Oracle® Traffic Director

Configuration File Reference

11g Release 1 (11.1.1.9)

E21038-05

December 2016

ORACLE®

Oracle Traffic Director Configuration File Reference 11g Release 1 (11.1.1.9)

E21038-05

Copyright © 2011, 2016, Oracle and/or its affiliates. All rights reserved.

Primary Author: Rajesh Gouthaman

Contributing Authors: Kumar Dhanagopal

Contributors: Arvind Srinivasan, Basant Kukreja

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are "commercial computer software" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information about content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Oracle.

Contents

3 Elements in server.xml

4 Syntax and Use of obj.conf

5 Predefined Server Application Functions and Filters in obj.conf

List of Tables

Preface

This document describes the purpose and use of the configuration files for Oracle Traffic Director, including `server.xml`, and `obj.conf`. It provides a comprehensive list of the elements and directives in these configuration files.

Audience

The intended audience for this document is the person who administers and maintains Oracle Traffic Director.

This document assumes you are familiar with the following topics:

- Working in a terminal window
- HTTP
- XML
- Executing operating system commands on UNIX-like platforms

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc`.

Access to Oracle Support

Oracle customers that have purchased support 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.

Related Documents

For more information, see the following documents, which are available on the Oracle Technology Network:

- *Oracle Traffic Director Installation Guide*
- *Oracle Traffic Director Command-line Reference*
- *Oracle Traffic Director Administrator Guide*
- *Oracle Virtual Assembly Builder User's Guide*

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

1

Overview of Configuration Files and Directories

The configuration and behavior of Oracle Traffic Director is determined by a set of configuration files. You can use the Oracle Traffic Director (OTD) Administrator Console and the command-line interface (CLI) to change the configuration file settings. You can also manually edit these files.

This chapter includes the following topics:

- Configuration Files
- Directory Structure
- Dynamic Reconfiguration

1.1 Configuration Files

Each server instance has its own directory, called `INSTANCE_HOME` in this document. The `INSTANCE_HOME/config` directory contains configuration files for the Oracle Traffic Director components. The exact number and names of the configuration files depend on the components that were enabled or loaded into the server. For the default location of the `INSTANCE_HOME`, see Section 1.1.4, "Default Paths".

The following sections describe the configuration files and related information pertaining to the Oracle Traffic Director:

- The server.xml File
- The obj.conf File
- The certmap.conf File

1.1.1 The server.xml File

The `server.xml` file contains the Oracle Traffic Director configuration. For more information about the `server.xml` file, see Chapter 2, "Syntax and Use of server.xml".

1.1.2 The obj.conf File

The `obj.conf` file contains directives for HTTP request processing. For more information about the `obj.conf` file, see Chapter 4, "Syntax and Use of obj.conf."

1.1.3 The certmap.conf File

The `certmap.conf` file describes how a certificate is mapped to an LDAP entry designated by `issuerDN`.

1.1.3.1 Syntax

```
certmap name issuerDN
name:property1 [value1]
name:property2 [value2]
...
```

The default certificate is named `default`, and the default *issuerDN* is also named `default`. Therefore, the first line defined in the `certmap.conf` file must be as follows:

```
certmap default default
```

Use `#` at the beginning of a line to indicate a comment.

1.1.3.2 Properties

Table 1–1 describes the properties of `certmap.conf` file.

Table 1–1 certmap.conf properties

Attribute	Allowed Values	Default Value	Description
`DNComps`	See description	Commented out	Used to form the base DN for performing an LDAP search while mapping the certificate to a user entry. Values are as follows: ■ Commented out - Takes the user's DN from the certificate as is ■ Empty - Searches the entire LDAP tree (DN == suffix) ■ Comma-separated attributes - Forms the DN
`FilterComps`	See description	Commented out	Used to form the filter for performing an LDAP search while mapping the certificate to a user entry. Values are as follows: ■ Commented out or empty - Sets the filter to `"objectclass=*"` ■ Comma-separated attributes - Forms the filter
`verifycert`	`on` or `off`	`off` (commented out)	Specifies whether certificates are verified.
`CmapLdapAttr`	Name of the LDAP attribute	`certSubjectDN` (commented out)	Specifies the name of the attribute in the LDAP database that contains the DN of the certificate.
`library`	Path to shared `lib` or `dll`	None	Specifies the library path for custom certificate mapping code.
`InitFn`	Name of initialization function	None	Specifies the initialization function in the certificate mapping code referenced by `library`.

1.1.4 Default Paths

The default paths that are used in Oracle Traffic Director are listed below. Table 1–2 describes the default paths and file names.

Table 1–2 Default Paths

Placeholder	Description
ORACLE_HOME	The directory in which Oracle Traffic Director is installed.
INSTANCE_HOME	The directory that contains the files pertaining to the Oracle Traffic Director administration server and Oracle Traffic Director instances.

1.2 Directory Structure

This section describes the directory structures that are created when you first install Oracle Traffic Director and setup the installation. In a typical OTD installation, except for the admin-server directory and net-*server-id* instance subdirectory, all other directories are located in the ORACLE_HOME directory. The admin-server directory and the net-*server-id* subdirectory are located in the INSTANCE_HOME directory. Creating an administration server or an administration node requires you to execute the configure-server command. To create an administration node in OTD, you must make sure that the administration server is up and running. The newly created administration node will be registered with the administration server. Before creating any instances, create an administration server by running the configure-server command. Later, execute the create-instance command to create instances in OTD. For more information about the CLI commands, see *Oracle Traffic Director Command-line Reference*. For more information about the default locations for these directories, see Section 1.1.4, "Default Paths".

The following are the directories under the Oracle Traffic Director installation directory, ORACLE_HOME:

- bin
- Legal
- lib
- admin-server
- net-server-id

1.2.1 bin

The bin directory contains the commands to administer Oracle Traffic Director, including the command-line interface (tadm).

1.2.2 Legal

The Legal directory contains the third party information pertaining to the Oracle Traffic Director software.

1.2.3 lib

The lib directory contains internal binaries, scripts, libraries, and bundled plug-ins. These files are private files, for internal use only.

Information pertaining to the `admin-server` directory and `net-`*`server-id`* subdirectory are listed below.

- admin-server
- net-server-id

1.2.4 admin-server

The `admin-server` directory contains the following subdirectories:

- `bin` - Contains the binary files that are required to start, stop, and restart Oracle Traffic Director. On UNIX systems, this directory also contains the file required for rotating logs.
- `config` - Contains the private configuration files for the administration server. These files are for internal use.
- `config-store` - Contains files used by the administration server to track server configuration information.

  Note: The files in this directory are created by Oracle Traffic Director for internal use. Do not edit, run scripts on, or otherwise access any files in the `config-store` directory.

- `generated` - Contains files generated by the instance, such as Java class files corresponding to JavaServer Pages (JSP).
- `logs` - Contains any error messages or access log files that are generated by a server instance.

1.2.5 net-server-id

An `net-`*`server-id`* directory is created for every instance you create in OTD. This directory has the following subdirectories and files:

- `bin` - Contains the commands for starting, stopping, restarting, and reconfiguring the server. It also contains the command for rotating the log files.
- `config` - Contains the following instance-specific configuration files:
  - `cert9.db` - NSS certificate database.
  - `<C1>-obj.conf` - Virtual server specific directory.
  - `key4.db` - NSS private key database.
  - `obj.conf` - Instructions for Oracle Traffic Director for handling HTTP requests from clients.
  - `pkcs11.txt` - NSS PKCS #11 module database.
  - `server.xml` - Most of the server configuration settings.
- `logs` - Contains log files generated by this server instance.

1.3 Dynamic Reconfiguration

Dynamic reconfiguration enables you to make configuration changes to a runtime Oracle Traffic Director. You do not have to stop or restart the Oracle Traffic Director for the changes to take effect.

Dynamic configuration happens in one of the following ways:

- When you deploy a configuration through the Administration Console or CLI
- When you run the `reconfig` script in the server instance's `bin` directory

You can dynamically change the configuration settings in the `obj.conf` file without restarting the server. In addition, most settings in the `server.xml` file can be changed without restarting the server. If you must restart the server, a warning message appears in the server log when you deploy the configuration or run the `reconfig` command.

You cannot dynamically reconfigure the following `server.xml` configuration parameters:

- `user`
- `temp-path`
- `log` (with the exception of log-level)
- `thread-pool`
- `pkcs11`
- `stats`
- `dns`
- `dns-cache`
- `ssl-session-cache`
- `access-log-buffer`

When you run the `reconfig` command, a new configuration object is created, and all new incoming requests are processed based on this new configuration object. The current configuration object gets removed when no HTTP requests are using the object.

In case a wrong configuration occurs during dynamic reconfiguration, the server displays an error message. The server logs the error message to a log file specified by the last configuration that worked.

Certain wrong configurations result in warning messages but do not cause the server to reject the configuration. Other wrong configurations result in error messages and cause the server to reject the configuration. If the server rejects a configuration during startup, the server does not start. If the server rejects a configuration during a dynamic reconfiguration, the server reverts to the last configuration that worked.

2

Syntax and Use of server.xml

The `server.xml` file contains most of the server configuration. This chapter describes the basic syntax of the `server.xml` file and provides a high-level view of the elements that are used to configure features of the server. This chapter contains the following topics:

2.1 Overview of server.xml

The `server.xml` file contains the elements that define the configuration. The `server.xml` file is located in the `INSTANCE_HOME/net-server-id/config` directory.

The encoding is UTF-8 to maintain compatibility with UNIX text editors.

2.1.1 Editing the server.xml File

The structure of the `server.xml` file is a hierarchy, with `server` as the topmost element. The `server` element has many subelements, many of which have subelements of their own.

In general, you do not need to edit `server.xml` directly. Instead, use the Administrator Console and the `tadm` command-line interface to change values in the `server.xml` file. The changes that are made using the Administrator Console and `tadm` command-line interface affects the `server.xml` file. Using `tadm` when creating scripts to change the `server.xml` file ensures forward compatibility. If you edit the `server.xml` file directly, ensure that the resulting `server.xml` file is valid.

2.1.1.1 Editing Element Values

To change the values in the `server.xml` file, change the value between the tags associated with the element you are editing. For example, to change the value of <log-level> from `NOTIFICATION:1` to `TRACE:1`, find the `log` child element of the `server` element. In this example, you see the following lines:

```
<log>
  <log-file>../logs/server.log</log-file>
  <log-level>NOTIFICATION:1</log-level>
</log>
```

`For example:`

Changing the log-level from `NOTIFICATION:1` to `TRACE:1` is shown below:

```
<log-level>NOTIFICATION:1</log-level>
```

to:

```
<log-level>TRACE:1</log-level>
```

After you make changes to the `server.xml` file, you must deploy your configuration for most changes to take effect. Use the command-line interface command `tadm pull-config` to pull the modified `server.xml` file, then use the Administrator Console or the `tadm deploy-config` command to deploy your changes. For some changes, you must restart the server before they take effect. For information about changes that require a restart and which do not, see Section 1.3, "Dynamic Reconfiguration".

