resource Management*.

By combining these components and deploying them with zones, you can have networks within a system similar to the following figure.

FIGURE 1–1 VNIC Configuration for a Single Interface

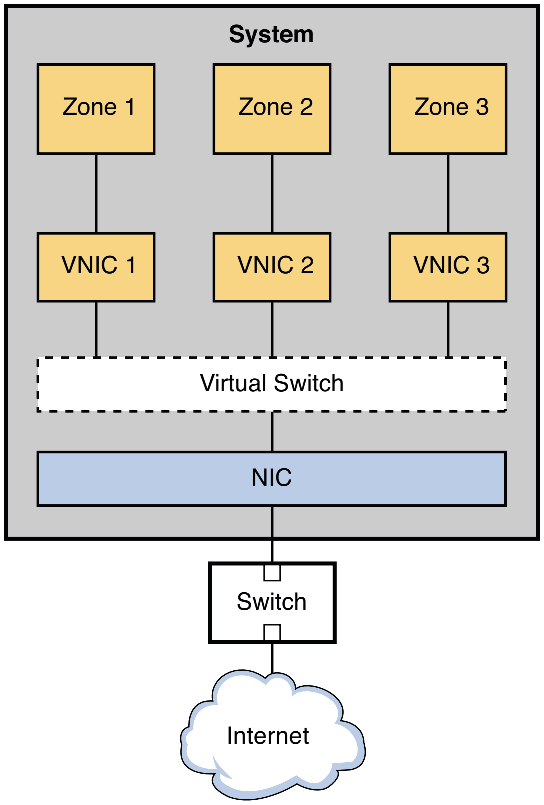


Figure 1–1 shows a single system with one NIC. The NIC is configured with three VNICs. Each VNIC supports a single zone. Zone 1, Zone 2, and Zone 3 constitute the virtual networks within the single system. The zones communicate with each other and with the external network by using their respective VNICs. In turn, the three VNICs connect to the underlying physical NIC through the virtual switch. The function of the virtual switch is equivalent to the connectivity that an external switch provides for the systems that are connected to the switch's ports.

When a virtual network is configured, a zone sends traffic to an external host in the same way as a system without a virtual network. Traffic flows from the zone, through the VNIC to the virtual switch, and then to the physical interface, which sends the data to the network.

The zones can also exchange traffic with one another inside the system. For example, packets pass from Zone 1 through its dedicated VNIC 1. The traffic then flows through the virtual switch to VNIC 3. VNIC 3 then passes the traffic to Zone 3. The traffic never leaves the system, and therefore never violates the Ethernet restrictions.

Alternatively, you can create a virtual network based on the etherstub. Etherstubs are entirely software based and do not require a network interface as the basis for the virtual network.

Who Should Implement Virtual Networks?

If you need to consolidate resources on Oracle's Sun servers, consider implementing VNICs and virtual networks. Consolidators at ISPs, telecommunications companies, and large financial institutions can use the following network virtualization features to improve the performance of their servers and networks.

- NIC hardware, including the powerful new interfaces that support hardware rings
- Multiple MAC addresses for the VNICs
- The substantial bandwidth provided by newer interfaces

You can replace many systems with a single system that has multiple zones or virtual machines, without significantly losing separation, security, and flexibility.

For a demonstration of the benefits of network virtualization, see Consolidating the Data Center With Network Virtualization (`http://download.oracle.com/otndocs/tech/OTN_Demos/data-center-consolidation.html`).

Commands for Configuring Virtualization Components

To create VNICs, use the `dladm create-vnic` command.

```
# dladm create-vnic -l link [-v vid] vnic
```

`-l` *link*	Refers to the name of the datalink over which the VNIC is configured.
`-v` *vid*	Refers to the VLAN ID for the VNIC if you want to create the VNIC as a VLAN. This option is not required. To configure a VNIC with a VLAN ID, see "How to Configure VNICs With VLAN IDs" on page 17. For more information about VLANs, see Chapter 3, "Working With VLANs," in *Managing Oracle Solaris 11.1 Network Performance*.

vnic	Refers to the name of the VNIC.

Note – You can configure other properties for a VNIC, such as MAC addresses, CPUs to be associated with the VNIC, and so on. For a list of these properties, refer to the `dladm`(1M) man page. Certain property modifications work only with VNICs. For example, with the `dladm create-vnic` command, you can configure a MAC address as well as assign a VLAN ID to create a VNIC as a VLAN. However, you cannot configure a MAC address directly for a VLAN by using the `dladm create-vlan` command.

You can create only one VNIC at a time over a datalink. As datalinks, VNICs have link properties that you can further configure as needed. "Datalink Properties for Resource Control" on page 11 lists some of these properties for managing the use of network resources in the system.

To create etherstubs, use the `dladm create-ether` command.

```
# dladm create-ether etherstub
```

Creating VNICs or etherstubs are only preliminary steps in configuring virtual networks. To use these components to create virtual networks on your system, see Chapter 2, "Creating and Administering Virtual Networks in Oracle Solaris."

Overview of Network Resource Management

This section explains different methods you can use to manage the use of network resources on a system.

Datalink Properties for Resource Control

In Oracle Solaris 11, quality of service (QoS) is obtained more easily and dynamically by managing network resources. Network resource management consists of setting datalink properties that pertain to network resources. By setting these properties, you determine how much of a given resource can be used for networking processes. For example, a link can be associated with a specific number of CPUs that are reserved exclusively for networking processes. Or, a link can be allotted a given bandwidth to process a specific type of network traffic.

After a resource property is defined, the new value takes effect immediately. This method makes managing resources flexible. You can set resource properties when you create the link. Alternatively, you can set these properties later, for example, after studying resource usage over time and determining how to better allocate the resource. The procedures for allocating resources apply to both the virtual network environment as well as the traditional physical network. For example, you use the `dladm set-linkprop` command to set properties that are related to network resources. The same syntax is used on both physical and virtual datalinks.

Network resource management is comparable to creating dedicated lanes for traffic. When you combine different resources to cater to specific types of network packets, those resources form a *network lane* for those packets. Resources can be assigned differently for each network lane. For example, you can allocate more resources to a lane where network traffic is heaviest. By configuring network lanes where resources are distributed according to actual need, you increase the system's efficiency to process packets. For more information about network lanes, see "Overview of Network Traffic Flow" on page 51.

Network resource management is helpful for the following tasks:

- Network provisioning
- Establishing service level agreements
- Billing clients
- Diagnosing security problems

You can isolate, prioritize, track, and control data traffic on an individual system without the complex QoS rule definitions.

Network Resource Management by Using Flows

A *flow* is a customized way of categorizing packets to further control how resources are used to process these packets. Network packets can be categorized according to an *attribute*. Packets that share an attribute constitute a flow and are labeled with a specific flow name. The flow can then be assigned specific resources.

The attributes that serve as the basis for creating flows are derived from the information in a packet's header. You can organize packet traffic into a flow according to one of the following attributes:

- IP address
- Transport protocol name (UDP, TCP, or SCTP)
- Application port number (for example, port 21 for FTP)
- DS field attribute, which is used for QoS in IPv6 packets only. For more information about the DS field, refer to *Managing IP Quality of Service in Oracle Solaris 11.1*.

A flow can be based on only one of the attributes in the list. For example, you can create a flow according to the port that is being used, such as port 21 for FTP, or according to IP addresses, such as packets from a specific source IP address. However, you cannot create a flow for packets from a specified IP address that are received on port number 21. Likewise, you cannot create a flow for all traffic from IP address `192.168.1.10` and then create a flow for transport layer traffic on `192.168.1.10`. Thus, you can configure multiple flows on a system, with each flow based on a different attribute.

Commands for Network Resource Management

The command to use for allocating network resources depends on whether you are directly working on datalinks or on flows.

- For datalinks, you use the appropriate `dladm` subcommand depending on whether you are setting the property while creating the link or while setting the property of an existing link. To simultaneously create a link and allocate resources to it, use the following syntax:

  ```
  # dladm create-vnic -l link -p property=value[,property=value] vnic
  ```

  where *link* can be either a physical link or a virtual link.

  To set the property of an existing link, use the following syntax:

  ```
  # dladm set-linkprop -p property=value[,property=value] link
  ```

  The following are link properties that you can set for resource allocation:

  - **Bandwidth** – You can limit a hardware's bandwidth for a certain link's use.
  - **NIC rings** – If a NIC supports ring allocation, its transmit and receive rings can be dedicated for use by datalinks. NIC rings are discussed in “Working With Clients, Transmit Rings, and Receive Rings” on page 31.
  - **CPU pools** – Pools of CPUs are generally created and associated with specific zones. These pools can be assigned to datalinks to reserve the sets of CPUs to manage the network processes of their associated zones. CPUs and pools are discussed in “Working With Pools and CPUs” on page 40.
  - **CPUs** – On a system with multiple CPUs, you can dedicate a given number of CPUs for specific network processing.

- For flows, you use `flowadm` subcommands. Managing resources on flows parallels the methods for managing resources on datalinks. To simultaneously create a flow and add resources to it, use the following syntax:

  ```
  # flowadm add-flow -l link -a attribute=value[,attribute=value] \
  -p property=value[,property=value] flow
  ```

  The set of defined attributes that characterizes the flows constitutes the system's *flow control policy*.

  To set the property of an existing flow, use the following syntax:

  ```
  # flowadm set-flowprop -p property=value[,property=value] flow
  ```

The properties for resource allocation that can be assigned to a flow are the same as the properties that are assigned directly to a link. Currently however, only the bandwidth properties can be associated with flows. Although the commands to set properties are different for datalinks and for flows, the syntax is similar. To configure the bandwidth properties, see the examples in "How to Configure Flows" on page 45

For more details about the `flowadm` command, refer to the `flowadm`(1M) man page. For a list of subcommands to use with the `flowadm` command, type the following:

```
# flowadm help
The following subcommands are supported:
Flow    : add-flow       remove-flow     reset-flowprop
          set-flowprop   show-flow       show-flowprop
For more info, run: flowadm help <subcommand>.
```

◆ ◆ ◆ CHAPTER 2

Creating and Administering Virtual Networks in Oracle Solaris

This chapter contains tasks for configuring virtual networks in a single system.

The following topics are covered:

- "Configuring the Components of Network Virtualization" on page 15
- "Building Virtual Networks" on page 18
- "Other Administrative Tasks for VNICs" on page 25

For an introduction to virtual networks, see Chapter 1, "Network Virtualization and Resource Management in Oracle Solaris."

Configuring the Components of Network Virtualization

In Oracle Solaris 11, etherstubs and VNICs are the basic components of network virtualization. This section describes the steps to configure these components in preparation for building the virtual network. For a description of these components, see "Components of Network Virtualization" on page 8.

The following procedures are described:

- "How to Configure VNICs and Etherstubs" on page 15
- "How to Configure VNICs With VLAN IDs" on page 17

▼ How to Configure VNICs and Etherstubs

The VNIC connects the virtual network to the external network. The VNIC also enables the zones to communicate with one another through the virtual switch that is automatically created with the VNIC. For a virtual network to host traffic internally between zones and with the external LAN and the Internet, each zone must have its own interface. Therefore, you must repeat this procedure as many times as the number of zones that will belong to the virtual network.

1 **Become an administrator.**

For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

2 **(Optional) Create the etherstub.**

```
# dladm create-etherstub etherstub
```

Perform this step only if you are creating a private virtual network which you want to restrict from being accessed by external systems. For a description of a private virtual network, see "Overview of Network Virtualization" on page 7.

Just like any datalink, you can name the etherstub in any way that is meaningful to your network setup. For guidelines on creating customized names, see "Rules for Valid Link Names" in *Introduction to Oracle Solaris 11 Networking*.

3 **Create the VNIC.**

```
# dladm create-vnic -l datalink [-v vid] vnic
```

If you are creating the VNIC for a private virtual network, then specify an etherstub for `datalink`. Include the `-v` *vid* in the command syntax only if you are creating the VNIC as a VLAN, where *vid* refers to the VNIC's VLAN ID. Otherwise, omit this option.

If you are creating a VNIC as a VLAN, refer to "How to Configure VNICs With VLAN IDs" on page 17 for additional steps that are specific to VNICs as VLANs.

You can assign any name to the VNIC. To assign customized names to VNICs, see "Rules for Valid Link Names" in *Introduction to Oracle Solaris 11 Networking*.

4 **Create an IP interface over the VNIC.**

```
# ipadm create-ip interface
```

5 **Assign a static IP address to the VNIC interface.**

```
# ipadm create-addr -a address interface
```

`-a` *address*	Specifies the IP address, which can be in CIDR notation.
interface	Specifies the VNIC that you created in the previous step.