2.1.1.2 Adding Elements

To add a new element to the file, add the element and any required subelements. Elements begin with a tag, for example `<virtual-server>`, and end with the closing tag, for example `</virtual-server>`. The tags are case-sensitive.

2.1.1.3 Validating server.xml

After editing the `server.xml` file, Oracle Traffic Director automatically validates the XML code when you start or dynamically reconfigure a server.

You can also use the `-configtest` option of the `startserv` script to validate your configuration. From the instance's `bin` directory, run:

```
startserv -configtest
```

2.2 Understanding server.xml

To change the `server.xml` file for your environment, you must know which elements contain the relevant settings. The following sections contain brief descriptions of the elements that configure the functional areas:

- Server Pools
- Health check
- High Availability
- HTTP Protocol
- Logging and Monitoring
- Performance Tuning
- SSL, TLS, and PKCS #11
- Variables
- Virtual Servers

For more information about all the `server.xml` elements and their subelements, see Elements in server.xml.

2.2.1 Server Pools

The `origin-server` element defines a member of a server pool. The `origin-server-pool` element configures a pool of origin servers that are used for load balancing requests. An origin server is a back-end server—such as an Oracle WebLogic Server instance or an Oracle iPlanet Web Server instance—to which Oracle Traffic Director should forward requests that it receives from clients, and from which it receives responses. A set of origin servers providing the same service constitute an origin server pool. For more information, see Section 3.1.17, "origin-server-pool", Section 3.1.18, "origin-server", Section 3.1.25, "server"

2.2.2 Health check

The `health-check` element configures the parameters that are used to determine the status of each origin server in an origin-server pool. The `health-check` element is a subelement of the `origin-server-pool` element. For more information, see Section 3.1.8, "health-check" and Section 3.1.17, "origin-server-pool"

2.2.3 High Availability

The `failover-group` element is a grouping of a VIP (Virtual IP), an instance that is designated as the primary server and another instance designated as the backup server. The Active-Passive or Active-Active cluster failover configurations are represented as Failover Groups. The `failover-group` element defines a failover group. For more information, see Section 3.1.7, "failover-group"

2.2.4 HTTP Protocol

The `http` element configures the general HTTP protocol options. The `keep-alive` element configures the HTTP keep-alive connection management. The `http-listener` element configures the ports and IP addresses on which the server listens for new HTTP connections. The `virtual-server` element configures a method by which the server processes the HTTP requests. For more information, see Section 3.1.9, "http", Section 3.1.12, "keep-alive", Section 3.1.10, "http-listener", and Section 3.1.39, "virtual-server".

2.2.5 Logging and Monitoring

The `access-log` element configures the file name and format of access logs. The `access-log-buffer` element configures the frequency of access log updates and ordering of the access log entries. For more information, see Section 3.1.1, "access-log" and Section 3.1.2, "access-log-buffer". For more information about the log file format, see Appendix B, "Using the Custom Access-Log File Format".

The `log` element configures the file name and contents of the server log. The `event` element configures the access log and server log rotation. For more information, see Section 3.1.14, "log" and Section 3.1.6, "event".

The `snmp` element configures Simple Network Management Protocol (SNMP), and the `stats` element configures statistics collection. For more information, see Section 3.1.26, "snmp" and Section 3.1.30, "stats".

2.2.6 Performance Tuning

The `thread-pool` element configures the number of threads used to process requests and the maximum number of HTTP connections that the server queues. For more information, see Section 3.1.35, "thread-pool".

The keep-alive element configures the HTTP keep-alive connection management. For more information, see Section 3.1.12, "keep-alive". The dns-cache element configures the DNS caching. For more information, see Section 3.1.5, "dns-cache".

2.2.7 SSL, TLS, and PKCS #11

The ssl element configures Secure Sockets Layer (SSL) and Transport Layer Security (TLS). SSL and TLS can be configured separately for each HTTP listener. For more information, see Section 3.1.27, "ssl" and Section 3.1.10, "http-listener".

The pkcs11 element configures the PKCS #11 subsystem, including certificate revocation lists (CRLs) and third-party cryptographic modules. For more information, see Section 3.1.19, "pkcs11".

2.2.8 Variables

The variable element defines a variable for use in expressions, log formats, and obj.conf parameters. For more information about the variable element, see Section 3.1.38, "variable". For more information about variable and expression use, see Appendix A, "Using Variables, Expressions, Wildcards, and String Interpolation".

2.2.9 Virtual Servers

The virtual-server element configures the virtual servers. Each virtual server processes HTTP requests from one or more HTTP listeners. The http-listener element configures the HTTP listeners. For more information, see Section 3.1.39, "virtual-server", and Section 3.1.10, "http-listener".

You can define variables within a virtual server using the variable element, as described in Section 2.2.8, "Variables".

2.3 Sample server.xml File

Example 2-1 shows an excerpt from a server.xml file.

Example 2–1 server.xml file

```
<?xml version="1.0" encoding="UTF-8" ?>

<!--
  Copyright (c) 2011, Oracle and/or its affiliates. All rights reserved
-->

<server>
  <cluster>
    <local-host>www.example.com</local-host>
    <instance>
      <host>www.example.com</host>
    </instance>
  </cluster>
  <log>
    <log-file>../logs/server.log</log-file>
    <log-level>NOTIFICATION:1</log-level>
  </log>
  <platform>64</platform>
  <temp-path>/tmp/net-test-8a4af444</temp-path>
  <user>myuser</user>
  <access-log>
```

```
   <file>../logs/access.log</file>
 </access-log>
<http-listener>
  <name>http-listener-1</name>
  <port>1894</port>
  <server-name>www.example.com</server-name>
  <default-virtual-server-name>test</default-virtual-server-name>
 </http-listener>
<virtual-server>
   <name>test</name>
   <http-listener-name>http-listener-1</http-listener-name>
   <host>www.example.com</host>
   <object-file>test-obj.conf</object-file>
</virtual-server>
<origin-server-pool>
   <name>origin-server-pool-1</name>
   <type>http</type>
   <proxy-server>
      <host>www-proxy.example.com</host>
      <port>80</port>
   </proxy-server>
<origin-server>
   <host>www.example.com</host>
   <port>20005</port>
   </origin-server>
  </origin-server-pool>
 </server>
```

3

Elements in server.xml

This chapter describes the elements in the `server.xml` file in alphabetical order.

3.1 List of Elements

This section describes the elements in the `server.xml` file in alphabetical order.

- server
- snmp
- ssl
- ssl3-tls-ciphers
- ssl-session-cache
- stats
- tcp-access-log
- tcp-listener
- tcp-proxy
- tcp-thread-pool
- thread-pool
- time
- token
- variable
- virtual-server
- webapp-firewall-ruleset

3.1.1 access-log

The `access-log` element configures the settings for the access log. This element can appear zero or more times within the `server` element and zero or more times within the `virtual-server` element. For more information, see Section 3.1.25, "server", and Section 3.1.39, "virtual-server".

Table 3–1 describes the subelements of `access-log`.

Table 3–1 `access-log` Subelements

Element	Occurrences	Description
`enabled`	0 or 1	Specifies whether the server writes to this access log. Default Value: `true`.
`name`	0 or 1	The name that uniquely identifies the access log. If you specify a name, the server does not automatically write to this access log. Instead, you explicitly configure this access log in an `obj.conf` `AddLog` directive.
`file`	1	The file name of the access log. If a relative path is used, it is relative to the server's `config` directory, for example, `../logs/access.log`.
`format`	0 or 1	The format of the access log entries. The default format is an extended custom log format. For more information about access log format, see Appendix B, "Using the Custom Access-Log File Format".

See Also: access-log-buffer, event, log

3.1.2 access-log-buffer

The `access-log-buffer` element configures the settings for access log buffering subsystem. This element can appear zero or one time within the `server` element. For more information, see Section 3.1.25, "server".

Table 3–2 describes the subelements of `access-log-buffer`.

Table 3–2 `access-log-buffer` Subelements

Element	Occurrences	Description
`direct-io`	0 or 1	Specifies if the file system cache access log writes. Default value: `false`. It indicates that the file system write to a cache. Setting the value to `true` indicates that the file system should not to write to a cache. The setting is purely advisory; either the server or the operating system may choose to ignore it.
`enabled`	0 or 1	Specifies whether the server buffers the access log entries. Default value: `true`.
`buffer-size`	0 or 1	The size (in bytes) of individual access log buffers. The value can be from 4096 to 1048576.
`max-buffers`	1	Specifies the maximum number of access-log buffers per server. Values: 1 to 65536.
`max-buffers-per-file`	0 or 1	Specifies the maximum number of access-log buffers per access-log file.
`max-age`	0 or 1	The maximum time (in seconds) to buffer a given access log entry. The value can be from 0.001 to 3600.

See Also: access-log, event, log

3.1.3 cluster

The `cluster` element defines the cluster to which the server belongs. This element can appear zero or one time within the `server` element. For more information, see Section 3.1.25, "server".

Table 3–3 describes the subelements of `cluster`.

Table 3–3 `cluster` Subelements

Element	Occurrences	Description
`local-host`	1	Defines the network address of an instance. The value is the `host` value from an `instance` element. For more information, see Section 3.1.11, "instance".
`instance`	1 or more	Defines a member of the server cluster. For more information, see Section 3.1.11, "instance".
`failover-group`	0 or more	Defines the configuration of a failover group. For more information, see Section 3.1.7, "failover-group"

3.1.4 dns

The `dns` element configures how the server uses the domain name system (DNS). This element can appear zero or one time within the `server` element. For more information, see Section 3.1.25, "server".

Table 3–4 describes the subelements of `dns`.

Table 3–4 `dns` Subelements

Element	Occurrences	Description
`enabled`	0 or 1	Specifies whether the server does DNS lookups. Default value: `false`.
`async`	0 or 1	Specifies whether the server uses its own asynchronous DNS resolver, instead of the Operating System's synchronous resolver. Default value: `true`.
`timeout`	0 or 1	Specifies the duration (in seconds) after which the asynchronous DNS lookups should time out. The value can be from 0.001 to 3600.

See Also: dns-cache

3.1.5 dns-cache

The `dns-cache` element configures the DNS cache. This element can appear zero or one time within the `server` element. For more information, see Section 3.1.25, "server".

Table 3–5 describes the subelements of `dns-cache`.

Table 3–5 `dns-cache` Subelements

Element	Occurrences	Description
`enabled`	0 or 1	Specifies whether the server writes to a cache for DNS lookup results. Default value: `true`.
`max-age`	0 or 1	Specifies the duration (in seconds) for which the entries must be kept in the cache. The value can be from 1 to 31536000.
`max-entries`	0 or 1	Specifies the maximum number of DNS lookup results to write to the cache. The value can be from 32 to 32768.

See Also: dns

3.1.6 event

The `event` element configures a recurring event. The element can appear zero or more times within the `server` element. For more information, see Section 3.1.25, "server".

Table 3–6 describes the subelements of `event`.

Table 3–6 `event` Subelements

Element	Occurrences	Description
`enabled`	0 or 1	Specifies whether the event is enabled at run time. Default value: `true`.

Table 3–6 (Cont.) `event` Subelements

Element	Occurrences	Description
`time`	0 or more	Configures a specific time when the event occurs. For more information, see Section 3.1.36, "time".
`interval`	0 or 1	Specifies the interval (in seconds) at which the event occurs. The value can be from 60 to 86400.
`rotate-log`	0 or 1	Rotates the log files. Default value: `false`.
`rotate-access-log`	0 or 1	Rotates the access log files. Default value: `false`.
`command`	0 or more	The command to execute to get an event to run.
`reconfig`	0 or 1	Dynamically reconfigures the server. Default value: `false`.
`restart`	0 or 1	Restarts the server. Default value: `false`.
`description`	0 or 1	The description of the event. The value of this element is in text format.

See Also: access-log, log

3.1.7 failover-group

The `failover-group` element defines a failover group. This element may appear zero or one time within the `cluster` element. For more information, see Section 3.1.3, "cluster"

Table 3–7 describes the subelements of `failover-group`.

Table 3–7 `failover-group` Subelements

Element	Occurrences	Description
`ip`	1	Specifies the virtual IP for the failover group. The value must be unique across failover groups in a configuration.
`network-prefix`	0 or 1	Specifies the subnet mask for the number of bits used to identify the network. Values: positive integer and 24 (max 32) by default for IPV4. Default value: 64 (max 128) for IPV6
`router-id`	0 or 1	Specifies the router identity for the failover-group. The value must be unique across the failover-groups. It is used to identify the router group of all the participating routers for the same VIP. Values are positive integer. Range of values: 1 to 255. Default value: 255.
`failover-instance`	1 or more	Defines the instances that are part of the `failover-group`.
`host`	1	Specifies the hostname of the administration node where the instance has been created. It must match one of the instance or the host elements in the cluster elements.
`priority`	0 or 1	Specifies the priority value for the instance. This value identifies whether the instance is the primary or the backup for the `failover-group`. Values: positive integer. Range of values: 1 to 254. Default value: 250.

Table 3–7 (Cont.) `failover-group` Subelements

Element	Occurrences	Description
`network-interface`	1	Indicates the network interface on the node where this instance is created on which the VIP is moderated.

3.1.8 health-check

The `health-check` element configures the parameters that are used to determine the status of each origin-server in an origin-server pool. This element may appear zero or one time within the `origin-server-pool` element. For more information, see Section 3.1.17, "origin-server-pool"

Table 3–8 describes the subelements of `health-check`.

Table 3–8 `health-check` Subelements

Elements	Occurrences	Description	TCP health check on HTTP servers	TCP health check on TCP servers
`protocol`	0 or 1	Specifies the type of connection—HTTP or TCP—that Oracle Traffic Director should attempt with the origin server to determine its health. Alternatively, specifies an external health check executable. TCP: Oracle Traffic Director attempts to open a TCP connection to each origin server. The success or failure of this attempt determines whether Oracle Traffic Director considers the origin server to be online or offline. HTTP: Oracle Traffic Director sends an HTTP GET or OPTIONS request to each origin server in the pool, and checks the response to determine the availability and health of the origin server. EXTERNAL: Oracle Trafic Director invokes the executable specified in `<command>` for the health check. Default value: `HTTP`.	Valid	Valid; HTTP is not a valid value for `origin-server-pool` elements that specify `tcp` in the `type` subelement.
`interval`	0 or 1	Specifies the time interval (in seconds) between successive health check operations. Default value: 30.	Valid	Valid
`failover-threshold`	0 or 1	Indicates the number of consecutive failures for marking a server down. It is indicated by a positive integer. The maximum possible value is 256. Default value: 3.	Valid	Valid

Table 3–8 (Cont.) `health-check` Subelements

Elements	Occurrences	Description	TCP health check on HTTP servers	TCP health check on TCP servers
`timeout`	0 or 1	Specifies the timeout value for a connection. It is indicated by a positive integer and in seconds. Default value: 5.	Valid	Valid
`command`	0 or 1	Specifies the full path of an external health check executable. You must configure this parameter if the protocol is EXTERNAL	N/A	N/A
`request-method`	0 or 1	Specifies the method that is used during HTTP health check operations. Default value: `OPTIONS`.	Ignored	Ignored
`request-uri`	0 or 1	Specifies the URI that is used for HTTP health check operations. Default value: `"/"`.	Ignored	Ignored
`response-code-match`	0 or 1	Indicates a modified regular expression that is used to specify what type of response status codes are acceptable for a healthy origin server. The expression is a union of three character patterns that contain only digits or 'x'. 'x' represents a digit, for example, the following three expressions are valid: `200, 2xx|304, 1xx|2xx|3xx|4xx`. Also, if the parameter is not specified, all other codes except `5xx` server error are considered acceptable. This is applicable only when protocol is HTTP.	Ignored	Ignored
`response-body-match`	0 or 1	A regular expression that is used to match the HTTP response body to determine the origin server's health. This is applicable only when protocol is HTTP.	Ignored	Ignored
`response-body-match-size`	0 or 1	Specifies the maximum length of the response body that should match. Default value: 2048.	Ignored	Ignored
`dynamic-server-discovery`	0 or 1	Specifies if the server should dynamically discover Oracle WebLogic Server cluster nodes and add them to the pool. Default value: `false`.	Valid for HTTP Health Check	Ignored

3.1.9 http

The `http` element configures the settings for the miscellaneous HTTP protocol options. This element can appear zero or one time within the `server` element. For more information, see Section 3.1.25, "server".

Table 3–9 describes the subelements of `http`.

Table 3–9 `http` Subelements

Element	Occurrences	Description
`version`	0 or 1	Specifies the highest HTTP protocol version the server supports. The default HTTP version string is `HTTP/1.1`.
`server-header`	0 or 1	Specifies the server header information such as server software and version. The default server header is `Oracle-Traffic-Director/11.1.1.6.0`.
`body-buffer-size`	0 or 1	Specifies the maximum size (in bytes) of the request body content that OTD will expose using the $body variable in obj.conf. The value can be from 0 to 2147483647. Default value: 1024. **Note**: All values must specify units. When body-buffer-size is > 0, and an error occurs while reading the body, then $body handles request received in one of the following ways: ▪ If content-length of the request is specified, OTD undefines $body and proceeds the request with empty body. ▪ If transfer encoding of the request is chunked, OTD undefines $body and rejects the request with 408 (protocol_request_timeout). ▪ If transfer encoding of the request is chunked and max-unchunk-size = 0, OTD undefines $body, and request is sent to the original server containing the partial body
`request-header-buffer-size`	0 or 1	Specifies the size (in bytes) of the buffer used to read HTTP request headers. The value can be from 0 to 2147483647.
`strict-request-headers`	0 or 1	Indicates whether the server rejects certain malformed HTTP request headers. Default value: `false`.
`max-request-headers`	0 or 1	Specifies the maximum number of header fields in an HTTP request header. The value can be from 1 to 512.
`output-buffer-size`	0 or 1	Specifies the size (in bytes) of the buffer for HTTP responses. The value can be from 0 to 2147483647.
`max-unchunk-size`	0 or 1	Specifies the maximum size (in bytes) of a chunked HTTP request body that the server will unchunk. The value can be from 0 to 2147483647.

Table 3–9 (Cont.) `http` Subelements

Element	Occurrences	Description
`unchunk-timeout`	0 or 1	Specifies the maximum time (in seconds) that the server waits for a chunked HTTP request body to arrive. The value can be from 0 to 3600, or -1 for no timeout.
`io-timeout`	0 or 1	Specifies the maximum time (in seconds) that the server waits for an individual packet. The value can be from 0 to 3600, or -1 for no timeout.
`request-header-timeout`	0 or 1	Specifies the maximum time (in seconds) that the server waits for a complete HTTP request header. The value can be from 0 to 604800, or -1 for no timeout.
`request-body-timeout`	0 or 1	Specifies the maximum time (in seconds) that the server waits for a complete HTTP request body. The value can be from 0 to 604800, or -1 for no timeout.
`favicon`	0 or 1	Specifies whether the server replies to requests for `favicon.ico` with its own built-in icon file. Default value: `true`.
`etag`	0 or 1	Controls if the server includes an `Etag` header field in its responses. Default value: `true`.
`ecid`	0 or 1	Specifies whether the server generates, propagates, and logs the execution context. The value of the ECID is a unique identifier that can be used to correlate individual events as being part of the same request execution flow. For example, events that are identified as being related to a particular request typically have the same ECID value. However, the format of the ECID string itself is determined by an internal mechanism that is subject to change; therefore, you should not have or place any dependencies on that format. ECID is defined as a part of the execution context. The execution context consists of ECID and RID. You may also refer to the whole execution context, which is the combination of ECID and RID, as just ECID. Default value: `true`.
`websocket-strict-upgrade`	0 or 1	Enables/disables strict RFC 6455 adherence during the WebSocket upgrade request. Default value: `false`.

See Also: http-listener, keep-alive, thread-pool, virtual-server

3.1.10 http-listener

The `http-listener` element configures an HTTP listener. This element can appear zero or more times within the `server` element. For more information, see Section 3.1.25, "server".

Table 3–10 describes the subelements of `http-listener`.