The static IP address can be either IPv4 and IPv6 addresses. For more information about configuring IP addresses, see "How to Configure an IP Interface" in *Connecting Systems Using Fixed Network Configuration in Oracle Solaris 11.1*.

For more information about configuring IP addresses, see "How to Configure an IP Interface" in *Connecting Systems Using Fixed Network Configuration in Oracle Solaris 11.1*.

6 **Add the address information to the `/etc/hosts` file.**

▼ How to Configure VNICs With VLAN IDs

In the virtual network, you can configure VNICs with VLAN IDs to host VLAN traffic. You also set the link property `vlan-announce` to propagate the VLAN configurations of each individual VNIC to the network.

Unlike a regular VLAN link, the VNIC configured as a VLAN has its own MAC address. For information about non-VNIC VLANs, see Chapter 3, "Working With VLANs," in *Managing Oracle Solaris 11.1 Network Performance*.

Note – The following procedure contains only the steps to create the VNIC with a VLAN ID and to set the appropriate properties that enable the VNIC to service VLAN traffic. Although intermediary ports and switches are automatically updated when you enable the property, the endpoints must be separately configured to define VLANs at these points.

---

1. **Become an administrator.**

   For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

2. **Create a VNIC with a VLAN ID.**

   ```
   # dladm create-vnic -l link -v vid vnic
   ```

3. **Broadcast the VNIC's VLAN configuration to the network.**

   ```
   # dladm set-linkprop -p vlan-announce=gvrp link
   ```

   This step enables a GARP VLAN Registration Protocol (GVRP) client system that automatically registers VLAN IDs with attached switches. By default, the `vlan-announce` property is set to `off`, and no VLAN broadcast messages are sent to the network. After you set the property to `gvrp`, then the VLAN configuration for that link is propagated to enable automatic VLAN port configuration of the network devices. VLAN traffic can thus be accepted and forwarded by these devices.

4. **(Optional) To configure the wait period between VLAN broadcasts, set the `gvrp-timeout` property.**

   ```
   # dladm set-linkprop -p gvrp-timeout=time link
   ```

   where *time* is in milliseconds. The default value is 250 milliseconds. A system with a heavy load might require a shorter interval when rebroadcasting VLAN information. This property enables you to adjust the interval.

5. **(Optional) To display the values of the properties `vlan-announce` and `gvrp-timeout`, use the following command:**

   ```
   # dladm show-linkprop -p vlan-announce,gvrp-timeout
   ```

Example 2–1 Configuring a VNIC as a VLAN

This example creates a VNIC with a VLAN ID and enables the VLAN configuration to be announced to the network.

```
# dladm create-vnic -l net0 -v 123 vnic0
# dladm set-linkprop -p vlan-announce=gvrp net0
# dladm show-linkprop -p vlan-announce,gvrp-timeout net0
LINK    PROPERTY        PERM   VALUE   DEFAULT   POSSIBLE
net0    vlan-announce   rw     gvrp    off       gvrp,off
net0    gvrp-timeout    rw     250     250       --
```

Building Virtual Networks

A virtual network combines zones and the components of virtualization. You create as many zones as you require and as the system can support. Each zone has its own virtual interface. The zones in the system can communicate with each other. The virtual network as a whole connects to destinations on the larger external network.

Building a virtual network consists of one or more steps to configure etherstubs or VNICs as well as steps to configure zones. Although these are independent sets of procedures, both must be performed to complete the construction of the virtual network.

The procedures in this section proceed based on the following assumptions:

- The virtual network on the system consists of three zones. The zones are in different stages of configuration: the first zone is created as a new zone, the second zone already exists on the system and needs to be reconfigured to use a VNIC, and the third zone is designated to be a private virtual network. Thus, the procedures demonstrate various ways to prepare zones for the virtual network.
- The system's physical interface is configured with the IP address `192.168.3.70`
- The router's IP address is `192.168.3.25`

In each procedure in this section, more details are added to the scenario to provide a more concrete context to the steps.

When building the virtual network, some steps are performed in the global zone, and some steps are performed in a non-global zone. For clarity, the prompts in the examples after each procedure indicate in which zone a specific command is issued. However, the actual path that the prompts display might vary depending on the prompts specified for your system.

In this section, the following procedures are discussed:

- "How to Configure a Zone for the Virtual Network" on page 19
- "How to Reconfigure a Zone to Use a VNIC" on page 21
- "How to Create a Private Virtual Network" on page 23

▼ How to Configure a Zone for the Virtual Network

This procedure explains how to configure a new zone with a new VNIC. Note that only the steps related to network virtualization are included in the procedure. For more detailed instructions on configuring zones, refer to Chapter 17, "Planning and Configuring Non-Global Zones (Tasks)," in *Oracle Solaris 11.1 Administration: Oracle Solaris Zones, Oracle Solaris 10 Zones, and Resource Management*.

The procedure assumes that this first zone for the virtual network is created as a fresh zone.

1 **Become an administrator.**

For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

2 **Configure the VNIC.**

See "How to Configure VNICs and Etherstubs" on page 15. However, for this specific procedure, omit the step to create an etherstub.

3 **Create the zone.**

```
global# zonecfg -z zone
```

When creating the zone, make sure that you set the `ip-type` parameter to `exclusive`, and that you assign the VNIC you had just created to be the zone's physical interface.

4 **To exit the zone configuration mode, verify and then commit the configuration.**

5 **Install the zone.**

```
global# zoneadm -z zone install
```

Note – The installation process can take a while.

---

6 **Start the zone.**

```
global# zoneadm -z zone boot
```

7 **After the zone completely boots up, log in to the zone.**

```
# zlogin -C zone
```

8 **Supply the information as you are prompted.**

Most of the information is supplied by selecting from a list of choices. Typically, the default options suffice. To configure the virtual network, you must supply or verify the following information:

- Host name of the zone, for example `zone1`.

- IP address of the zone which is based on the IP address of the zone's VNIC.
- Whether IPv6 should be enabled.
- Whether the system with the virtual network is part of a subnet.
- Netmask of the IP address.
- Default route, which can be the IP address of the physical interface on which the virtual network is built.

After you have supplied the required information, the zone restarts.

Example 2–2 Configuring a Zone for the Virtual Network

This example includes detailed steps to create `zone1`. However, only the zone parameters that are relevant to the creation of a virtual network are listed.

```
global # zonecfg -z zone1
zonecfg:zone1> create
zonecfg:zone1> set zonepath=/export/home/zone1
zonecfg:zone1> set autoboot=true
zonecfg:zone1> set ip-type=exclusive
zonecfg:zone1> add net
zonecfg:zone1:net> set physical=vnic1
zonecfg:zone1:net> end
zonecfg:zone1> verify
zonecfg:zone1> commit
zonecfg:zone1> exit

global# zoneadm -z zone1 install
Preparing to install zone <zone1>
Creating list of files to copy from the global zone.
.
.
Zone <zone1> is initialized.

global# zoneadm -z zone1 boot

zlogin -C zone1
What type of terminal are you using?
.
.
.
8) Sun Workstation
9) Televideo 910
10) Televideo 925
11) Wyse Model 50
12) X Terminal Emulator (xterms)
13) CDE Terminal Emulator (dtterm)
14) Other
Type the number of your choice and press Return: 13
.
(More prompts)
..
```

For network information, the following information is supplied:

```
Hostname: zone1
IP address: 192.168.3.80
System part of a subnet: Yes
Netmask: 255.255.255.0
Enable IPv6: No
Default route: 192.168.3.70
Router IP address: 192.168.3.25
```

▼ How to Reconfigure a Zone to Use a VNIC

This procedure refers to the second zone in the virtual network. The zone already exists, but its current configuration prevents it from becoming a part of the virtual network. Specifically, the zone's IP type is a shared type and its current interface is `net0`. Both of these configurations must be changed.

1. **Become an administrator.**

   For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

2. **Create the VNIC.**

   ```
   global# dladm create-vnic [-v vid] -l datalink vnic
   ```

   where *vid* refers to the VLAN ID that you assign to the VNIC. Specify the VLAN ID only if you want to create the VNIC as a VLAN.

   Do not configure the VNIC's interface yet. You will perform this step later in this procedure.

3. **Change the zone's IP type from shared to exclusive.**

   ```
   global# zonecfg -z zone
   zonecfg:zone1> set ip-type=exclusive
   zonecfg:zone1>
   ```

4. **Change the zone's interface to use a VNIC.**

   ```
   zonecfg:zone1> remove net physical=NIC
   zonecfg:zone1> add net
   zonecfg:zone1:net> set physical=vnic
   zonecfg:zone1:net> end
   zonecfg:zone1>
   ```

5. **Verify and commit the changes you have implemented and then exit the zone.**

   ```
   zonecfg:zone1 verify
   zonecfg:zone1> commit
   zonecfg:zone1> exit
   global#
   ```

6 **Reboot the zone.**

```
global# zoneadm -z zone reboot
```

7 **Log in to the zone.**

```
global# zlogin zone
```

8 **Configure the VNIC with a valid IP address.**

If you are assigning a static address to the VNIC, you would type the following:

```
zone# ipadm create-addr -a address interface
```

where *address* can use CIDR notation.

9 **From the global zone, add the address information to the `/etc/hosts` file.**

Example 2–3 Reconfiguring a Zone Configuration to Use a VNIC

In this example, `zone2` already exists as a shared zone. The zone also uses the primary interface of the system rather than a virtual link. You need to modify `zone2` to use `vnic2`. To use `vnic2`, `zone2`'s IP type must first be changed to exclusive. Note that some of the output is truncated to focus on the relevant information that relates to virtual networks.

```
global# dladm create-vnic -l net0 vnic2

global# zonecfg -z zone2
zonecfg:zone1> set ip-type=exclusive
zonecfg:zone1> remove net physical=net0
zonecfg:zone1> add net
zonecfg:zone1:net> set physical=vnic2
zonecfg:zone1:net> end
zonecfg:zone1> verify
zonecfg:zone1> commit
zonecfg:zone1> exit
global# zoneadm -z zone2 reboot

global# zlogin zone2
zone2# ipadm create-ip vnic2
zone2# ipadm create-addr -a 192.168.3.85/24 vnic2
ipadm: vnic2/v4

zone2# exit

global# vi /etc/hosts
#
::1             localhost
127.0.0.1       localhost
192.168.3.70    loghost   #For net0
192.168.3.80    zone1   #using vnic1
192.168.3.85    zone2   #using vnic2
```

▼ How to Create a Private Virtual Network

The following procedure explains how to configure the third zone of the virtual network. Although the zone is part of the virtual network, it will be inaccessible from external systems. To enable the isolated zone to send network traffic beyond the system, then you must use network address translation (NAT). NAT translates the VNIC's private IP addresses to routeable IP addresses of the physical network interface. However, the private IP addresses themselves are not visible from the external network. For more information about NAT, see "Using IP Filter's NAT Feature" in *Securing the Network in Oracle Solaris 11.1*.

The use of etherstubs constitutes the main difference between a regular virtual network and a private virtual network. In a private virtual network, the VNICs that are assigned to the zones are configured over an etherstub. Thus, they are isolated from network traffic that flows through the system.

This procedure assumes that the zone already exists, but currently does not have any associated interface.

1. **Become an administrator.**

   For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

2. **Create the etherstub.**

   ```
   global# dladm create-etherstub etherstub
   ```

3. **Create a VNIC over the etherstub.**

   ```
   global# dladm create-vnic -l etherstub vnic
   ```

   Do not configure the VNIC's interface yet. You will perform this step later in this procedure.

4. **Assign the VNIC to the zone.**

   ```
   global# zonecfg -z zone
    zone# set physical=vnic
   ```

5. **Verify and commit the changes you have implemented and then exit the zone.**

   ```
   zonecfg:zone1 verify
   zonecfg:zone1> commit
   zonecfg:zone1> exit
   global#
   ```

6. **Log in to the zone.**

   ```
   # zlogin zone
   ```

7. **In the zone, create an IP interface over the VNIC that is now assigned to the zone.**

   ```
   # ipadm create-ip interface
   ```

8 **Configure the VNIC with a valid IP address.**

If you are assigning a static address to the VNIC, you would type the following:

```
zone# ipadm create-addr -a address interface
```

where *address* can use CIDR notation.

9 **From the global zone, add the address information to the `/etc/hosts` file.**

10 **From the global zone, set the primary interface to perform IP forwarding.**

```
# ipadm set-ifprop -p forwarding=on -m ipv4 primary-interface
```

Note – Typically in Oracle Solaris 11, the primary interface uses the name `net0`.