Table 3–10 ***http-listener Subelements***

Element	Occurrences	Description
`enabled`	0 or 1	Specifies whether the HTTP listener is enabled to accept connection requests. Default value: `true`.
`name`	1	Specifies the name that uniquely identifies the HTTP listener.
`ip`	0 or 1	Specifies an IP address to listen. The value of this element is a specific IP address or an asterisk `*` to listen on all IP addresses.
`port`	1	Specifies the port to listen. The value of this element is the port number.
`acceptor-threads`	0 or 1	Specifies the number of threads dedicated to accept connections received by this listener. The value can be from 1 to 128.
`server-name`	1	Specifies the default server name. Tells the server what to put in the host name section of any URLs it sends to the client. This affects URLs the server automatically generates; it doesn't affect the URLs for directories and files stored in the server. This name should be the alias name if your server uses an alias. If a colon and port number are appended, that port will be used in URLs that the server sends to the client. Values: The value can include a scheme (for example, prefix http://) and port suffix (for example, :80)
`blocking-io`	0 or 1	Specifies whether the server uses blocking I/O. Default value: `false`.
`family`	0 or 1	Specifies the socket family that is used to connect to the origin server. Values: `inet`, `inet6`, `inet-sdp`, and default. `inet` and `inet6` represent IPV4 and IPV6 protocols respectively. `inet-sdp` is used for Sockets Direct Protocol (SDP). Default value: `inet`.
`handle-protocol-mismatch`	0 or 1	Indicates whether the server responds to SSL or non-SSL protocol mismatches in client requests. Default value: `true`, meaning the server will attempt to detect SSL or non-SSL protocol mismatches and send an HTTP redirect or SSL alert when a mismatch is detected.
`listen-queue-size`	0 or 1	Specifies the size (in bytes) of the listen queue. The value of this element can be from 1 to 1048576.
`receive-buffer-size`	0 or 1	Specifies the size (in bytes) of the operating system socket receive buffer. The value of this element can be from 1 to 1048576.
`send-buffer-size`	0 or 1	Specifies the size (in bytes) of the operating system socket send buffer. The value of this element can be from 1 to 1048576.
`default-virtual-server-name`	1	Specifies the name of the virtual server that processes request that do not match a host. The value of this element is the `name` value from a `virtual-server` element. For more information, see Section 3.1.39, "virtual-server".
`ssl`	0 or 1	Configures SSL/TLS. For more information, see Section 3.1.27, "ssl".

Table 3–10 (Cont.) `http-listener` Subelements

Element	Occurrences	Description
`description`	0 or 1	Specifies the description of the HTTP listener. The value of this element must be in text format.
`blocking-accept`	0 or 1	Enables/disables blocking of the server listen socket while retaining client end points as non blocking (useful when MaxProcs > 1). Default value: `false`.

See Also: http, keep-alive, virtual-server

3.1.11 instance

The `instance` element defines a member of a server cluster. This element can appear one or more times within the `cluster` element. For more information, see Section 3.1.3, "cluster".

Table 3–11 describes the subelements of `instance`.

Table 3–11 `instance` Subelements

Element	Occurrences	Description
`enabled`	0 or 1	Specifies whether the instance is enabled at run time. Default value: `true`.
`host`	1	The network address of the instance. The value is the host name or the IP address.

See Also: cluster

3.1.12 keep-alive

The `keep-alive` element configures the settings for the keep-alive subsystem. This element can appear zero or one time within the `server` element. For more information, see Section 3.1.25, "server".

Table 3–12 describes the subelements of `keep-alive`.

Table 3–12 `keep-alive` Subelements

Element	Occurrences	Description
`enabled`	0 or 1	Specifies whether the keep-alive subsystem is enabled at runtime. Default value: `true`.
`threads`	0 or 1	Specifies the number of keep alive subsystem threads. The value can be from 1 to 128. Default value: 1.
`max-connections`	0 or 1	Specifies the maximum number of concurrent keep alive connections that the server supports. The value can be from 1 to 1048576. Default value: 200.
`timeout`	0 or 1	Specifies the timeout (in seconds) after which an inactive keep alive connection can be used. The value can be from 0.001 to 3600. Default value: 30 seconds.

Table 3–12 (Cont.) `keep-alive` Subelements

Element	Occurrences	Description
`poll-interval`	0 or 1	Specifies the interval (in seconds) between polls. The value can be from 0.001 to 1. Default value: .001.

See Also: http, http-listener, virtual-server, thread-pool

3.1.13 localization

The `localization` element defines a method by which the server chooses a language with which it presents information to the client. This element may appear zero or one time within the `server` element, and zero or one time within the `virtual-server` element. For more information, see Section 3.1.25, "server", and Section 3.1.39, "virtual-server".

Table 3–13 describes the subelement of `localization`.

Table 3–13 `localization` Subelements

Element	Occurrences	Description
`default-language`	0 or 1	The default language with which the messages and content are displayed. The value is a language tag.
`negotiate-client-language`	0 or 1	Specifies whether the server uses the accept-language HTTP header to negotiate the content language with clients. Default value: `false`.

3.1.14 log

The `log` element configures the logging subsystem. This element can appear zero or one time within the `server` element. For more information, see Section 3.1.25, "server".

Table 3–14 describes the subelements of `log`.

Table 3–14 `log` Subelements

Element	Occurrences	Description
`log-stdout`	0 or 1	Specifies whether the server logs data that applications write to `stdout`. Default value: `true`.
`log-stderr`	0 or 1	Specifies whether the server logs data that applications write to `stderr`. Default value: `true`.
`log-virtual-server-name`	0 or 1	Specifies whether the server includes the virtual server name in log messages. Default value: `false`.
`create-console`	0 or 1	Specifies if the server creates a console window (Windows only). Default value: `false`.
`log-to-console`	0 or 1	Specifies whether the server writes log messages to the console. Default value: `true`.

Table 3–14 (Cont.) `log` Subelements

Element	Occurrences	Description
`log-to-syslog`	0 or 1	Specifies whether the server writes log messages to `syslog`. Default value: `false`.
`archive-command`	0 or 1	This is executed after the server rotates a log file. The program is passed the post-rotation file name of the log file as an argument. A program command line, for example: `gzip`
`log-level`	0 or 1	Specifies the log verbosity for the server as a whole. Values: `INCIDENT_ERROR:1`, `NOTIFICATION:1`, `ERROR:1`, `ERROR:16`, `ERROR:32`, `WARNING:1`, `TRACE:1`, `TRACE:16` `TRACE:32`. Default value: `NOTIFICATION:1`
`log-file`	0 or 1	Specifies the name and location of the log file. Value: User defined name and location. Default value: `../logs/server.log`

See Also: access-log, access-log-buffer, event

3.1.15 maintenance

The `maintenance` element defines a member of a server pool. This element may appear zero or one time within the `origin-server-pool` element. For more information see, Section 3.1.17, "origin-server-pool"

Table 3–15 describes the subelements of `maintenance`.

Table 3–15 `maintenance` Subelements

Element	Occurrences	Description
`enabled`	0 or 1	Indicates the maintenance state of the origin server pool. Supported values are "true", "false". A value of "true" means the maintenance is enabled for the server pool and the server pool does not handle any (sticky or non-sticky) requests. Instead requests are responded to based on the `response-code` and `response-file` configurations. A value of "false" means the maintenance is disabled for the server pool and origin servers of this pool (depending on the state of the origin server) can handle sticky and non-sticky requests.

Table 3–15 (Cont.) `maintenance` Subelements

Element	Occurrences	Description
`response-code`	0 or 1	Specifies the response code of the request when it lands on a maintenance enabled origin server pool. If set to 200, `response-file` is expected to be configured. If `response-file` is not specified, it is considered as a misconfiguration and the server throws an error at start-up. Supported value: 200 or HTTP response codes in the range 400 to 599.
`response-file`	0 or 1	Specifies the absolute path of an HTML file to send to the client when the request lands on a maintenance enabled origin server pool. The file is sent as text/html regardless of its name or actual type. If the file does not exist or is not accessible, the server returns the default 503 response code.

3.1.16 max-fd

The `max-fd` element specifies a configurable upper limit on the file descriptor usage of the OTD server process. The default value of `max-fd` element is 2 million. This means that by default, Oracle Traffic Director does not assume more than 2 million available file descriptors even if the actual file descriptor availability is configured to be higher.

3.1.17 origin-server-pool

The `origin-server-pool` element configures a pool of origin servers that are used for load balancing requests. This element may appear zero or more times within the `server` element. For more information, see Section 3.1.25, "server".

Table 3–16 describes the subelements of `origin-server-pool`.

Table 3–16 `origin-server-pool` Subelements

Element	Occurrences	Description
`name`	1	Specifies the name by which the server pool is identified.
`load-distribution`	0 or 1	The load-balancing method that should be used for distributing requests to the origin-server pool. Values: `round-robin`, `least-connection-count`, `ip-hash`, and `least-response-time`. Default value: `least-connection-count`. For more information about the various load-balancing methods, see the section Modifying an Origin-Server Pool in the *Oracle Traffic Director Administrator's Guide*.

Table 3–16 (Cont.) `origin-server-pool` Subelements

Element	Occurrences	Description
`type`	1	Indicates the kind of requests that are handled by every server in the server pool. Values: `http`, `https` and `tcp`. Default: `http`.
`family`	0 or 1	Specifies the socket family that is used to connect to the origin server. Values: `inet`, `inet6`, `inet-sdp`, and `default`. `inet` and `inet6` represent IPV4 and IPV6 protocols respectively. `inet-sdp` is used for Sockets Direct Protocol (SDP). Default value: `inet`.
`origin-server`	0 or more	Represents an origin server that belongs to the server pool.
`proxy-server`	0 or 1	Specifies an optional HTTP forward proxy server to be associated with the origin server pool so that all member origin servers (of said pool) are communicated with through the configured HTTP forward proxy server.
`health-check`	0 or 1	Specifies the health check settings for the sever pool
`maintenance`	0 or 1	Specifies the maintenance specific configurations, the response code and page to be displayed when the pool is configured to be under maintenance.

3.1.18 origin-server

The `origin-server` element defines a member of a server pool. This element may appear zero or more times within the `origin-server-pool` element. For more information see, Section 3.1.17, "origin-server-pool"

Table 3–17 describes the subelements of `origin-server`.

Table 3–17 `origin-server` Subelements

Element	Occurrences	Description
`host`	1	Specifies the host name or the IP address of the origin server.
`port`	0 or 1	Specifies the port number of the origin server. Value: Integer. 80 is the default port if the origin server pool type is HTTP. 443 is the default port if the origin server pool type is HTTPS.
`weight`	0 or 1	Specifies the load distribution weight for the origin server. The value is an integer. Default value: 1.
`enabled`	0 or 1	Specifies whether requests can be routed to the origin server. Default value: `true`.

Table 3–17 (Cont.) `origin-server` Subelements

Element	Occurrences	Description
`backup`	0 or 1	Specifies whether the origin sever is a backup server. Requests will be sent to the backup origin server only when none of the primary (non-backup) origin servers is available. Default value: `false`.
`max-connections`	0 or 1	Specifies the maximum number of concurrent connections to the server. Values: 0 to 20480. Default value: 0. The value 0 indicates no limit.
`ramp-up-time`	0 or 1	The time (in seconds) that Oracle Traffic Director should take to ramp up the request sending rate to the full capacity of this origin server. Default value: Any positive integer. If `max-connections` is set to 0, `ramp-up-time` is ignored.

3.1.19 pkcs11

The `pkcs11` element configures the PKCS #11 subsystem. This element may appear zero or one time within the `server` element. For more information, see Section 3.1.25, "server".

Table 3–18 describes the subelements of `pkcs11`.

Table 3–18 `pkcs11` Subelements

Element	Occurrences	Description
`enabled`	0 or 1	Specifies whether the server initializes PKCS #11 tokens, prompting for personal identification numbers (PINs) as necessary. Default value: `true` if SSL is enabled and `false` if SSL is not enabled.
`crl-path`	0 or 1	Specifies the directory that contains dynamically updated CRL files. The value is the name of the directory. If a relative path is used, it is relative to the server's `config` directory.
`token`	0 or more	Configures a PKCS #11 token. For more information, see Section 3.1.37, "token".

See Also: ssl, http-listener

3.1.20 pkcs11 bypass

The `pkcs11 bypass` element instructs the NSS to bypass the PKCS#11 layer during SSL/TLS processing, thereby improving performance.

During startup, the server automatically verifies each token, holding a server key, to assess if they support PKCS#11 bypass. If the tokens support bypass in the current configuration the PKCS#11 layer is bypassed; otherwise the bypass is disabled. Thus, the server automatically takes advantage of the performance benefits of `pkcs11 bypass` whenever possible.

In certain unique circumstances, you can disable PKCS#11 bypass manually by using the `server.xml` element `<allow-bypass>`.

```
<pkcs11>
```

```
 <enabled>1</enabled>
 <allow-bypass>0</allow-bypass>
</pkcs11>
```

3.1.21 property

The `property` element defines a `name-value` pair. The effect of defining a property `name-value` pair depends on the context in which the property element appears.

Table 3–19 describes the subelements of `property`.

Table 3–19 `property` Subelements

Element	Occurrences	Description
`name`	1	The name of the property.
`value`	1	The value of the property.
`encoded`	0 or 1	Specifies if the property value was encoded using the uunencode algorithm. Default value: `false`.
`encrypted`	0 or 1	Specifies if the property value is encrypted. Default value: `false`.
`description`	0 or 1	The description of the property.

See Also: variable

3.1.22 proxy-cache

The proxy-cache element configures the HTTP reverse proxy cache configuration. This element can appear zero or one time within the `server` element. For more information, see Section 3.1.25, "server".

Table 3–20 describes the subelements of `proxy-cache`.

Table 3–20 `proxy-cache` Subelements

Element	Occurrences	Description
`enabled`	0 or 1	Specifies whether response caching is enabled. Default value: `true`.
`max-heap-space`	0 or 1	Specifies the maximum number (in bytes) of heap that is used for caching response objects. Values: 0 to 1099511627776 (1024 GB). Default value: 10485760 (10 MB).
`max-heap-object-size`	0 to 1	Specifies the maximum size of objects that should be cached. Objects larger than the specified size are not cached. Values: 0 to 214783647. Default value: 524288 (512 KB).

Table 3–20 (Cont.) `proxy-cache` Subelements

Element	Occurrences	Description
`replacement`	0 to 1	Specifies the algorithm for cache replacement. Values: `lru`, `lfu`, and `false`. Default value: `lru`. ■ lru (Least Recently Used): Oracle Traffic Director discards the least recently used entry first. ■ lfu (Least Frequently Used): Oracle Traffic Director discards the least frequently used entry first. ■ false: Cache replacement is disabled.
`max-entries`	0 to 1	Specifies the maximum number of entries in the cache. The range is 1 to 1073741824. Default value: 1024.

3.1.23 proxy-server

The `proxy-server` element defines an optional HTTP forward proxy server. You can optionally associate an HTTP forward proxy server with an origin server pool so that all member origin servers (of said pool) are communicated with through the configured HTTP forward proxy server.

This element may appear zero or one time within the `origin-server-pool` element. For more information see, Section 3.1.17, "origin-server-pool"

Table 3–21 describes the subelements of `proxy-server`.

Table 3–21 `proxy-server` Subelements

Element	Occurrences	Description
`host`	1	The network address of the HTTP forward proxy server. The value is the host name or the IP address.
`port`	1	The port for the HTTP forward proxy server. The value of this element is the port number.

3.1.24 qos-limits

The qos-limits element configures the QoS limits. This element may appear zero or one time within the `server` element and zero or one time within the `virtual-server` element. For more information, see Section 3.1.25, "server" and Section 3.1.39, "virtual-server".

Table 3–22 describes the subelements of `qos-limits`.

Table 3–22 `qos-limits` Subelements

Element	Occurrences	Description
`enabled`	0 or 1	Specifies whether the QoS limits are enforced at runtime. Default value: `true`.
`max-bps`	0 or 1	Specifies the maximum transfer rate (bytes/second). Range of value: 1 to 2147483647
`max-connections`	0 or 1	Specifies the maximum number of concurrent connections. Range of value: 1 to 1048576

3.1.25 server

The `server` element defines a server. This is the root element, and there can be only one `server` element in the `server.xml` file.

Table 3–23 describes the subelements of `server`.

Table 3–23 `server` Subelements

Element	Occurrences	Description
`cluster`	0 or 1	The server cluster to which the server belongs. For more information, see Section 3.1.3, "cluster".
`log`	0 or 1	Configures the logging subsystem. For more information, see Section 3.1.14, "log,".
`user`	0 or 1	The account the server runs as (UNIX only). The value is the user account. If the server is started as `root`, any UNIX account can be specified. If the server is started by a `non-root` account, only that `non-root` account can be specified.
`temp-path`	0 or 1	The directory where the server stores its temporary files. If a relative path is used, it is relative to the server's `config` directory. The directory must be owned by the account that the server runs as.
`variable`	0 or more	Defines a variable for use in expressions, log formats, and `obj.conf` parameters. For more information, see Section 3.1.38, "variable".
`localization`	0 or 1	Configures localization. For more information, see Section 3.1.13, "localization".
`http`	0 or 1	Configures the HTTP protocol options. For more information, see Section 3.1.9, "http".
`keep-alive`	0 or 1	Configures the HTTP keep-alive subsystem. For more information, see Section 3.1.12, "keep-alive".
`thread-pool`	0 or 1	Configures the HTTP request processing threads. For more information, see Section 3.1.35, "thread-pool".
`pkcs11`	0 or 1	Configures the PKCS #11 subsystem. For more information, see Section 3.1.19, "pkcs11".
`stats`	0 or 1	Configures the statistics collection subsystem. For more information, see Section 3.1.30, "stats".
`dns`	0 or 1	Configures the server's use of DNS. For more information, see Section 3.1.4, "dns".
`dns-cache`	0 or 1	Configures the DNS cache. For more information, see Section 3.1.5, "dns-cache".
`ssl-session-cache`	0 or 1	Configures the SSL/TLS session cache. For more information, see Section 3.1.29, "ssl-session-cache".

Table 3–23 (Cont.) `server` Subelements

Element	Occurrences	Description
`access-log-buffer`	0 or 1	Configures the access log buffering subsystem. For more information, see Section 3.1.2, "access-log-buffer".
`snmp`	0 or 1	Configures SNMP. For more information, see Section 3.1.26, "snmp".
`access-log`	0 or more	Configures an HTTP access log for the server. For more information, see Section 3.1.1, "access-log".
`http-listener`	0 or more	Configures an HTTP listener. For more information, see Section 3.1.10, "http-listener".
`virtual-server`	0 or more	Configures a virtual server. For more information, see Section 3.1.39, "virtual-server".
`event`	0 or more	Configures a recurring event. For more information, see Section 3.1.6, "event".
`origin-server-pool`	0 or more	Configures a pool of origin servers that are used for handling load balancing requests. For more information, see Section 3.1.17, "origin-server-pool"
`proxy-cache`	0 or 1	Defines the HTTP reverse proxy caching configuration mechanism. For more information, see Section 3.1.22, "proxy-cache"
`qos-limits`	0 or 1	Specifies information related to QoS settings. For more information, see Section 3.1.24, "qos-limits"
`tcp-thread-pool`	0 or 1	Configures the TCP request processing threads. For more information, see Section 3.1.34, "tcp-thread-pool"
`tcp-access-log`	0 or 1	Configures TCP access log for the server. For more information, see Section 3.1.31, "tcp-access-log"
`tcp-listener`	0 or more	Configures a TCP listener. For more information, see Section 3.1.32, "tcp-listener"
`tcp-proxy`	0 or more	Configures a TCP service. For more information, see Section 3.1.33, "tcp-proxy"
`webapp-firewall-ruleset`	0 or more	Specifies the path to a file containing the Web Application Firewall (WAF) module rules. For more information, see Section 3.1.40, "webapp-firewall-ruleset"

3.1.26 snmp

The `snmp` element configures the server's SNMP subagent. This element can appear zero or one time within the `server` element. For more information, see Section 3.1.25, "server".

Table 3–24 describes the subelements of `snmp`.

Table 3–24 `snmp` Subelements

Element	Occurrences	Description
`enabled`	0 or 1	Specifies whether the SNMP agent is enabled. If enabled, the SNMP subagent gathers information about the server and passes the information to the master agent. Default value: `true`.
`description`	0 or 1	(Optional) Specifies the description of the server. The value must be in text format.
`organization`	0 or 1	(Optional) Specifies the name of the organization responsible for the server. The value must be in text format.
`location`	0 or 1	(Optional) Specifies the location of the server. The value must be in text format.
`contact`	0 or 1	(Optional) Specifies the contact information of the person responsible for the server. The value must be in text format.

See Also: stats

3.1.27 ssl

The `ssl` element configures the SSL/TLS settings. This element can appear zero or one time within the `http-listener` element. For more information, see Section 3.1.10, "http-listener".

Table 3–25 describes the subelements of `ssl`.

Table 3–25 `ssl` Subelements

Element	Occurrences	Description
`enabled`	0 or 1	Specifies whether SSL support is enabled for the listener. Default value: `true`.
`server-cert-nickname`	0 or more	Specifies the nickname of the certificate that the server presents to the clients. You can specify zero or one RSA certificate, and zero or one ECC certificate.
`ssl3`	0 or 1	Specifies whether SSL3 connections are accepted. Default value: `true`.
`tls`	0 or 1	Specifies whether TLS connections are accepted. Default value: `true`.
`tls11`	0 or 1	Specifies whether TLS 1.1 connections are accepted. Default value: `true`.
`tls12`	0 or 1	Specifies whether TLS 1.2 connections are accepted. Default value: `true`.
`tls-rollback-detection`	0 or 1	Specifies whether the server detects and blocks TLS version rollback attacks. Default value: `true`.