---

11 **From the global zone, configure network address translation (NAT) in the `/etc/ipnat.conf` file for the primary interface.**

12 **Start the IP filter service to enable NAT.**

```
# svcadm enable network/ipfilter
```

13 **Reboot the zone.**

```
# zoneadm -z zone reboot
```

Example 2–4 Creating a Private Virtual Network Configuration

In this example, `zone3` is configured to be isolated as a private network. NAT and IP forwarding are also configured to allow the virtual private network to send packets outside the host while still concealing its private address from the external network. The zone is already configured with an exclusive IP type. However, no IP interface is assigned to it.

```
global# dladm create-etherstub ether0
global# dladm create-vnic -l ether0 vnic3
global# zonecfg -z zone3
zonecfg:zone3> add net
zonecfg:zone3:net> set physical=vnic3
zonecfg:zone3:net> end
zonecfg:zone3> verify
zonecfg:zone3> commit
zonecfg:zone3> exit
global#

global# zlogin zone3
zone3# ipadm create-ip vnic3
zone3# ipadm create-addr -a 192.168.0.10/24 vnic3
ipadm: vnic3/v4
zone3# exit

global# cat /etc/hosts
```

```
::1             localhost
127.0.0.1       localhost
192.168.3.70    loghost   #For net0
192.168.3.80    zone1   #using vnic1
192.168.3.85    zone2   #using vnic2
192.168.0.10    zone3   #using vnic3

global# ipadm set-ifprop -p forwarding=on -m ipv4 vnic3

global# vi /etc/ipf/ipnat.conf
map vnic3 192.168.0.0/24 -> 0/32  portmap tcp/udp auto
map vnic3 192.168.0.0/24 -> 0/32

global# svcadm enable network/ipfilter
global# zoneadm -z zone3 boot
```

Other Administrative Tasks for VNICs

This section describes tasks you can perform on VNICs after performing basic configuration. This section covers the following topics:

- "Modifying the VLAN ID of a VNIC" on page 26
- "Modifying VNIC MAC Addresses" on page 26
- "Migrating VNICs" on page 27
- "Displaying VNIC Information" on page 29
- "How to Delete a VNIC" on page 29

Note – VNICs can be configured as VLANs. A parallel subcommand, `dladm modify-vlan`, enables you to modify direct VLANs that have been created with the `dladm create-vlan` command.. You must use the correct subcommand depending on whether you are modifying VLANs or VNICs configured as VLANs. Use the `modify-vlan` subcommand on VLANs displayed by the `dladm show-vlan` subcommand. Use the `modify-vnic` subcommand on VNICs, including those with VLAN IDs,, displayed by the `dladm show-vnic` subcommand. To modify direct VLANs, see "Modifying VLANs" in *Managing Oracle Solaris 11.1 Network Performance*.

Two types of VNIC modification are available:

- Global modification changes an attribute of all the VNICs on a specific datalink at once. You use the `-L` option to identify the underlying datalink whose VNICs you want to modify.
- Selective modification modifies an attribute of selected VNICs. Instead of using the `-L` option to identify the underlying datalink, you specify the VNICs whose attribute you want to change.

You can modify the following attributes: the VLAN ID, the MAC address, and the underlying link. Modifying the underlying link means moving a VNIC to another datalink. The following sections discuss these modifications in detail.

Modifying the VLAN ID of a VNIC

To change a VNIC's VLAN ID, use one of the following commands:

- `dladm modify-vnic -v` *vid* `-L` *datalink*

  In this command, *vid* specifies the new VLAN ID that you assign to the VNIC. *datalink* refers to the underlying link over which the VNIC is configured. You can use this command syntax if only a single VNIC exists on the datalink. The command fails on a datalink that has multiple configured VNICs because these VNICs must have unique VLAN IDs.

- `dladm modify-vnic -v` *vid vnic*

  Use this command to change the unique VLAN IDs of multiple VNICs over a single datalink. Because each VLAN ID is unique for VNICs on the same datalink, you must change the VLAN IDs one at a time. Suppose that you want to change the VLAN IDs of `vnica0`, `vnicb0`, and `vnicc0`, which are configured over `net0`. You would proceed as follows:

  ```
  # dladm modify-vnic -v 123 vnica0
  # dladm modify-vnic -v 456 vnicb0
  # dladm modify-vnic -v 789 vnicc0
  ```

- `dladm modify-vnic -v` *vid vnic*`,`*vnic*`,[...]`

  Use this command to change the VLAN ID of VNICs as a group, provided that each VNIC is on a different datalink. Suppose that you want to change the VLAN IDs of `vnic0`, `vnic1`, and `vnic2`. These VNICs are configured over `net0`, `net1`, and `net2`, respectively. You would use the following command:

  ```
  # dladm modify-vnic -v 123 vnic0,vnic1,vnic2
  ```

Modifying VNIC MAC Addresses

VNICs have unique MAC addresses. To modify these addresses, use one of the following commands as appropriate to your specific circumstances:

- `dladm modif-vnic -m` *mac-address vnic*

  Use this command to assign a specific MAC address to a specific VNIC.

- `dladm modif-vnic -m random -L` *datalink*

  This command performs a global modification, in which you change the MAC address of all the VNICs on the datalink. The system automatically assigns unique MAC addresses to the VNICs. In this command, the `-m random` option is equivalent to the `-m auto` option.

- `dladm modif-vnic -m random` *`vnic`*`,`*`vnic`*`,[...]`

  This command performs a selective VNIC modification. Note that for both global and selective modifications, you specify `random` for the `-m` option.

You can modify a VNIC's VLAN ID and MAC address with a single command. However, be careful about using commands to modify multiple VNIC attributes globally, which might cause unexpected behavior. Changing multiple attributes one VNIC at a time is preferable to changing multiple attributes of a group of VNICs all at the same time.

The following example shows the output before and after you modify a VNIC's VLAN ID and 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 -m random -v 123 vnic0
# dladm show-vnic vnic0
LINK      OVER     SPEED  MACADDRESS        MACADDRTYPE    VID
vnic0     net0     1000   2:8:20:0:1:2      random         123
```

Migrating VNICs

You can move one or more VNICs from one underlying datalink to another underlying datalink without deleting and reconfiguring the VNICs. The underlying link can be a physical link, a link aggregation, or an etherstub.

To successfully migrate VNICs, the underlying datalink to which the VNICs are moved must be able to accommodate the datalink properties of the VNICs. If those properties are not supported, then migration fails and the user is notified. After a successful migration, all the applications that use the VNICs continue to operate normally, provided that the VNICs remain connected to the network.

Certain hardware-dependent properties might change after a VNIC migration, such as the datalink state, link speed, MTU size, and so on. The values of these properties are inherited from the datalink to which the VNICs are migrated.

You can also migrate VNICs globally or selectively. Global migration means that you migrate all the VNICs over a datalink to another datalink. To perform a global migration, you only need to specify the source datalink and the target datalink. The following example moves all the VNICs from `ether0` to `net1`:

```
# dladm modify-vnic -l net1 -L ether0
```

where

- `-l` *datalink* refers to the destination datalink to which the VNICs are migrated.

- `-L` *datalink* refers to the original datalink over which the VNICs are configured.

Note – You must specify the destination datalink before the source datalink.

To perform selective VNIC migration, you specify the VNICs that you want to move. The following example moves selected VNICs from `net0` to `net1`:

```
# dladm modify-vnic -l net1 vnic0,vnic1,vnic2
```

Note – The `-L` option is restricted to global modification only.

While migrating a group of VNICs, you can also modify their VLAN IDs at the same time. However, to assign new VLAN IDs, you must migrate the VNICs one at a time, as shown in the following example:

```
# dladm modify-vnic -l net1 -v 123 vnic0
# dladm modify-vnic -l net1 -v 456 vnic1
# dladm modify-vnic -l net1 -v 789 vnic2
```

The effects of migration on the MAC address depend on whether the VNIC is using a factory MAC address from the source datalink.

- If you do not specify the `-m` option during migration, then after migration, the factory MAC address is replaced by an address randomly assigned from the target datalink.
- If you use the `-m` *address* option during migration, then that address is assigned to the VNIC after migration.

Randomly assigned MAC addresses are unaffected and are retained by their respective VNICs after migration.

The following example shows how to migrate multiple VNICs. Note that the VNICs are using randomly assigned MAC addresses. Thus, these addresses are unchanged after migration.

```
# dladm show-vnic
LINK      OVER     SPEED  MACADDRESS        MACADDRTYPE    VID
vnic1     net0     1000   2:8:20:c2:39:38   random         0
vnic2     net0     1000   2:8:20:5f:84:ff   random         0

# dladm modify-vnic -l net1 -L net0
# dladm show-vnic vnic0
LINK      OVER     SPEED  MACADDRESS        MACADDRTYPE    VID
vnic1     net1     1000   2:8:20:c2:39:38   random         0
vnic2     net1     1000   2:8:20:5f:84:ff   random         0
```

Displaying VNIC Information

To obtain information about the VNICs on your system, use the `dladm show-vnic` command.

```
# dladm show-vnic
LINK      OVER     SPEED          MACADDRESS          MACADDRTYPE
vnic1     net0     1000 Mbps      2:8:20:c2:39:38     random
vnic2     net0     1000 Mbps      2:8:20:5f:84:ff     random
```

VNICs are also datalinks. Thus, you can also use any dladm command that shows information about datalinks to include information about VNICs if these exist on the system. For example, `dladm show-link` includes VNICs on the list. Or, you can use the `dladm show-linkprop` command to check the properties of VNICs. To obtain property information about a single VNIC, specify the VNIC when you display link properties:

```
# dladm show-linkprop [-p property] vnic
```

▼ How to Delete a VNIC

This procedure explains how to delete a VNIC configuration from the system. The steps assume that the VNIC is attached to a zone. You must be in the global zone to perform this procedure.

1 **Because the VNIC is attached to a zone, halt the zone.**

```
global# zoneadm -z zone halt
```

Note – To determine the links used by a zone, use the `dladm show-link` command.

---

2 **Remove or detach the VNIC from the zone.**

```
global# zonecfg -z zone remove net physical=vnic
```

3 **Delete the VNIC from the system.**

```
global# dladm delete-vnic vnic
```

4 **Reboot the zone.**

```
global# zonecfg -z zone boot
```

Example 2–5 Deleting a VNIC From the System

In this example, `vnic1` is removed from `zoneB` and from the system.

```
Global# dladm show-link
LINK            CLASS   MTU    STATE   OVER
net0            phys    1500   up      --
net2            phys    1500   up      --
```

```
net1            phys    1500   up      --
net3            phys    1500   up      --
zoneA/net0      vnic    1500   up      net0
zoneB/net0      vnic    1500   up      net0
vnic0           vnic    1500   up      net1
zoneA/vnic0     vnic    1500   up      net1
vnic1           vnic    1500   up      net1
zoneB/vnic1     vnic    1500   up      net1

Global# zoneadm -z zoneB halt
Global# zonecfg -z zoneB remove net physical=vnic1
Global# dladm delete-vnic vnic1
Global# zonecfg -z zoneB reboot
```

◆ ◆ ◆ CHAPTER 3

Managing Network Resources in Oracle Solaris

This chapter explains how to manage resources on datalinks, including virtual links such as VNICs. Network resource management implements IP quality of service (QoS) to enhance performance, especially for the virtual network.

The following topics are covered:

- "Working With Clients, Transmit Rings, and Receive Rings" on page 31
- "Working With Pools and CPUs" on page 40
- "Managing Resources on Flows" on page 44

Working With Clients, Transmit Rings, and Receive Rings

On NICs, receive (Rx) rings and transmit (Tx) rings are hardware resources through which the system receives and sends network packets, respectively. The following sections provide an overview of rings, followed by procedures that are used to allocate rings for networking processes. Examples are also provided to show how the mechanism works when you issue commands to allocate rings.

MAC Clients and Ring Allocation

MAC clients such as VNICs and other datalinks are configured over a NIC to enable communication between a system and other network nodes. After a client is configured, it uses both Rx and Tx rings to receive or transmit network packets respectively. A MAC client can be either hardware-based or software-based. A hardware-based client fulfills any one of the following conditions:

- It has dedicated use of one or more Rx rings.
- It has dedicated use of one or more Tx rings.
- It has dedicated use of one or more Rx rings and one or more Tx rings.

Clients that do not fulfill any of these conditions are software-based MAC clients.

Hardware-based clients can be assigned rings for exclusive use depending on the NIC. NICs such as `nxge` support *dynamic ring allocation*. On such NICs, you can configure not only hardware-based clients. You also have the flexibility to determine the number of rings to allocate to such clients, assuming that rings remain available for allocation. The use of rings is always optimized for the primary interface, for example, `net0`. The primary interface is also known as the *primary client*. Any available rings that have not been assigned for exclusive use by other clients are automatically assigned to the primary interface.

Other NICs such as `ixge` only support *static ring allocation*. On these NICs, you can only create hardware-based clients. The clients are automatically configured with a fixed set of rings per client. The fixed set is determined during the NIC driver's initial configuration. For more information about a driver's initial configuration for static ring allocation, refer to the *Oracle Solaris 11.1 Tunable Parameters Reference Manual*.

Software-based clients do not have exclusive use of rings. Rather, they share rings with other existing software-based clients or with the primary client. The rings that software-based clients use depends on the number of hardware-based clients that have priority in ring allocation.

It is important to understand the distinction between the *primary client* and other *secondary clients*. The primary client is the physical datalink of the NIC. Based on the generic names provided by Oracle Solaris during installation, the primary client would be named `net`*N*, where *N* is an instance number. For an explanation of generic names for datalinks, see "Network Devices and Datalink Names" in *Introduction to Oracle Solaris 11 Networking*. VNICs are *secondary clients* that are created over the physical datalink. If these clients are hardware-based clients, then they can have exclusive use of rings. Otherwise, the clients are software-based.

Ring Allocation in VLANs

With VLANs, the assignment of rings proceeds differently depending on how the VLAN is created. VLANs are created in one of two ways:

- By using the `dladm create-vlan` subcommand:

  ```
  # dladm create-vlan -l link -v vid vlan
  ```
- By using the `dladm create-vnic` subcommand:

  ```
  # dladm create-vnic -l link -v vid vnic
  ```

A VLAN that is created by the `dladm create-vlan` subcommand shares the same MAC address as the underlying interface. Consequently, that VLAN also shares the Rx and Tx rings of the underlying interface. A VLAN that is created as a VNIC by using the `dladm create-vnic` command has a different MAC address from its underlying interface. The allocation of rings for such a VLAN is independent of the allocation for the underlying link. Thus, that VLAN can be assigned its own dedicated rings, assuming that the NIC supports hardware-based clients.

Datalink Properties for Ring Allocation

To administer rings, two ring properties can be set by using the `dladm` command:

- `rxrings` refers to the number of assigned Rx rings to a specified link.
- `txrings` refers to the number of assigned Tx rings to a specified link.

You can set each property to one of three possible values:

- `sw` indicates that you are configuring a software-based client. The client does not have exclusive use of rings. Rather, the client shares rings with any other existing clients that are similarly configured.
- *n > 0* (number greater than zero) applies to the configuration of a hardware-based client only. The number refers to the quantity of rings that you allocate to the client for its exclusive use. You can specify a number only if the underlying NIC supports dynamic ring allocation.
- `hw` also applies to the configuration of a hardware-based client. However, for such a client, you cannot specify the actual number of dedicated rings. Rather, the fixed number of rings per client is already set according to the NIC driver's initial configuration. You set the `*rings` properties to `hw` only if the underlying NIC supports static ring allocation.

To provide information about current ring assignments and use, the following additional read-only ring properties are available:

- `rxrings-available` and `txrings-available` indicate the number of Rx and Tx rings that are available for allocation.
- `rxhwclnt-available` and `txhwclnt-available` indicate the number of Rx and Tx hardware-based clients that can be configured over a NIC.

Commands for Working With Receive and Transmit Rings

To manage the use of receive and transmit rings of datalinks, you use the following principal `dladm` subcommands:

- `dladm show-linkprop` – Displays the current values of link properties, including Rx and Tx rings. The output provides the following information about a datalink's ring support capabilities. You need the information to determine what type of client you can configure to use Rx and Tx rings.
  - The available clients that you can create
  - The available rings that you can allocate to available clients
  - The capability to support dynamic or static ring allocation

- If only static ring allocation is supported, the current distribution of rings to existing clients

  "Datalink Properties for Ring Allocation" on page 33 further describes how to interpret the output of this command.
- `dladm show-phys -H` *datalink* – Displays how the rings of a physical datalink are currently being used by existing clients.
- `dladm create-vnic -p` *ring-properties vnic* – Creates a client with a specific number of Tx or Rx rings to use to service traffic.
- `dladm set-linkprop -p` *ring-properties datalink* – Allocates rings to a specific client, provided that rings are available and ring allocation is supported.

Obtaining and Interpreting Ring Information

This section describes the `dladm show-linkprop` output that displays ring-related properties of a datalink.

Displaying Ring Allocation Capabilities of a Datalink

This section provides examples of command output about ring-related properties and explains the type of information you can obtain. The following NICs are used in the examples:

- `net0` (over `nxge`)
- `net1` (over `ixgbe`)
- `net2` (over `e1000g`)

EXAMPLE 3–1 nxge Ring Information

The following example shows ring information for `nxge`:

```
# dladm show-linkprop net0
LINK     PROPERTY            PERM  VALUE   DEFAULT   POSSIBLE
...
net0     rxrings             rw    --      --        sw,<1-7>
...
net0     txrings             rw    --      --        sw,<1-7>
...
net0     rxrings-available   r-    5       --        --
net0     txrings-available   r-    5       --        --
net0     rxhwclnt-available  r-    2       --        --
net0     txhwclnt-available  r-    2       --        --
...
```

With `net0`, the values of the `POSSIBLE` field are `sw` and `<1-7>` for `rxrings` and `txrings`. These values indicate that `nxge` supports hardware-based clients as well as software-based clients. The range `<1-7>` indicates the limits of the number of Rx rings or Tx rings you can set for clients. The range also indicates that the NIC supports dynamic ring allocation for both the receive and transmit sides.

EXAMPLE 3–1 nxge Ring Information *(Continued)*

In addition, the `*rings-available` properties indicate that five Rx rings and five Tx rings are available to allocate to hardware-based clients.

However, the `*clnt-available` properties show that you can configure only two clients that can have exclusive use of available Rx rings. Likewise, you can configure only two clients that can have exclusive use of available Tx rings.

EXAMPLE 3–2 ixgbe Ring Information

The following example shows ring information for `ixgbe`:

```
# dladm show-linkprop net1
LINK   PROPERTY            PERM  VALUE  DEFAULT  POSSIBLE
...
net1   rxrings             rw     --    --       sw,hw
...
net1   txrings             rw     --    --       sw,hw,<1-7>
...
net1   rxrings-available   r-     0     --       --
net1   txrings-available   r-     5     --       --
net1   rxhwclnt-available  r-     0     --       --
net1   txhwclnt-available  r-     7     --       --
...
```

With `net1`, the `POSSIBLE` field values `sw` and `hw` for both `rxrings` and `txrings` indicate that `ixgbe` supports both hardware-based clients and software-based clients. For Rx rings, only static ring allocation is supported, where the hardware assigns a fixed set of Rx rings to each hardware-based client. However, for Tx rings, the range <1–7> indicates that dynamic allocation is supported. You can determine the number of Tx rings to assign to a hardware-based client, in this example, up to seven rings.

In addition, the `*rings-available` properties indicate that five Tx rings are available to allocate to hardware-based clients, but no Rx rings can be assigned.

Finally, based on the `*hwclnt-available` properties, you can configure seven hardware-based Tx clients to use Tx rings exclusively. However, you cannot create hardware-based clients with exclusive use of Rx rings because dynamic Rx ring allocation is not supported.

A zero (`0`) under the `VALUE` field for either of the `*rings-available` properties can mean one of the following:

- No more rings are available to allocate to clients.
- Dynamic ring allocation is not supported.

You can verify the meaning of the zero by comparing the `POSSIBLE` field for `rxrings` and `txrings` with the `VALUE` field for `rxrings-available` and `txrings-available`.

For example, suppose that `txrings-available` is `0`, as follows:

EXAMPLE 3–2 ixgbe Ring Information *(Continued)*

```
# dladm show-linkprop net1
LINK   PROPERTY           PERM  VALUE  DEFAULT  POSSIBLE
...
net1   rxrings            rw    --     --       sw,hw
net1   txrings            rw    --     --       sw,hw,<1-7>
net1   rxrings-available  r-    0      --       --
net1   txrings-available  r-    0      --       --
...
```

In this output, the `VALUE` field for `rxrings-available` is `0` while the `POSSIBLE` field for `rxrings` is `sw,hw`. The combined information means that no Rx rings are available because the NIC does not support dynamic ring allocation. On the transmit side, the `VALUE` field for `txrings-available` is `0` while the `POSSIBLE` field for `txrings` is `sw,hw,<1-7>`. The combined information indicates that no Tx rings are available because all the Tx rings have already been allocated. However, as the `POSSIBLE` field for `txrings` indicates, dynamic ring allocation is supported. Thus, you can allocate Tx rings as these rings become available.

EXAMPLE 3–3 e1000g Ring Information

The following example shows ring information for `e1000g`:

```
# dladm show-linkprop net2
LINK   PROPERTY            PERM  VALUE  DEFAULT  POSSIBLE
...
net2   rxrings            rw     --      --     --
...
net2   txrings            rw     --      --     --
...
net2   rxrings-available  r-     0       --     --
net2   txrings-available  r-     0       --     --
net2   rxhwclnt-available r-     0       --     --
net2   txhwclnt-available r-     0       --     --
...
```

The output indicates that neither rings nor hardware-based clients can be configured because ring allocation is not supported in `e1000g`.

Displaying Ring Use and Ring Assignments on a Datalink

Two read-only datalink properties provide information about how rings are currently used by existing clients on the datalink:

- `rxrings-effective`
- `txrings-effective`

To obtain information about ring use and which rings are distributed to clients, use both the `dladm show-linkprop` and `dladm show-phys -H` subcommands.

The following examples show different types of output generated by the two commands with respect to the use of Rx and Tx rings and how these rings are distributed among clients.

EXAMPLE 3–4 Ring Use of the Primary Client

The primary client is the interface that is configured over the physical datalink of the NIC. For this example, the NIC is an `ixgbe` card. By default its datalink is `net0`. The IP interface over `net0` is the primary client.

```
# dladm show-linkprop net0
LINK   PROPERTY              PERM  VALUE  DEFAULT  POSSIBLE
...
net0   rxrings              rw    --     --       sw,hw
net0   rxrings-effective    r     2      --       --
net0   txrings              rw    --     --       sw,hw,<1-7>
net0   txrings-effective    r     8      --       --
net0   txrings-available    r-    7      --       --
net0   rxrings-available    r-    0      --       --
net0   rxhwclnt-available   r-    3      --       --
net0   txhwclnt-available   r-    7      --       --
...
# dladm show-phys -H net0
LINK   RINGTYPE   RINGS   CLIENTS
net0   RX         0-1     <default,mcast>
net0   TX         0-7     <default>net0
net0   RX         2-3     net0
net0   RX         4-5     --
net0   RX         6-7     --
```

The output provides the following information about the use and distribution of rings for the primary client `net0`:

- `rxrings-effective` shows that `net0` automatically receives two Rx rings. `txrings-effective` shows that `net0` has use of eight Tx rings. By default, all unused rings are automatically assigned to the primary client.
- Based on the output of the `dladm show-phys -H` command, the two Rx rings allocated to `net0` are rings `2` and `3`. For Tx rings, `net0` uses rings `0` through `7`.

EXAMPLE 3–5 Ring Use of a Secondary Client

This example assumes the configuration of a VNIC client `vnic1` over `net0`, the physical datalink of the `ixgbe` card.