Table 3–25 (Cont.) `ssl` Subelements

Element	Occurrences	Description
`ssl3-tls-ciphers`	0 or 1	Configures the SSL3 and TLS cipher suites. For more information, see Section 3.1.28, "ssl3-tls-ciphers".
`client-auth`	0 or 1	Specifies the method of client certificate authentication. The value can be `required`, `optional`, or `false`. When you choose required option, the server requests the client for a certificate; if the client does not provide a certificate, the connection is closed. When you choose optional option, the server requests the client for a certificate, but does not require it. The connection is established even if the client does not provide a certificate. Default value: `false`. The client authentication is disabled by default.
`client-auth-timeout`	0 or 1	Indicates the duration (in seconds) after which a client authentication handshake fails. The value can be from 0.001 to 3600.
`max-client-auth-data`	0 or 1	Specifies the number of characters of authentication data that the server can buffer. The value can be from 0 to 2147483647.
`tls-session-tickets-enabled`	0 or 1	Specifies whether TLS session Ticket Extension feature is enabled. Default value: `false`.
`strict-sni-vs-host-match`	0 or 1	Server Name Indication (SNI) is a feature that improves the SSL and TLS protocols. It permits the client to request the domain name before the certificate is committed to by the server. This is essential for using TLS in virtual hosting mode. Default value: `false`. If the value is `false`, the default certificate is sent to clients which do not support SNI extension. If the value is `true`, and if for the HTTP listener, at least one of the virtual servers has certificates, then Oracle Traffic Director returns a 403 Forbidden error if any of the following conditions is fulfilled: ■ The client did not send the SNI host extension ■ The request did not have the Host header ■ The SNI host extension sent by the client did not match the Host header in the HTTP

See Also: http-listener, pkcs11, ssl3-tls-ciphers, ssl-session-cache

3.1.28 ssl3-tls-ciphers

The `ssl3-tls-ciphers` element configures SSL3 and TLS cipher suites. This element can appear zero or one time within the `ssl` element. For more information, see Section 3.1.27, "ssl".

Table 3–26 describes the subelements of `ssl3-tls-ciphers`.

Table 3–26 ssl3-tls-ciphers Subelements

Element	Occurrences	Description
SSL_RSA_WITH_RC4_128_SHA	0 or 1	Specifies whether SSL_RSA_WITH_RC4_128_SHA cipher suite is enabled at runtime. Default value: true.
SSL_RSA_WITH_3DES_EDE_CBC_SHA	0 or 1	Specifies whether SSL_RSA_WITH_3DES_EDE_CBC_SHA cipher suite is enabled at runtime. Default value: true.
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA	0 or 1	Specifies whether TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA cipher suite is enabled at runtime. Default value: true.
TLS_ECDH_RSA_WITH_AES_128_CBC_SHA	0 or 1	Specifies whether TLS_ECDH_RSA_WITH_AES_128_CBC_SHA cipher suite is enabled at runtime. Default value: true.
TLS_ECDH_RSA_WITH_RC4_128_SHA	0 or 1	Specifies whether TLS_ECDH_RSA_WITH_RC4_128_SHA cipher suite is enabled at runtime. Default value: true.
TLS_ECDH_RSA_WITH_3DES_EDE_CBC_SHA	0 or 1	Specifies whether TLS_ECDH_RSA_WITH_3DES_EDE_CBC_SHA cipher suite is enabled at runtime. Default value: true.
TLS_ECDH_RSA_WITH_AES_256_CBC_SHA	0 or 1	Specifies whether TLS_ECDH_RSA_WITH_AES_256_CBC_SHA cipher suite is enabled at runtime. Default value: true.
TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA	0 or 1	Specifies whether TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA cipher suite is enabled at runtime. Default value: true.
TLS_ECDH_ECDSA_WITH_RC4_128_SHA	0 or 1	Specifies whether TLS_ECDH_ECDSA_WITH_RC4_128_SHA cipher suite is enabled at runtime. Default value: true.
TLS_ECDH_ECDSA_WITH_3DES_EDE_CBC_SHA	0 or 1	Specifies whether TLS_ECDH_ECDSA_WITH_3DES_EDE_CBC_SHA cipher suite is enabled at runtime. Default value: true.
TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA	0 or 1	Specifies whether TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA cipher suite is enabled at runtime. Default value: true.

Table 3–26 (Cont.) `ssl3-tls-ciphers` Subelements

Element	Occurrences	Description
`TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA`	0 or 1	Specifies whether `TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA` cipher suite is enabled at runtime. Default value: `true`.
`TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256`	0 or 1	Specifies whether `TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256` cipher suite is enabled at runtime. Default value: `true`.
`TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256`	0 or 1	Specifies whether `TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256` cipher suite is enabled at runtime. Default value: `true`.
`TLS_RSA_WITH_AES_128_CBC_SHA`	0 or 1	Specifies whether `TLS_RSA_WITH_AES_128_CBC_SHA` cipher suite is enabled at runtime. Default value: `true`.
`TLS_RSA_WITH_AES_128_CBC_SHA256`	0 or 1	Specifies whether `TLS_RSA_WITH_AES_128_CBC_SHA256` cipher suite is enabled at runtime. Default value: `true`.
`TLS_RSA_WITH_AES_128_GCM_SHA256`	0 or 1	Specifies whether `TLS_RSA_WITH_AES_128_GCM_SHA256` cipher suite is enabled at runtime. Default value: `true`.
`TLS_RSA_WITH_AES_256_CBC_SHA`	0 or 1	Specifies whether `TLS_RSA_WITH_AES_256_CBC_SHA` cipher suite is enabled at runtime. Default value: `true`.
`TLS_RSA_WITH_AES_256_CBC_SHA256`	0 or 1	Specifies whether `TLS_RSA_WITH_AES_256_CBC_SHA256` cipher suite is enabled at runtime. Default value: `true`.
`TLS_ECDHE_ECDSA_WITH_3DES_EDE_CBC_SHA`	0 or 1	Specifies whether `TLS_ECDHE_ECDSA_WITH_3DES_EDE_CBC_SHA` cipher suite is enabled at runtime. Default value: `true`.
`TLS_ECDHE_ECDSA_WITH_RC4_128_SHA`	0 or 1	Specifies whether `TLS_ECDHE_ECDSA_WITH_RC4_128_SHA` cipher suite is enabled at runtime. Default value: `true`.
`TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA`	0 or 1	Specifies whether `TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA` cipher suite is enabled at runtime. Default value: `true`.
`TLS_ECDHE_RSA_WITH_RC4_128_SHA`	0 or 1	Specifies whether `TLS_ECDHE_RSA_WITH_RC4_128_SHA` cipher suite is enabled at runtime. Default value: `true`.
`TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA`	0 or 1	Specifies whether `TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA` cipher suite is enabled at runtime. Default value: `true`.
`TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA`	0 or 1	Specifies whether `TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA` cipher suite is enabled at runtime. Default value: `true`.

Table 3–26 (Cont.) `ssl3-tls-ciphers` Subelements

Element	Occurrences	Description
`TLS_RSA_WITH_CAMELLIA_128_CBC_SHA`	0 or 1	Specifies whether cipher suite is enabled or not. The cipher suite is implicitly disabled if this element is omitted. The cipher suite is enabled if the element is present while the value is not specified. Default value: `false`.
`TLS_RSA_WITH_CAMELLIA_256_CBC_SHA`	0 or 1	Specifies whether cipher suite is enabled or not. The cipher suite is implicitly disabled if this element is omitted. The cipher suite is enabled if the element is present while the value is not specified. Default value: `false`.
`TLS_RSA_WITH_SEED_CBC_SHA`	0 or 1	Specifies whether cipher suite is enabled or not. The cipher suite is implicitly disabled if this element is omitted. The cipher suite is enabled if the element is present while the value is not specified. Default value: `false`

See Also: http-listener, pkcs11, ssl, ssl-session-cache

3.1.29 ssl-session-cache

The `ssl-session-cache` element configures the SSL/TLS session cache. This element can appear zero or one time within the `server` element. For more information, see Section 3.1.25, "server".

Table 3–27 describes the subelements of `ssl-session-cache`.

Table 3–27 `ssl-session-cache` Subelements

Element	Occurrences	Description
`enabled`	0 or 1	Specifies whether the server writes SSL/TLS sessions to the cache. Default value: `true`.
`max-entries`	0 or 1	Specifies the maximum number of SSL/TLS sessions that are written to the cache by the server. The value can be from 1 to 524288.
`max-ssl3-tls-session-age`	0 or 1	Specifies the maximum amount of time (in seconds) a SSL/TLS session is written to the cache. The value can be from 5 to 86400.

See Also: http-listener, pkcs11, ssl, ssl3-tls-ciphers

3.1.30 stats

The `stats` element configures the statistics collection subsystem. This element can appear zero or one time within the `server` element. For more information, see Section 3.1.25, "server".

Table 3–28 describes the subelements of `stats`.

Table 3–28 `stats` Subelements

Element	Occurrences	Description
`enabled`	0 or 1	Specifies whether the server collects the statistics. Default value: `true`.
`interval`	0 or 1	Specifies the interval (in seconds) at which statistics are updated. The value can be from 0.001 to 3600.
`profiling`	0 or 1	Specifies whether the performance buckets used to track NSAPI function execution time are enabled at runtime. Default value: `true`.

See Also: snmp

3.1.31 tcp-access-log

The `tcp-access-log` element configures the settings for the TCP access log. If the `tcp-access-log` element is missing TCP access logging is disabled. For more information, see Section 3.1.25, "server".

Table 3–29 describes the subelements of `tcp-access-log`.

Table 3–29 `tcp-access-log` Subelements

Element	Occurrences	Description
`enabled`	0 or 1	Specifies whether TCP access logging is enabled. If the element is enabled, the server writes a log entry for every request received by TCP listeners. Default value: `true`.
`file`	1	Specifies the filename of the access log file (absolute path or path relative to the server's config directory).

See Also: tcp-thread-pool, tcp-listener, tcp-proxy

3.1.32 tcp-listener

The `tcp-listener` element configures a TCP listener. For more information, see Section 3.1.25, "server".

Table 3–30 describes the subelements of `tcp-listener`.

Table 3–30 `tcp-listener` Subelements

Element	Occurrences	Description
`enabled`	0 or 1	Specifies whether the TCP listener is enabled to accept connection requests. Default value: `true`.
`name`	1	Specifies the name that uniquely identifies the TCP listener.
`ip`	0 or 1	Specifies the IP address to listen. The value of this element is a specific IP address or an asterisk * to listen on all IP addresses.
`port`	1	Specifies the port to listen. The value of this element is the port number.

Table 3–30 (Cont.) `tcp-listener` Subelements

Element	Occurrences	Description
`family`	0 or 1	Specifies the socket family that is used to connect to the origin server. Values: `inet`, `inet6`, `inet-sdp` and `default`. `inet` and `inet6` represent IPV4 and IPV6 protocols respectively. `inet-sdp` is used for Sockets Direct Protocol (SDP). Default value: `inet`.
`acceptor-threads`	0 or 1	Specifies the number of threads dedicated to accept connections received by this listener. The value can be from 1 to 128. Default value: 1 per CPU.
`tcp-proxy-name`	1	Specifies the name of the TCP proxy that processes requests received by the listener.
`listen-queue-size`	0 or 1	Specifies the size (in bytes) of the listen queue. Value: 1 to 1048576.
`receive-buffer-size`	0 or 1	Specifies the size (in bytes) of the operating system socket receive buffer. Value: 1 to 1048576.
`send-buffer-size`	0 or 1	Specifies the size (in bytes) of the operating system socket send buffer. Value: 1 to 1048576.
`ssl`	0 or 1	Configures SSL/TLS. For more information, see Section 3.1.27, "ssl".
`description`	0 or 1	Specifies the description of the TCP listener. The value of this element must be in text format.
`blocking-accept`	0 or 1	Enables/disables blocking of the server listen socket, while retaining client end points as non-blocking (useful when MaxProcs > 1). Default value: `false`.

See Also: tcp-access-log, tcp-thread-pool, tcp-proxy

3.1.33 tcp-proxy

The `tcp-proxy` element is used to support LDAP/T3 listeners. For more information, see Section 3.1.25, "server".

Table 3–31 describes the subelements of `tcp-proxy`.

Table 3–31 `tcp-proxy` Subelements

Element	Occurrences	Description
`enabled`	0 or 1	Specifies whether the TCP service is enabled. Default value: `true`.
`name`	1	A name that uniquely identifies the TCP proxy.
`session-idle-timeout`	0 or 1	Specifies the maximum timeout (in seconds) that the server waits while receiving/sending data Default value: 300
`origin-server-pool-name`	0 or 1	Specifies the name of a server pool that provides the TCP service. The value must be a name value from an `origin-server-pool` element.

See Also: tcp-access-log, tcp-listenertcp-thread-pool

3.1.34 tcp-thread-pool

The `tcp-thread-pool` element configures the threads used to process WebSocket requests and requests received by TCP listeners. For more information, see Section 3.1.25, "server".

Table 3–32 describes the subelements of `tcp-thread-pool`.

Table 3–32 `tcp-thread-pool` Subelements

Element	Occurrences	Description
`enabled`	0 or 1	Specifies whether the pool is enabled. Default value: `true`.
`threads`	0 or 1	Specifies the number of TCP/WebSocket request processing threads. The value can be from 1 to 512. Default value: 1 per CPU.
`max-connections`	0 or 1	Specifies the maximum number of connection pairs that the server will support. The value can be from 1 to 1048576. Default value: the default value is the value of the keep-alive `max-connections` value.
`timeout`	0 or 1	Specifies the idle timeout (in seconds), after which connection pairs will be closed. The value will be overridden by the tcp or WebSocket subsystem. The value can be from 0.001 to 3600. Default value: 300 seconds.
`stack-size`	0 or 1	Specifies the stack size (in bytes) for each thread. The value can be from 8192 to 67108864, or 0. Default value: 32768.
`poll-interval`	0 or 1	Specifies the interval (in seconds) between polls. The value can be from 0.001 to 1. Default value: 0.010 seconds.
`buffer-size`	0 or 1	Specifies the size of the buffer (in bytes), used by each connection for transferring data. The value can be from 1 to 1048576. Default value: 16384.

See Also: tcp-access-log, tcp-listener, tcp-proxy

3.1.35 thread-pool

The `thread-pool` element configures the threads used to process HTTP requests. This element can appear zero or one time within the `server` element. For more information, see Section 3.1.25, "server".

Table 3–33 describes the subelements of `thread-pool`.

Table 3–33 `thread-pool` Subelements

Element	Occurrences	Description
`min-threads`	0 or 1	Specifies the minimum number of HTTP request processing threads. The value can be from 1 to 4096.
`max-threads`	0 or 1	Specifies the maximum number of HTTP request processing threads. The default value is based on the number of processors. For example, if there are 1 or 2 processors, the default value is 256. Similarly, if there are 3 or 4 processors, the default value is 512. The default value is never more than quarter of the maximum number of file descriptors available for the process.

Table 3–33 (Cont.) `thread-pool` Subelements

Element	Occurrences	Description
`stack-size`	0 or 1	Specifies the stack size (in bytes) for HTTP request processing threads. The value can be from 8192 to 67108864.
`queue-size`	0 or 1	Specifies the maximum number of concurrent HTTP connections that can be queued for processing. The value can be from 1 to 1048576.

See Also: http, keep-alive

3.1.36 time

The `time` element schedules when an event occurs. This element can appear zero or more times within the `event` element. For more information, see Section 3.1.6, "event".

Table 3–34 describes the subelement of `time`.

Table 3–34 `time` Subelements

Element	Occurrences	Description
`time-of-day`	1	Specifies the time when the event occurs. The value must be in the `hh:mm` format.
`day-of-week`	0 or 1	Specifies the day of the week. The value can be `Sun`, `Mon`, `Tue`, `Wed`, `Thu`, `Fri`, or `Sat`.
`day-of-month`	0 or 1	Specifies the day of month. The value can be from 1 to 31.
`month`	0 or 1	Specifies the name of the month. The value can be `Jan`, `Feb`, `Mar`, `Apr`, `May`, `Jun`, `Jul`, `Aug`, `Sep`, `Oct`, `Nov`, or `Dec`.

3.1.37 token

The `token` element configures a PKCS #11 token. This element can appear zero or more times within the `pks11` element. For more information, see Section 3.1.19, "pkcs11".

Table 3–35 describes the subelements of `token`.

Table 3–35 `token` Subelements

Element	Occurrences	Description
`enabled`	0 or 1	Specifies whether the server initializes this PKCS #11 token, prompting for a PIN if necessary. Default value: `true`.
`name`	1	The name of the token. The server's built-in token is named `internal`.
`pin`	0 or 1	The PIN required to initialize the token.
`pin-encrypted`	0 or 1	Specifies whether the property value is encoded or not. Default value: `false`.

3.1.38 variable

The `variable` element defines a variable for use in expressions, log formats, and `obj.conf` parameters. This element can appear zero or more times within the `server` element, and zero or more times within the `virtual-server` element. For more information, see Section 3.1.25, "server", and Section 3.1.39, "virtual-server".

Table 3–36 describes the subelements of `variable`.

Table 3–36 List of `variable` Subelements

Element	Occurrences	Description
`name`	1	Specifies the name of the variable. The value must be in text format.
`value`	1	Specifies the value of the variable. The value must be in text format.
`description`	0 or 1	The description of the variable. The value must be in text format.

See Also: property, Using Variables, Expressions, Wildcards, and String Interpolation

3.1.39 virtual-server

The `virtual-server` element configures an HTTP virtual server. Each server typically has at least one virtual server. This element can appear zero or more times within the `server` element. For more information, see Section 3.1.25, "server".

Table 3–37 describes the subelements of `virtual-server`.

Table 3–37 `virtual-server` Subelements

Element	Occurrences	Description
`enabled`	0 or 1	Specifies whether the virtual server is enabled at runtime. Default value: `true`.
`name`	1	A name that uniquely identifies the virtual server.
`http-listener-name`	0 or more	The name of a HTTP listener associated with one or more of the virtual server's host name. The value is the name from an `http-listener` element. For more information, see Section 3.1.10, "http-listener".
`host`	0 or more	Indicates the host name that the virtual-server services. The values can be a host name or a wildcard. For more information about wildcards, see Section A.5, "Wildcard Patterns"
`canonical-server-name`	0 or 1	The canonical name of the virtual server. Requests using a different name are redirected to the canonical name. The value is a host name or URL prefix.
`object-file`	1	The `obj.conf` file that controls request processing for virtual server. Default value: `default-virtual-server-name-obj.conf`, and the user can specify any valid file
`default-object-name`	0 or 1	The name of the root `obj.conf` object. Default value: `default`.
`localization`	0 or 1	Configures localization. For more information, see Section 3.1.13, "localization".

Table 3–37 (Cont.) `virtual-server` Subelements

Element	Occurrences	Description
`access-log`	0 or more	Configures an HTTP access log for the virtual server. For more information, see Section 3.1.1, "access-log".
`log-file`	0 or 1	Specifies the log file for the virtual server. The value is the log file name, for example, `../logs/errors`.
`variable`	0 or more	Defines an `obj.conf` variable for the virtual server. For more information, see Section 3.1.38, "variable".
`description`	0 or 1	The description of the virtual server.
`server-cert-nickname`	0 or 1 RSA certificate or 1 ECC certificate	Specifies the nickname of the certificate that the server presents to the clients. Values: zero or one for RSA and zero or one for ECC
`qos-limits`	0 or 1	Specifies information related to QoS settings.
`webapp-firewall-ruleset`	0 or multiple	Specifies the path to a file containing Web Application Firewall (WAF) rules or configuration.

See Also: http, http-listener, keep-alive

3.1.40 webapp-firewall-ruleset

The `webapp-firewall-ruleset` element configures the path to a web application firewall configuration file, which contains ModSecurity rules/configuration directives. The path may be an absolute path or a relative path. If a relative path is used, it is relative to the server's `config` directory. The file name component may contain wildcard characters to specify multiple files within the given directory.

The `webapp-firewall-ruleset` element may be present at the `virtual-server` level as well as at the `server` level and can appear zero or more times within the `server` and `virtual-server` elements. Configuration settings at the `virtual-server` level take precedence over the `server` level. However some configuration directives can only be specified at the `server` level. The scope of these directives is considered to be Main. Similarly, scope of directives that can be specified at either `server` level or `virtual-server` level is considered to be Any. Note that if a directive with Main scope is specified within the `virtual-server` level configuration file, then an error will be logged and the server will fail to start. For information about the scope of different directives, see the Web Application Firewall section in the *Oracle Traffic Director Administrator's Guide*.

Note: For information about various web application firewall use cases, see the appendix, Web Application Firewall Examples and Use Cases in the *Oracle Traffic Director Administrator's Guide*.

4

Syntax and Use of obj.conf

The `obj.conf` file contains directives for HTTP request processing. The `obj.conf` file is in the *`INSTANCE_HOME/net-configuration_name`*`/config` directory.

During the installation of Oracle Traffic Director, an `obj.conf` file is created. If you configure multiple virtual servers using the Oracle Traffic Director Administrator Console or Command-Line Interface (CLI), separate `obj.conf` files can be created for each virtual server. These files are named *`virtual-server-name-obj.conf`*, where *`virtual-server-name`* is the name of the virtual server.