```
# dladm show-linkprop vnic1
LINK    PROPERTY             PERM  VALUE  DEFAULT  POSSIBLE
...
vnic1   rxrings              rw    hw     --       sw,hw
vnic1   rxrings-effective    r-    2      --       --
vnic1   txrings              rw    hw     --       sw,hw,<1-7>
vnic1   txrings-effective    r-    1      --       --
...
# dladm show-linkprop net0
LINK    PROPERTY             PERM  VALUE  DEFAULT  POSSIBLE
...
net0  rxrings             rw     --    --       sw,hw
net0  rxrings-effective   r-     2     --       --
net0  txrings             rw     --    --       sw,hw,<1-7>
```

EXAMPLE 3–5 Ring Use of a Secondary Client *(Continued)*

```
net0  txrings-effective   r-     --    --       --
net0  txrings-available   r-     6     --       --
net0  rxrings-available   r-     0     --       --
net0  rxhwclnt-available  r-     3     --       --
net0  txhwclnt-available  r-     6     --       --
...
# dladm show-phys -H net0
LINK     RINGTYPE   RINGS     CLIENTS
net0   RX         0-1       <default,mcast>
net0   TX         0,2-7     <default>net0
net0   RX         2-3       net0
net0   RX         4-5       vnic1
net0   RX         6-7       --
net0   TX         1         vnic1
```

The combined output from the three commands provides the following information:

- `rxrings-effective` for `vnic1` shows that this VNIC automatically receives two Rx rings. `txrings-effective` shows that `vnic1` has use of one Tx ring. These rings are statically allocated, as indicated by the `hw` value that is set for the `*ring` properties.
- Based on the output of the `dladm show-phys -H` command, the two Rx rings allocated to `net0` are rings `2` and `3`. For Tx rings, `net0` uses ring `0` and rings `2` through `7`. `vnic1` uses ring `1` for a Tx ring and rings `4` and `5` for Rx rings.

Note that `vnic1` is configured as a hardware-based client with static ring allocation. Consequently, the number of available Tx hardware clients (`txhwclnt-available`) that can be created over `net0` is reduced to six.

▼ How to Configure Clients and Allocate Rings

This procedure explains how to configure clients on a datalink based on the type of support for ring allocation. Make sure that you can interpret the output of the `dladm` commands that display datalink ring properties, as explained in "Displaying Ring Allocation Capabilities of a Datalink" on page 34 and "Displaying Ring Use and Ring Assignments on a Datalink" on page 36. The information guides you in the way you configure the clients.

1 **Display the datalink's ring properties.**

```
# dladm show-linkprop datalink
```

From the output, determine the following:

- Whether the NIC supports hardware-based clients
- The type of ring allocation that the NIC supports
- The availability of rings to allocate to hardware-based clients
- The availability of hardware-based clients that you can configure on the link

2 **Depending on the information from the previous step, perform one of the following:**

- If the NIC supports dynamic ring allocation, create the hardware-based client with the following syntax:

  ```
  # dladm create-vnic -p rxrings=number[,txrings=number] -l link vnic
  ```

  If the client was previously created, use the following syntax:

  ```
  # dladm set-linkprop -p rxrings=number[,txrings=number] vnic
  ```

  Note – Some NICs support dynamic ring allocation on either Rx rings or Tx rings, but not on both types. You specify *number* on the ring type for which dynamic ring allocation is supported.

- If the NIC supports static ring allocation, create the hardware-based client with the following syntax:

  ```
  # dladm create-vnic -p rxrings=hw[,txrings=hw] -l link vnic
  ```

  If the client was previously created, use the following syntax:

  ```
  # dladm set-linkprop -p rxrings=hw[,txrings=hw] vnic
  ```

  Note – Some NICs support static ring allocation on either Rx rings or Tx rings, but not on both types. You specify `hw` on the ring type for which static ring allocation is supported.

- If the NIC supports only software-based clients, create the client with the following syntax:

  ```
  # dladm create-vnic -p rxrings=sw[,txrings=sw] -l link vnic
  ```

  If the client was previously created, use the following syntax:

  ```
  # dladm set-linkprop -p rxrings=sw[,txrings=sw] vnic
  ```

3. **(Optional) Check the newly created client's ring information.**

   ```
   # dladm show-linkprop vnic
   ```

4. **(Optional) Check how the datalink's rings are distributed among different clients.**

   ```
   # dladm show-phys -H datalink
   ```

See Also For an example that shows how to use flows and how to allocate system resources, including Rx and Tx rings, to process network traffic in a virtual network, see Example 3–8.

Working With Pools and CPUs

The `pool` link property enables you to bind network processing to a pool of CPUs. With this property, you can better integrate network resource management with CPU binding and administration in zones. In Oracle Solaris, zone administration includes the binding of non-networking processes to a pool of CPU resources by using the `zonecfg` or `poolcfg` command. To dedicate that same pool of resources to also manage network processes, you use the `dladm set-linkprop` command to configure a link's `pool` property. Then you assign that link to the zone.

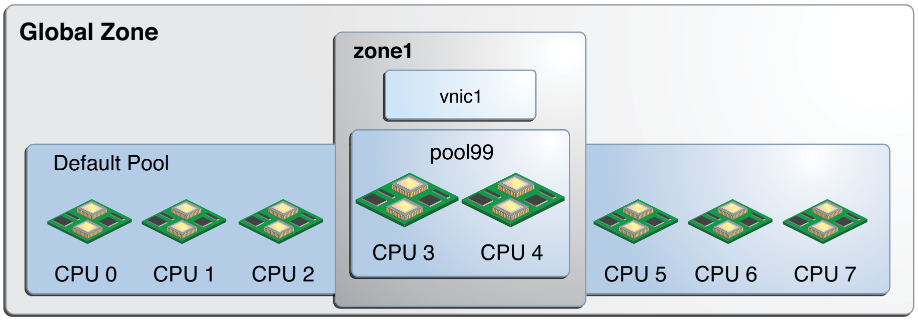
By setting the `pool` property for a link and assigning the link as the zone's network interface, that link becomes bound to a zone's pool as well. If that zone is set to become an exclusive zone, then CPU resources in the pool can no longer be used by other datalinks that are not assigned to that zone.

Note – A separate property, `cpu`, can be set to assign specific CPUs to a datalink. The two properties, `cpu` and `pool`, are mutually exclusive. You cannot set both properties for a given datalink. To assign CPU resources to a datalink by using the `cpu` property, see "How to Allocate CPUs to a Link" on page 43.

For more information about pools within a zone, see Chapter 13, "Creating and Administering Resource Pools (Tasks)," in *Oracle Solaris 11.1 Administration: Oracle Solaris Zones, Oracle Solaris 10 Zones, and Resource Management*. For more information about creating pools and assigning CPU sets to the pools, refer to the `poolcfg`(1M) man page.

The following figure show how pools work when the `pool` property is assigned to a datalink.

FIGURE 3–1 pool Property of a VNIC Assigned to a Zone

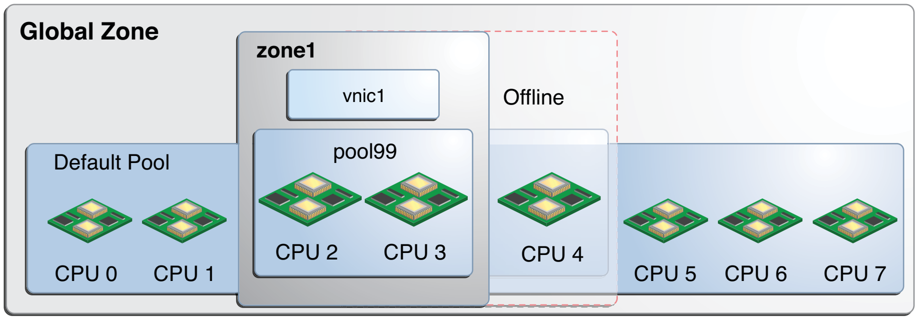


In the figure, the system has eight CPUs. When no pools are configured on the system, all the CPUs belong to the *default pool* and are used by the global zone. However, in this example, the `pool99` pool has been created and consists of `CPU 3` and `CPU 4`. This pool is associated with zone1, which is an exclusive zone. If pool99 is set as a property of vnic1, then pool99 becomes dedicated to also manage vnic1's networking processes. After vnic1 is assigned to be zone1's network interface , the CPUs in pool99 become reserved to manage both networking and non-networking processes of zone1.

The pool property is dynamic in nature. Zone pools can be configured with a range of CPUs, and the kernel determines which CPUs are assigned to the pool's CPU set. Changes to the pool are automatically implemented for the datalink, which simplifies pool administration for that link. In contrast, assigning specific CPUs to the link by using the cpu property requires you to specify the CPU to be assigned. You have to set the cpu property every time you want to change the CPU components of the pool.

For example, suppose that the system CPU 4 in Figure 3–1 is taken offline. Because the pool property is dynamic, the software automatically associates an additional CPU with the pool. Thus, the pool's original configuration of two CPUs is preserved. For vnic1, the change is transparent. The adjusted configuration is shown in the following figure.

FIGURE 3–2 Automatic Reconfiguration of the pool Property



Additional pool-related properties display information about a datalink's use of CPUs or a pool of CPUs. These properties are read-only and cannot be set by the administrator.

- pool-effective displays the pool that is being used for network processes.
- cpus-effective displays the list of CPUs that are being used for network processes.

To manage CPU resources of a zone, setting a datalink's pool property is not normally performed as an initial step. More frequently, commands such as zonecfg and poolcfg are used to configure a zone to use a pool of resources. The cpu and pool link properties themselves are not set. In such cases, the pool-effective and the cpus-effective properties of these datalinks are set automatically according to the zone configurations when the zone is booted. The default pool is displayed under pool-effective, and the value of cpus-effective is selected by the system. Thus, if you use the dladm show-linkprop command, the pool and cpu properties will be empty, but the pool-effective and cpus-effective properties will contain values.

Directly setting the `pool` and `cpu` properties of a datalink is an alternative step that you can use to bind a zone's CPU pool for networking processes. After you configure these properties, their values are reflected in the `pool-effective` and `cpus-effective` properties as well. Note, however, that this alternative step is used less often to manage a zone's network resources.

▼ How to Configure a CPU Pool for a Datalink

As with other link properties, the `pool` property can be set for a datalink either at the moment when the link is created or later when the link requires further configuration.

To set the `pool` property while you create the VNIC, you use the following syntax:

```
# dladm create-vnic -p pool=pool-name -l link vnic
```

To set the `pool` property of an existing VNIC, you use the following syntax:

```
# dladm setlinkprop -p pool=pool-name vnic
```

The following procedure explains how to configure a CPU pool for a VNIC.

Before You Begin You must have completed the following:

- Created a processor set with its assigned number of CPUs
- Created a pool with which the processor set will be associated
- Associated the pool with the processor set

Note – For the instructions to complete these prerequisites, see "How to Modify a Configuration" in *Oracle Solaris 11.1 Administration: Oracle Solaris Zones, Oracle Solaris 10 Zones, and Resource Management*.

---

1 **Set the link's `pool` property to the pool of CPUs that you created for the zone. Perform one of the following steps, depending on whether the VNIC exists:**

- If the VNIC has not yet been created, use the following syntax:

  ```
  # dladm create-vnic -l link -p pool=pool vnic
  ```

  where *pool* refers to the name of the pool that was created for the zone.

- If the VNIC exists, use the following syntax:

  ```
  # dladm setlinkprop -p pool=pool vnic
  ```

2 **Set a zone to use the VNIC.**

```
zonecfg>zoneid:net> set physical=vnic
```

Note – For the instructions that explain how to assign a networking interface to a zone, refer to the "Configuring, Verifying, and Committing a Zone" in *Oracle Solaris Administration: Oracle Solaris Zones, Oracle Solaris 10 Zones, and Resource Management*

Example 3–6 Assigning a Link's CPU Pool to a Zone With an Exclusive IP-Type

This example shows how a pool is assigned to a zone's datalink. The scenario is based on the configuration in Figure 3–1. The example assumes that a pool of CPUs named `pool99` has already been configured for the zone. The pool is then assigned to a VNIC. Finally, the non-global zone `zone1` is set to use the VNIC as the network interface.

```
# dladm create-vnic -l net1 -p pool99 vnic1

# zonecfg -c zone1
zonecfg:zone1> set ip-type=exclusive
zonecfg:zone1> add net
zonecfg:zone1>net> set physical=vnic1
zonecfg:zone1>net> end
zonecfg:zone1> exit
```

▼ How to Allocate CPUs to a Link

The following procedure explains how to assign specific CPUs to process traffic traversing a datalink by configuring the `cpu` property.

1 **Check CPU assignments for the interface.**

```
# dladm show-linkprop -p cpus link
```

By default, no CPUs are assigned to any specific interface. Thus, the parameter `VALUE` in the command output will not contain any entry.

2 **List the interrupts and the CPUs with which the interrupts are associated.**

```
# echo ::interrupts | mdb -k
```

The output lists parameters for each link on the system, including the CPU number.

3 **Assign CPUs to the link.**

The CPUs can include those with which the link's interrupts are associated.

```
# dladm set-linkprop -p cpus=cpu1,cpu2,... link
```

where *cpu1* is the CPU number to be assigned to the link. You can dedicate multiple CPUs to the link.

4 **Check the link interrupt to verify the new CPU assignments.**

```
# echo ::interrupts | mdb -k
```

5 **(Optional) Display the CPUs that are associated with the link.**

```
# dladm show-linkprop -p cpus link
```

Example 3–7 Allocating CPUs to a Link

This example shows how to dedicate specific CPUs to the datalink `net0`.

Note the following information in the output that is generated by the different commands. For clarity, the significant information is emphasized in the output.

- By default `net0` has no dedicated CPU. Thus, `VALUE` is `--`.
- The interrupt of `net0` is associated with CPU 18.
- After CPUs are allocated, `net0` displays a new CPU list under `VALUE`.

```
# dladm show-linkprop -p cpus net0
LINK          PROPERTY     PERM     VALUE     DEFAULT     POSSIBLE
net0     cpus          rw       --        --          --

# echo ::interrupts | mdb -k
Device  Shared   Type   MSG #   State   INO    Mondo   Pil   CPU
net#0   no       MSI    2       enbl    0x1a   0x1a    6     18

# dladm set-linkprop -p cpus=14,18,19,20 net0

# dladm show-linkprop -p cpus net0
LINK   PROPERTY   PERM   VALUE        DEFAULT   POSSIBLE
net0   cpus       rw     14,18,19,20  --        --
```

All the supporting threads, including the interrupt, are now confined to the newly assigned set of CPUs.

See Also For an example that shows how to use flows and how to allocate system resources, including CPUs and CPU pools, to process network traffic in a virtual network, see Example 3–8.

Managing Resources on Flows

Flows consist of network packets that are organized according to an attribute. Flows enable you to further allocate network resources. For an overview of flows, see "Network Resource Management by Using Flows" on page 12.

Using flows for managing resources involves the following general steps:

1. Creating the flow by basing it on a specific attribute as listed in "Network Resource Management by Using Flows" on page 12.
2. Customizing the flow's use of resources by setting properties that pertain to network resources. Currently, only the bandwidth for processing packets can be set.

▼ How to Configure Flows

1 **If necessary, list the available links to determine the link on which you will configure flows.**

```
# dladm show-link
```

2 **Verify that IP interfaces over the selected link are properly configured with IP addresses.**

```
# ipadm show-addr
```

3 **Create flows according to the attribute you have determined for each flow.**

```
# flowadm add-flow -l link -a attribute=value[,attribute=value] flow
```

link	Refers to the link on which you are configuring the flow.
attribute	Refers to one of the following classifications by which you can organize network packets into a flow: ■ IP address ■ Transport protocol (UDP, TCP, or SCTP) ■ Port number for an application (for example port 21 for FTP) ■ DS field attribute, which is used for quality of service in IPv6 packets only. For more information about the DS field, refer to "DS Codepoint" in *Managing IP Quality of Service in Oracle Solaris 11.1*.
flow	Refers to the name that you assign to the particular flow.

For more details about flows and flow attributes, see the `flowadm`(1M) man page.

4 **(Optional) Display the possible range of values for the datalink's bandwidth.**

```
# dladm show-linkprop -p maxbw link
```

where *link* is the datalink on which the flow is configured.

The range of values is listed on the `POSSIBLE` field.

5 **Allocate bandwidth share to the flow.**

```
# flowadm set-flowprop -p maxbw=value flow
```

The value you set must be within the allowed range of values for the link's bandwidth.

Note – Currently, only a flow's bandwidth can be customized.

---

6 **(Optional) Display the flows that you have created over the link.**

```
# flowadm
```

Note – The `flowadm` command, if used without any subcommand, provides the same information as the `flowadm show-flow` command.

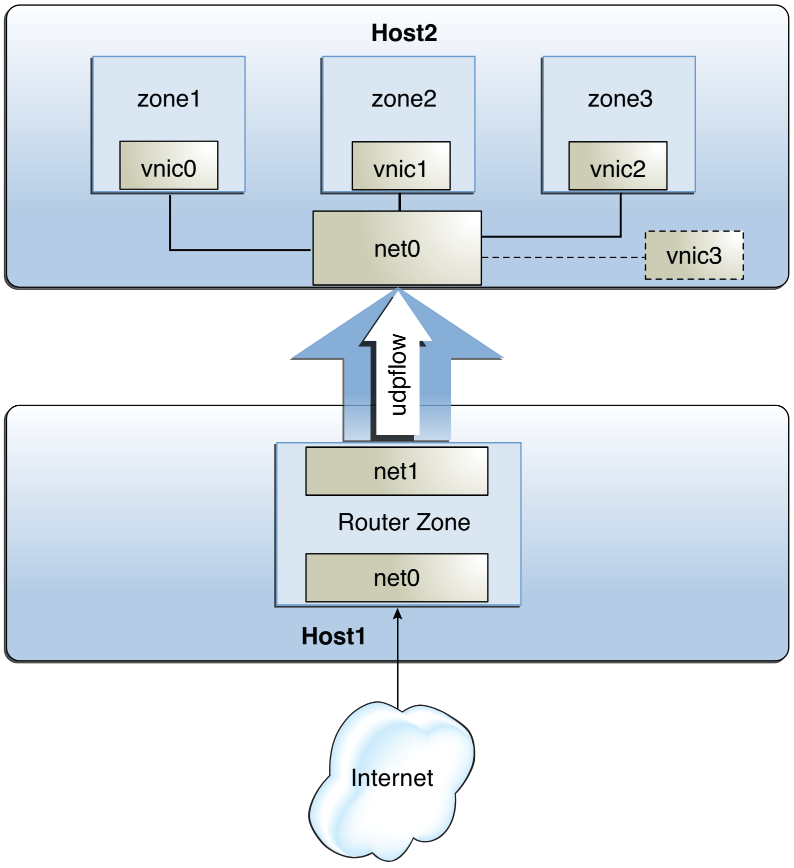
---

7 **(Optional) Display the property values for a specified flow.**

```
# flowadm show-flowprop flow
```

Example 3–8 Managing Resources by Setting Link and Flow Properties

This example combines the steps for allocating network resources to both datalinks and flows. The example is based on the configuration shown in the following figure.



The figure shows two physical hosts that are connected to each other.

- Host1 has the following configuration:
  - It has one non-global zone that functions as a router zone. Two interfaces are assigned to the zone: net0 connects to the Internet and net1 connects to the internal network including the second host.
  - udpflow is a flow configured over net0 to isolate UDP traffic and implement control over how UDP packets use resources. For information about configuring flows, see "Managing Resources on Flows" on page 44.
- Host2 has the following configuration:
  - It has three non-global zones and their respective VNICs. The VNICs are configured over net0 whose card supports dynamic ring allocation. For more information about ring allocation, see "Working With Clients, Transmit Rings, and Receive Rings" on page 31.
  - Each zone's network processing load is different. For the purposes of this example, the load for zone1 is heavy, the load for zone2 is moderate, and the load for zone3 is light. Resources are assigned to these zones according to their loads.
  - A separate VNIC is configured as a software-based client. For an overview of MAC clients, see "MAC Clients and Ring Allocation" on page 31.

The tasks in this example involve the following:

- Creating a flow and configuring flow controls – A flow is created over net1 to create separate resource controls over UDP packets that are received by Host2.
- Configuring network resource properties for the VNICs on Host2 – Based on the processing load for each zone, each zone's VNIC is configured with a set of dedicated rings. A separate VNIC is also configured without dedicated rings as an example of a software-based client.

Note that the example does not include any procedures for zone configuration. To configure zones, refer to Chapter 17, "Planning and Configuring Non-Global Zones (Tasks)," in *Oracle Solaris 11.1 Administration: Oracle Solaris Zones, Oracle Solaris 10 Zones, and Resource Management*.

First, view information about links and IP interfaces on Host1.

```
# ipadm
NAME              CLASS/TYPE STATE     UNDER      ADDR
lo0               loopback   ok        --         --
    lo0/v4         static     ok        --         127.0.0.1/8
net0              ip         ok        --         --
    net0/v4        static     ok        --         10.10.6.5/24
net1              ip         failed    ipmp0      --
    net1/v4        static     ok        --         10.10.12.42/24
```

Next, create a flow over net1 to isolate UDP traffic to Host2. Then, implement resource controls on the flow.

```
# flowadm add-flow -l net1 -a transport=udp udpflow
# flowadm set-flowprop -p maxbw=80 udpflow
```

Then, check the information about the created flow.

```
flowadm
FLOW        LINK   IPADDR   PROTO   LPORT   RPORT   DFSLD
udpflow     net1   --       udp     --      --      --

# flowadm show-flowprop
FLOW         PROPERTY        VALUE          DEFAULT        POSSIBLE
udpflow      maxbw              80          --             --
```

On `Host2`, configure VNICs over `net0` for each zone. Implement resource controls on each VNIC. Then, assign the VNICs to their respective zones.

```
# dladm create-vnic -l net0 vnic0
# dladm create-vnic -l net0 vnic1
# dladm create-vnic -l net0 vnic2

# dladm set-prop -p rxrings=4,txrings=4 vnic0
# dladm set-prop -p rxrings=2,txrings=2 vnic1
# dladm set-prop -p rxrings=1,txrings=1 vnic2


# zonecfg -z zone1
# zonecfg:zone1> add net
# zonecfg:zone1:net> set physical=vnic0
# zonecfg:zone1:net> end
# zonecfg:zone1> commit
# zonecfg:zone1> exit
# zoneadm -z zone1 reboot

# zonecfg -z zone2
# zonecfg:zone2> add net
# zonecfg:zone2:net> set physical=vnic1
# zonecfg:zone2:net> end
# zonecfg:zone2> commit
# zonecfg:zone2> exit
# zoneadm -z zone2 reboot
#

# zonecfg -z zone3
# zonecfg:zone3> add net
# zonecfg:zone3:net> set physical=vnic2
# zonecfg:zone3:net> end
# zonecfg:zone3> commit
# zonecfg:zone3> exit
# zoneadm -z zone3 reboot
#
```

Suppose that `pool1`, a set of CPUs in `Host2`, was previously configured for use by `zone1`. Bind that pool of CPUs to also manage network processes for `zone1` as follows:

```
# dladm set-prop -p pool=pool1 vnic0
```

Finally, create a software-based client that shares rings with `net0`, the primary interface.

```
# dladm create-vnic -p rxrings=sw,txrings=sw -l net0 vnic3
```

◆ ◆ ◆ CHAPTER 4

Monitoring Network Traffic and Resource Usage in Oracle Solaris

This chapter describes tasks for monitoring and gathering statistics about the use of network resources in a physical as well as a virtual network environment Oracle Solaris 11. The information can help you analyze resource allocation for provisioning, consolidation, and billing purposes. This chapter introduces the two commands that you use to display statistics: `dlstat` and `flowstat`.

The following subjects are discussed:

- "Overview of Network Traffic Flow" on page 51
- "Commands for Monitoring Traffic Statistics" on page 54
- "Gathering Statistics About Network Traffic on Links" on page 54
- "Gathering Statistics About Network Traffic on Flows" on page 58
- "Configuring Network Accounting for Network Traffic" on page 60

Overview of Network Traffic Flow

Packets traverse a path when they flow into or out of a system. On a granular level, packets are received and transmitted through receive (Rx) rings and transmit (Tx) rings of a NIC. From these rings, received packets are passed up the network stack for further processing while outbound packets are sent to the network.

This section introduces the concept of network lanes. A *network lane* is a combination of system resources that are allocated to manage network traffic. Thus, network lanes are customized paths for specific types of network traffic. Each lane can be either a *hardware lane* or a *software lane*. In addition, each lane type can be either a *receive* lane or a *transmit* lane.

The distinction between hardware and software lanes is based on a NIC's ability to support rings and ring allocation as described in "Working With Clients, Transmit Rings, and Receive Rings" on page 31. This chapter focuses primarily on incoming traffic that is received through receive lanes.

Note – Rx and Tx rings, as well as other network resources, are assigned by setting properties of datalinks. Therefore, datalinks are the network lanes on a system.

On hardware lanes, packets have exclusive use of rings that are assigned to those lanes. In contrast, rings on software lanes are shared by all network packets on those lanes.. Datalinks are configured to share rings for the following reasons:

- Administrative intent. The datalink might not be performing intensive processes to require dedicated rings.
- The NIC does not support ring allocation.
- Despite support for ring allocation, rings are no longer available to be assigned for exclusive use.

The following figure shows different hardware lanes.

FIGURE 4–1 Hardware Lanes

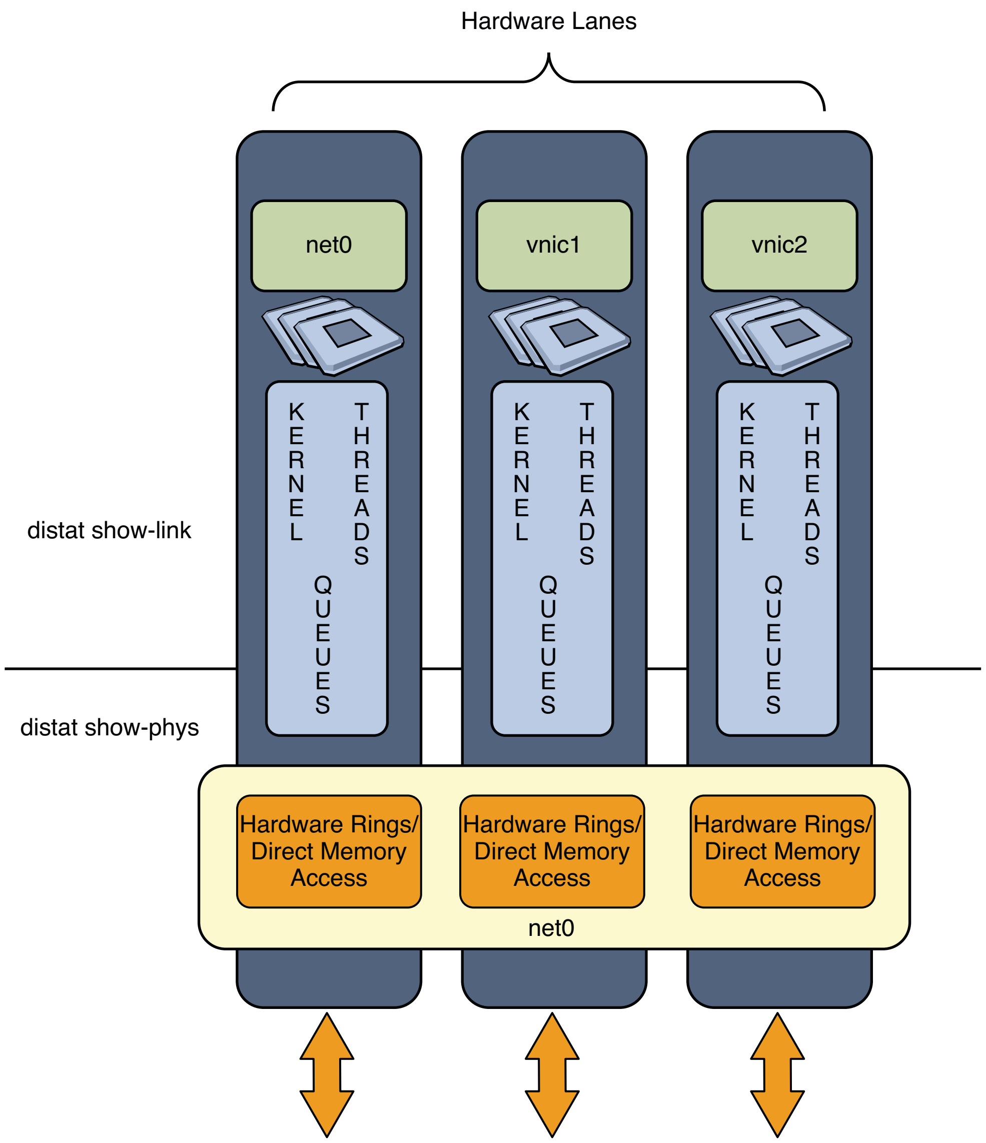


Figure 4–1 shows the following configuration:

- Two VNICs are configured over `net0`: `vnic1` and `vnic2`. As the primary client, `net0` is also the primary lane. The VNICs as secondary clients are also network lanes.
- Sets of hardware rings are assigned to the secondary clients. Each client, including the primary client, functions as a hardware lane.

- A set of CPUs is allocated to each lane.

The subsequent sections describe how you monitor the traffic that flows through these lanes.

Commands for Monitoring Traffic Statistics

The `dlstat` and `flowstat` commands enable you to monitor and obtain statistics about network traffic on datalinks and flows, respectively. These commands are equivalent to the `dladm` and `flowadm` commands. The following table compares the functions of the pair of `*adm` commands to the pair of `*stat` commands.

Administrative Commands		**Monitoring Commands**	
Command	**Function**	**Command**	**Function**
`dladm` command options	User interface and tool for configuring and administering datalinks	`dlstat` command options	User interface and tool for obtaining statistics on traffic on datalinks
`flowadm` command options	User interface and tool for configuring and administering flows	`flowstat` command options	User interface and tool for obtaining statistics on traffic on flows

The subsequent sections describe each monitoring command in greater detail.

Gathering Statistics About Network Traffic on Links

You can use the following variants of the `dlstat` command to gather network traffic information.

Command	**Information Provided**
`dlstat` [*link*] `dlstat -rt` [*link*] `dlstat show-link` [*link*]	Inbound and outbound traffic statistics per lane
`dlstat show-link -rt` [*link*]	Inbound and outbound traffic statistics per ring per lane
`dlstat show-phys` [*link*]	Inbound and outbound traffic statistics per network physical device
`dlstat show-phys -rt` [*link*]	Inbound and outbound traffic statistics per ring per network physical device

Command	Information Provided
`dlstat show-aggr` [*link*] `dlstat show-aggr -rt` [*link*]	Inbound and outbound traffic statistics per port per aggregation

You can use either the `-r` option or the `-t` option to the `dlstat` command to restrict the statistics information to the receive side or the transmit side, respectively. Moreover, you can use other options with the `dlstat` command aside. For more information, refer to the `dlstat`(1M) man page.

Obtaining Network Traffic Statistics on Network Devices

The `dlstat show-phys` subcommand provides statistics that refer to the physical network device. As shown in Figure 4–1, the subcommand operates on the hardware rings. which are on the device layer of the network stack. The equivalent subcommand `dladm show-phys` also operates on the same level of the stack. Compare Figure 4–1 with the network stack illustrated in "Oracle Solaris Implementation" in *Introduction to Oracle Solaris 11 Networking*.

The following example shows statistics for all the physical links on the system:

```
# dlstat show-phys
 LINK    IPKTS    RBYTES    OPKTS     OBYTES
net0     2.14M   257.48M    3.19M    210.88M
net1     1.15M   120.32M    1.00M     98.70M
net2     1.10M   110.10M    1.28     183.00M
...
```

The output shows both incoming and outgoing traffic statistics on each link on the system. The number of packets and their byte sizes are displayed.

The following example shows receive-side statistics on each of `net0`'s hardware rings:

```
# dlstat show-phys -r net0
 LINK    TYPE       ID   INDEX     IPKTS     RBYTES
net0       rx    local      --         0          0
net0       rx       hw       1         0          0
net0       rx       hw       2     1.73M      2.61G
net0       rx       hw       3         0          0
net0       rx       hw       4     8.44M     12.71G
net0       rx       hw       5     5.68M      8.56G
net0       rx       hw       6     4.99M      7.38G
net0       rx       hw       7         0          0
```

In the second output, the `net0` device has eight receive rings, which are identified under the `INDEX` field. An even distribution of packets per ring is an ideal configuration that indicates that the rings are properly allocated to links according to the links' load. An uneven distribution might indicate a disproportionate distribution of rings per link. The resolution of the uneven distribution depends on whether the NIC supports dynamic ring allocation. If it does, you can redistribute rings per link to process packets more evenly. For more information about dynamic ring allocation, see "Working With Clients, Transmit Rings, and Receive Rings" on page 31.

The following example shows information about traffic that is being received on the devices every second. The interval is specified by using the `-i` option. To stop the display from refreshing, press Control-C.

```
# dlstat show-phys -r -i 1
LINK  TYPE   INDEX      IPKTS    RBYTES
net0    rx       0   101.91K    32.86M
net1    rx       0     9.61M    14.47G
net2    rx       8      336K         0
net0    rx       0         0         0
net1    rx       0    82.13K   123.69M
net2    rx       0         0         0
...
^C
```

This example shows the usage of the transmit rings for `net1` as a network device.

```
# dlstat show-phys -t net1
LINK TYPE INDEX   OPKTS   OBYTES
net1   tx     0      44    3.96K
net1   tx     1       0        0
net1   tx     2   1.48M  121.68M
net1   tx     3   2.45M  201.11M
net1   tx     4   1.47M  120.82M
net1   tx     5       0        0
net1   tx     6   1.97M  161.57M
net1   tx     7   4.59M  376.21M
net1   tx     8   2.43M  199.24M
net1   tx     9       0        0
net1   tx     10  3.23M  264.69M
net1   tx     11  1.88M  153.96M
```

Obtaining Network Traffic Statistics on Lanes

The `dlstat show-link` subcommand provides statistics that refer to the lanes that are configured over the physical link The lanes are constituted by the datalinks. As shown in Figure 4–1, the subcommand operates on the datalink layer of the network stack. The equivalent subcommand `dladm show-link` also operates on the same level of the stack. Compare Figure 4–1 with the network stack illustrated in "Oracle Solaris Implementation" in *Introduction to Oracle Solaris 11 Networking*.

The following example shows receive-side traffic statistics for `vnic0`.

```
# dlstat show-link -r vnic0
 LINK  TYPE    ID  INDEX    IPKTS   RBYTES    INTRS    POLLS  IDROPS
```

```
vnic0    rx    hw      2    1.73M    2.61G    1.33M  400.22K       0
vnic0    rx    hw      4    8.44M   12.71G    4.35M    4.09M       0
```

The previous output shows traffic statistics for the lane `vnic0`. This lane has been allocated two receive rings (ring 2 and ring 4) for exclusive use. The output shows how these two rings are used for incoming network traffic. However, the data might also reflect the implementation of other resource allocations, such as bandwidth limits and priority processing.

Suppose the following information is displayed for `net0`, the primary lane:

```
# dlstat show-link -r net0
LINK TYPE    ID INDEX   IPKTS  RBYTES    INTRS  POLLS  IDROPS
net0   rx local    --       0       0        0      0       0
net0   rx    sw    -- 794.28K   1.19G  794.28K      0       0
...
```

Based on the output, one of the Rx rings, ring `0`, is being shared with another client. Rings are shared if secondary clients are configured without any allocated rings. Ring might not be allocated for the following reasons:

- Hardware clients are no longer available to be created over the link.
- Hardware rings are no longer available to be allocated.
- The administrator intentionally configures a software client.

The statistics for interrupt (`INTRS`) and drops (`*DROPS`) are also significant. Low interrupt numbers and zero packet drops indicate greater efficiency in performance. If the interrupt numbers or packet drop numbers are high, then you might need to add more resources to the lane.

The following example shows statistics about outbound packets on the rings used by `net1`, the primary lane. The output shows that `net1` uses all of the Tx rings.

```
# dlstat show-link -t net1
LINK   TYPE   ID  INDEX    OPKTS    OBYTES   ODROPS
net1     tx   hw      0       32     1.44K        0
net1     tx   hw      1        0         0        0
net1     tx   hw      2    1.48M    97.95M        0
net1     tx   hw      3    2.45M   161.87M        0
net1     tx   hw      4    1.47M    97.25M        0
net1     tx   hw      5        0       276        0
net1     tx   hw      6    1.97M   130.25M        0
net1     tx   hw      7    4.59M   302.80M        0
net1     tx   hw      8    2.43M   302.80M        0
net1     tx   hw      9        0         0        0
net1     tx   hw      10   3.23M   213.05M        0
net1     tx   hw      11   1.88M   123.93M        0
```

The following command shows the usage of receive-side statistics for the link `net1`. In addition, with the use of the `-F` option in the command, the output also provides fanout information. Specifically, the fanout count is two (`0` and `1`). Network traffic that is received on the hardware lane that uses ring `0` is split and passed on across the two fanouts. Likewise, network traffic that is received on the hardware lane that uses ring `1` is also split and divided across the two fanouts.

```
# dlstat show-link -r -F net1
LINK     ID   INDEX   FOUT   IPKTS
net1  local      --      0       0
net1     hw       0      0  382.47K
net1     hw       0      1        0
net1     hw       1      0  367.50K
net1     hw       1      1  433.24K
```

Obtaining Network Traffic Statistics on Link Aggregations

The dlstat show-aggr command shows network packet statistics for each aggregation's ports when traffic traverses the aggregation on the system.

```
# dlstat show-aggr
 LINK            PORT    IPKTS   RBYTES    OPKTS   OBYTES
aggr1              --        0        0        0        0
aggr1            net0        0        0        0        0
aggr1            net1        0        0        0        0
```

The output indicates the configuration of a link aggregation aggr1 with two underlying links, net0 and net1. As network traffic is received or sent by the system through the aggregation, information about incoming and outgoing packets and their respective sizes is reported for every port. The ports are identified by the underlying links of the aggregation.

Gathering Statistics About Network Traffic on Flows

Flow statistics help you evaluate packet traffic on any defined flows on the system. To obtain flow information, you use the flowstat command. For more information about this command, refer to the flowstat(1M) man page.

The most commonly used syntax of the flowstat command follows:

# flowstat [-r|-t] [-i *interval***] [-l** *link***] [***flow***]**

[-r|-t] Displays either receive-side statistics only (-r option) or transmit-side statistics only (-t option). To display statistics for both the receive-side and the transmit-side, omit the option.

-i *interval* Specifies the time in seconds at which you want the displayed statistics to be refreshed. If you do not use this option, then static output is displayed.

-l *link* Indicates that you want to monitor the statistics for all the flows on the specified datalink. If you do not use this option, then information about all the flows on all the datalinks is displayed.

flow Indicates that you want to monitor the statistics of a specified flow only. If you do not use this option, then depending on whether you specified a link, all flow statistics are displayed.

The following examples show different ways to display information about configured flows on the system.

EXAMPLE 4–1 Displaying Traffic Statistics for All Flows at One-Second Intervals

This example shows information every second about incoming and outgoing traffic on all configured flows on the system.

```
# flowstat -i 1
FLOW     IPKTS   RBYTES  IERRS    OPKTS  OBYTES  OERRS
flow1  528.45K  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
...
^C
```

This example shows statistics about outgoing traffic for all configured flows.

EXAMPLE 4–2 Displaying Transmit-Side Statistics for All Flows

```
# flowstat -t
 FLOW    OPKTS   OBYTES   OERRS
flow1   24.37M    1.61G       0
flow2        0        0       0
flow1        4      216       0
```

EXAMPLE 4–3 Displaying Receive-Side Statistics for All Flows on a Specified Link

This example shows incoming traffic in hardware lanes on all the flows that were created over the datalink `net0`.

```
# flowstat -r -i 2 -l net0
    FLOW    IPKTS   RBYTES    IERRS
tcp-flow  183.11K  270.24M        0
udp-flow        0        0        0
tcp-flow  373.83K  551.52M        0
udp-flow        0        0        0
tcp-flow  372.35K  549.04M        0
udp-flow        0        0        0
tcp-flow  372.87K  549.61M        0
udp-flow        0        0        0
tcp-flow  371.57K  547.89M        0
udp-flow        0        0        0
tcp-flow  191.92K  282.95M        0
udp-flow  206.51K  310.70M        0
tcp-flow        0        0        0
udp-flow  222.75K  335.15M        0
tcp-flow        0        0        0
```

EXAMPLE 4–3 Displaying Receive-Side Statistics for All Flows on a Specified Link *(Continued)*

```
udp-flow  223.00K  335.52M       0
tcp-flow        0        0       0
udp-flow  160.22K  241.07M       0
tcp-flow        0        0       0
udp-flow  167.89K  252.61M       0
tcp-flow        0        0       0
udp-flow    9.52K   14.32M       0
^C
```

Configuring Network Accounting for Network Traffic

You can use the extended accounting facility to set up network accounting on the system. Network accounting involves capturing statistics about network traffic in a log file. In this manner, you can maintain records of traffic for tracking, provisioning, consolidation, and billing purposes. Later, you can refer to the log file to obtain historical information about network use over a period of time.

To set up network accounting, you use the extended accounting facility's `acctadm` command. After you have completed setting up network accounting, you use the `flowstat` command to record traffic statistics.

This section describes the following procedures:

- "How to Set Up Network Accounting" on page 60
- "How to Obtain Historical Statistics on Network Traffic" on page 62

▼ How to Set Up Network Accounting

1. **On the system with the interfaces whose network usage you want to track, become an administrator.**

   For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

2. **View the status of the accounting types that can be enabled by the extended accounting facility.**

   ```
   # acctadm [process | task | flow | net]
   ```

   The extended accounting facility can enable four types of accounting. The optional operands of the `acctadm` command correspond to these accounting types. You use an operand with the command to configure a specific type of accounting.

   - Process accounting
   - Task accounting

- Flow accounting for IPQoS
- Network accounting links and flows

Note – Network accounting also applies to flows that are managed by the `flowadm` and `flowstat` commands as discussed in "Managing Resources on Flows" on page 44. Therefore, to set up accounting for these flows, use the `net` option with the `acctadm` command. Do *not* use the `flow` option, which enables flow accounting for IPQoS configurations.

Specifying `net` displays the status of network accounting. If `net` is not used, then the status of all four accounting types is displayed.

---

3 **Enable extended accounting for network traffic.**

```
# acctadm -e extended -f filename net
```

where *filename* includes the full path of the log file that will capture network traffic statistics. The log file can be created in any directory that you specify.

4 **Verify that extended network accounting has been activated.**

```
# acctadm net
```

Example 4–4 Setting Up Network Accounting on the System

This example shows how to configure network accounting to capture and display historical traffic information on the system.

First, view the status of all accounting types as follows:

```
# acctadm
            Task accounting: inactive
       Task accounting file: none
     Tracked task resources: none
   Untracked task resources: extended
         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
            Network accounting: inactive
       Network accounting file: none
     Tracked Network resources: none
   Untracked Network resources: extended
```

The output shows that network accounting is not active.

Next, enable extended network accounting.

```
# acctadm -e extended -f /var/log/net.log net
# acctadm net
            Net accounting: active
       Net accounting file: /var/log/net.log
     Tracked net resources: extended
   Untracked net resources: none
```

▼ How to Obtain Historical Statistics on Network Traffic

After you have enabled network accounting, you can use the `dlstat` and `flowstat` commands to extract information from the log file. This procedure describes the steps.

Before You Begin You must enable extended accounting for the network before you can display historical data about the network. Further, to display historical data about traffic on flows, you must first configure flows on the system as explained in "Managing Resources on Flows" on page 44.

1 On the system with the interfaces whose network usage you want to track, become an administrator.

For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

2 To extract and display historical information about resource usage on datalinks, use the following command:

```
# dlstat show-link -h [-a] -f filename [-d date] [-F format] [-s start-time] [-e end-time] [link]
```

`-h`	Displays a summary of historical information about resource usage by incoming and outgoing packets on datalinks.
`-a`	Displays resource usage on all datalinks, including those that have already been deleted after the data capture.
`-f` *filename*	Specifies the log file that was defined when network accounting was enabled with the `acctadm` command.
`-d` *date*	Displays logged information for the specified date.
`-F` *format*	Displays the data in a specific format that can then be plotted for analysis. Currently, `gnuplot` is the only supported format.
`-s` *start-time*, `-e` *end-time*	Displays available logged information for a specified date and time range. Use the `MM/DD/YYY,hh:mm:ss` format. The `hour` (`hh`) must use the 24-hour clock notation. If you do not include the date, then data for the specified time range for the current date is displayed.
link	Displays historical data for a specified datalink. If you do not use this option, then historical network data for all configured datalinks is displayed.

3 To extract and display historical information about network traffic on configured flows, use the following command:

`# `**`flowstat -h [-a] -f`** *filename* **`[-d`** *date*`]` **`[-F`** *format*`]` **`[-s`** *start-time*`]` **`[-e`** *end-time*`]` **`[`***flow***`]`**

`-h`	Displays a summary of historical information about resource usage by incoming and outgoing packets on configured flows.
`-a`	Displays resource usage on all configured flows, including those that have already been deleted after the data capture.
`-f` *filename*	Specifies the log file that was defined when network accounting was enabled with the `acctadm` command.
`-d`	Displays logged information for the specified date.
`-F` *format*	Displays the data in a specific format. Currently, `gnuplot` is the only supported format.
`-s` *start-time*, `-e` *end-time*	Displays available logged information for a specified date and time range. Use the `MM/DD/YYY,hh:mm:ss` format. The `hour` (`hh`) must use the 24-hour clock notation. If you do not include the date, then data for the specified time range for the current date is displayed.
flow	Displays historical data for a specified flow. If you do not use this option, then historical network data for all configured flows is displayed.

Example 4–5 Displaying Historical Information About Resource Usage on Datalinks

The following example shows historical statistics about network traffic and its use of resources on a specified datalink:

```
# dlstat show-link -h -f /var/log/net.log net0
 LINK  DURATION  IPACKETS RBYTES      OPACKETS OBYTES      BANDWIDTH
net0   80        1031     546908      0        0           2.44 Kbps
```

Example 4–6 Displaying Historical Information About Resource Usage on Flows

The following examples show different ways of displaying historical statistics about network traffic on a flow and its use of resources.

The following example displays historical statistics of resource usage by traffic on a flow:

```
# flowstat -h -f /var/log/net.log
FLOW      DURATION  IPACKETS RBYTES      OPACKETS OBYTES      BANDWIDTH
flowtcp   100       1031     546908      0        0            43.76Kbps
flowudp   0         0        0           0        0            0.00Mbps
```

The following example displays historical statistics of resource usage by traffic on a flow over a given date and time range:

```
# 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
```

The following example displays historical statistics of resource usage by traffic on a flow over a given date and time range. The information is displayed by using the `gnuplot` format.

```
# 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
10:40:06 0.00
```

Index

A
`acctadm` command, 60–64
attributes, flows, 12

B
bandwidth
    on flows, 14
    setting on flows, 45
    VNICs, 13

C
CPU pools, 13, 40
    assigning to links, 42
    default pool, 40
CPUs, 13, 40
    CPU allocation, 43–44

D
data center consolidation, 10
datalinks
    resource control properties, 11–12, 13
datalinks, resource control properties, 13
dedicated CPUs for interfaces, 43–44
`dladm` command
    `create-etherstub`, 11, 16
    `create-vlan`, 32
    `create-vnic`, 10–11, 13, 16

`dladm` command *(Continued)*
    `delete-vnic`, 29–30
    `modify-vnic`, 26
    `set-linkprop`, 11, 13
    `show-linkprop`, 33–34
    `show-phys`, 34
    `show-vnic`, 29
`dlstat` command, 51, 54
    `show-aggr`, 58
    `show-link`, 56–58
    `show-phys`, 55–56
DS field, 12
dynamic ring grouping, 31–39

E
etherstubs, 8–10, 15–16
    creating, 11, 16
    private virtual networks, 23–25
extended accounting facility, 60–64
    flow accounting for IPQoS, 61
    network accounting for links and flows, 61
    process accounting, 60
    task accounting, 60
external virtual networks, 7–11

F
flow control, 44–49
`flowadm` command, 44–49
    `add-flow`, 13, 45

`flowadm` command *(Continued)*
  `help`, 14
  `set-flowprop`, 13, 45
  `show-flowprop`, 46
flows, 12, 44–49
  attributes, 12
  bandwidth setting, 14, 45
  creating, 13, 45
  displaying information, 45
  setting properties, 13
`flowstat` command, 51, 54, 58–60

G
GARP VLAN Registration Protocol (GVRP), 17

H
hardware-based clients, 31
hardware rings, 31–39

I
internal virtual networks, 7–11
interrupts, 44, 57
`ipadm` command
  `create-addr`, 16
  `create-ip`, 16
  displaying IP information, 47

M
MAC address, on VNICs, 8
MAC clients, 31
  allocating rings, 38–39
  configuring, 38–39
  hardware-based, 31
  primary, 32, 37
  secondary, 32, 37
  software-based, 31
managing network resources, 31
migrating VNICs, 27–28
monitoring network use, 51

N
network accounting, 60–64
network lanes, 11, 12
  hardware lanes, 51
  software lanes, 51
network resource management, 31
  by using flows, 12
  CPU pools, 40
  CPUs, 40
  `dladm` subcommands for implementation, 13
  on links, 11
network statistics, 54–58
  historical traffic information, 61, 62–64
  monitoring network use, 51
  network traffic on lanes, 56–58
  resource usage on flows, 58–60
  traffic on aggregation ports, 58
  use of devices for network traffic, 55–56
network virtualization, 7–11
  components, 8–10
  data center consolidation, 10
  `dladm` subcommands for implementation, 13
  etherstubs, 8–10
  virtual switches, 8–10
  VNICs, 8–10
NIC rings, 13
  See also ring grouping

P
packet drops, 57
primary client, 32, 37
private virtual networks, 7–11
  configuring with etherstubs, 23–25
pseudo Ethernet NICs, *See* etherstubs

Q

quality of service (QoS), 11–12

R

resource control, 11–12, 31
ring allocation, *See* ring grouping
ring grouping
  datalink properties for, 33
  displaying information about, 33
  dynamic and static, 31–39
  interpreting ring information, 34, 35, 36
  receive and transmit rings, 33–34
  ring use and ring assignments, 36–38
  steps to allocate, 38–39
  support in NICs, 34–36
  VLANs, on, 32
rings, transmit and receive, 31–39

S

SCTP, 12
secondary clients, 32, 37
service level agreements (SLAs), 12
software-based clients, 31
static ring grouping, 31–39

T

TCP, 12
transport protocols, 12

U

UDP, 12

V

virtual network cards, *See* VNICs
virtual networks
  See also VNICs
  building, 18–25
  configuring a VNIC, 15–16
  configuring zones for, 19–21, 21–22
  etherstubs, 15–16
  exclusive IP-type zones, 22
  external and internal, 7–11
  private, 7–11
  using with zones, 9
virtual switch, 8–10
virtualization and quality of service, 31
VLANs
  as VNICs, 17–18
  modifying the VLAN ID of a VNIC, 26
VNICs, 7–11
  assigning CPU pool resources, 42
  assigning to a zone, 21
  bandwidth, 13
  changing the underlying link, 27–28
  configuring, 19–21
  creating, 10–11, 16
  creating a VNIC IP interface, 16
  deleting, 29–30
  displaying information, 29
  migrating, 27–28
  modifying MAC addresses, 26–27
  modifying VLAN IDs, 26
  private virtual networks, 23–25
  setting properties, 13
  using with zones, 21–22
  VLAN IDs for, 10
  with VLAN IDs, 17–18

Z

zones
  assigning VNICs, 21
  configuring for network virtualization, 19–21
  configuring for virtual networks, 21–22