When changes made through the Administrator Console or CLI do not impact the `obj.conf` file, no new virtual server specific `obj.conf` files are created, and the default `obj.conf` file will be used for all the virtual servers of the configuration.

When changes made through Administrator Console or CLI impact the `obj.conf` file, a new `obj.conf` file is created for each virtual server. The `server.xml` file is immediately updated to reflect the appropriate `obj.conf` file used for each virtual server. If there are two virtual servers, `vs1` and `vs2`, two new virtual server specific `obj.conf` files are created, `vs1-obj.conf` and `vs2-obj.conf`. These new files are updated in the `server.xml` file with the exact `object-file` used by these two virtual servers.

```
<object-file>vs1-obj.conf</object-file>
```

```
<object-file>vs2-obj.conf</object-file>
```

From this point onward, the default `obj.conf` file is neither updated, used, nor deleted. However, if you want to modify the `obj.conf` file for either of the two virtual servers, you should edit the respective file only, not the original `obj.conf` file.

In this document, the `obj.conf` file refers to the `obj.conf` file specified by the `object-file` sub element of the `virtual-server` element in the `server.xml` file.

This chapter describes the `obj.conf` directives; the use of `Object`, `Client`, `If`, `ElseIf`, and `Else` tags; the flow of control in `obj.conf`; and the syntax rules for editing `obj.conf`.

This chapter includes the following topics:

4.1 Request-Handling Process Overview

When you first start Oracle Traffic Director, it performs some initialization tasks and then waits for an HTTP request from a client (such as a browser). When Oracle Traffic Director receives a request, it first selects a virtual server. The `obj.conf` file of the selected virtual server determines how Oracle Traffic Director handles a request.

The `obj.conf` file contains a series of instructions known as directives that tell Oracle Traffic Director what to do at each stage in the request-handling process. These directives are grouped inside `Object` tags. Each directive invokes a function with one or more arguments.

Each directive is applied to a specific stage in the request-handling process. For example, a directive that is applied during the authorization stage in the request-handling process is an `AuthTrans` directive.

4.1.1 Steps in the Request-Handling Process

1. `AuthTrans` (authorization translation)

   Verify the authorization information (such as name and password) sent in the request.

2. `NameTrans` (name translation)

   Translate the logical URI into a local file system path.

3. `PathCheck` (path checking)

   Check the local file system path for validity and check if the requestor has access privileges to the requested resource on the file system.

4. `ObjectType` (object typing)

   Controls the flow of information from Oracle Traffic Director to the origin server and also configures the Oracle Traffic Director to origin server connection attributes.

5. `Input` (prepare to read input)

   Select filters that will process incoming request data read by the `Service` step.

6. `Output` (prepare to send output)

   Select filters that will process outgoing response data generated by the `Service` step.

7. `Route` (request routing)

   Select where to route the request.

8. `Service` (generate the response)

   Generate and return the response to the client.

9. `AddLog` (adding log entries)

   Add entries to log files.

10. `Error` (error handling)

    Send an error message to the client and exit processing. This step is executed only if an error occurs in the previous steps.

4.2 Directives in obj.conf

The directives in the `obj.conf` file invoke functions known as server application functions (SAFs). Each directive calls a function, indicating when to call it and specifying parameters for it.

The syntax of each directive is:

```
Directive fn="function" name1="value1"...nameN="valueN"
```

The value of the function (`fn`) parameter is the name of the SAF to execute. All directives must supply a value for the `fn` parameter; if there is no function, the instruction does nothing. The remaining parameters are the arguments needed by the function, and they vary from function to function.

Parameters can contain references to variables and expressions. The variables can be predefined variables, variables defined at request time using the `set-variable` SAF, or variables defined in the `server.xml` file. For more information about the `set-variable` SAF, see Section 5.12.6, "set-variable". For more information about defining variables in the `server.xml` file, see Section 3.1.38, "variable". For more information about expressions and variables, see Appendix A, "Using Variables, Expressions, Wildcards, and String Interpolation".

Oracle Traffic Director has a set of built-in SAFs that you can use to create and modify directives in the `obj.conf` file. Section 5, "Predefined Server Application Functions and Filters in obj.conf" describes these SAFs in detail.

The `magnus.conf` file contains `Init` directive SAFs that initialize NASPI plug-ins.

4.3 Objects in obj.conf

Directives in the `obj.conf` file are grouped into `Object` tags. The `default` object contains instructions to Oracle Traffic Director about how to process requests by default. Each new object modifies the `default` behavior of the object.

An `Object` tag can contain a `name` or `ppath` attribute. Either parameter can be a wildcard pattern.

Oracle Traffic Director starts handling a request by processing the directives in the `default` object. However, Oracle Traffic Director switches to processing directives in another object after the `NameTrans` stage of the `default` object if either of the following conditions is true:

- The successful `NameTrans` directive specifies a `name` argument.
- The physical path name that results from the `NameTrans` stage matches the `ppath` attribute of another object.

When Oracle Traffic Director is alerted to use an object other than the `default` object, it processes the directives in the other object before processing the directives in the `default` object. For some steps in the process, Oracle Traffic Director stops processing directives in that particular stage (such as the `Service` stage) as soon as one is successfully executed. Whereas for other stages, Oracle Traffic Director processes all directives in that stage, including the ones in the `default` object and those in the additional object. For more information, see Flow of Control in obj.conf.

4.3.1 Objects That Use the name Attribute

If a NameTrans directive in the default object specifies a name argument, Oracle Traffic Director switches to processing the directives in the object of that name before processing the remaining directives in the default object.

4.3.2 Objects That Use the ppath Attribute

When Oracle Traffic Director completes processing the NameTrans directives in the default object, the logical URL of the request is converted to a physical path name. If this physical path name matches the ppath attribute of another object in the obj.conf file, Oracle Traffic Director switches to processing the directives in that object before processing the remaining ones in the default object.

4.3.3 Using the Client, If, Elself, and Else Tags

Additional tags are available to use within the Object tag. These tags give you greater flexibility when invoking directives within an object.

4.3.3.1 Client

The Client tag enables you to limit the execution of a set of directives to requests received from specific clients. Directives listed within the Client tag are executed only when information in the client request matches the parameter values specified.

Table 4–1 describes the parameters for the Client tag.

Table 4–1 Client Tag Parameters

Parameter	Description
browser	The User-Agent string sent by a browser to Oracle Traffic Director.
chunked	A Boolean value set by a client requesting chunked encoding.
code	The HTTP response code.
dns	The DNS name of the client.
internal	The Boolean value indicating internally generated request.
ip	The IP address of the client.
keep-alive	The Boolean value indicating whether the client has requested a keep-alive connection.
keysize	The key size used in an SSL transaction.
match	The match mode for the Client tag. The valid values are all, any, and none.
method	The HTTP method used by the browser.
name	The name of an object as specified in a previous NameTrans statement.
odds	A random value for evaluating the enclosed directive. The value can be a percentage or a ratio (for example, 20% or 1/5).
path	The physical path to the requested resource.
ppath	The physical path of the requested resource.
query	The query string sent in the request.
reason	The text version of the HTTP response code.
restarted	A Boolean value indicating that a request was restarted.

Table 4–1 (Cont.) Client Tag Parameters

Parameter	Description
secret-keysize	The secret key size used in an SSL transaction.
security	An encrypted request.
type	The type of document requested (such as text/html or image/gif).
uri	The URI section of the request from the browser.
urlhost	The DNS name of the virtual server requested by the client. (The value is provided in the Host header of the client request).

variable-headers	Prevents access to a specific site, based on the request by the client, for example:
	`Client variable-headers="Weferer:SKVFVWRKJVZCMHVIBGDA`
	`Service type="image/*" fn="deny-existence"`
	`</Client>`

The Client tag parameter provides greater control when the If directive is executed. In the following example, the odds parameter gives the request a 25% chance of being redirected:

```
<Client odds="25%">
NameTrans fn="redirect"
          from="/Pogues"
          url-prefix="http://pogues.example.com"
</Client>
```

One or more wildcard patterns can be used to specify the Client tag parameter values. Wildcards can also be used to exclude clients that match the parameter value specified in the Client tag. In the following example, the Client tag and the AddLog directive are combined to direct Oracle Traffic Director to log access requests from all clients except those from the specified subnet:

```
<Client ip="*~192.85.250.*">
AddLog fn="flex-log" name="access"
</Client>
```

You can also create a negative match by setting the match parameter of the Client tag to none. In the following example, access requests from the specified subnet are excluded as are all requests to the virtual server example.com:

```
<Client match="none" ip="192.85.250.*" urlhost="www.example.com">
AddLog fn="flex-log" name="access"
</Client>
```

For more information about wildcard patterns, see Section A.5, "Wildcard Patterns".

4.3.3.2 If, ElseIf, and Else

Similar to the Client tag, these tags can only appear inside an Object tag. In addition, these tags can evaluate an expression, then conditionally execute one or more contained directives. For more information, see Section A.1, "If, ElseIf, and Else Tags".

4.4 Flow of Control in obj.conf

Before Oracle Traffic Director can process a request, it must direct the request to the correct virtual server. After the virtual server is determined, Oracle Traffic Director

executes the obj.conf file of the specified virtual server. This section describes how Oracle Traffic Director determines the directives to execute in obj.conf.

4.4.1 AuthTrans

When Oracle Traffic Director receives a request, it executes the AuthTrans directives in the default object to check if the client is authorized to access Oracle Traffic Director. If there is more than one AuthTrans directive, Oracle Traffic Director executes them in sequence until one succeeds in authorizing the user, unless one of them results in an error. If an error occurs, Oracle Traffic Director skips all other directives except for the Error directive.

AuthTrans directives work in conjunction with the PathCheck directives. The AuthTrans directive checks if the user name and password associated with the request are acceptable, but it does not allow or deny access to the request; that is done by the PathCheck directive.

The authorization process is split into two steps to incorporate multiple authorization schemes easily and provide the flexibility to have resources that record authorization information.

When a client initially makes a request, the user name and password are unknown. The AuthTrans directive gets the user name and password from the headers associated with the request. The AuthTrans and PathCheck directives work together to reject the request if they cannot validate the user name and password. When a request is rejected, Oracle Traffic Director displays a dialog box. The client includes the user name and password in the headers and resubmits the request.

4.4.2 NameTrans

Oracle Traffic Director executes NameTrans directives in the default object to associate a named object (for example, an object that specifies the routing rules) with the URL of the requested resource.

Oracle Traffic Director evaluates each NameTrans directive in the default object in turn, until it finds one that can be applied.

Because Oracle Traffic Director might not execute all NameTrans directives, the order in which the directives appear is important.

4.4.2.1 How and When Oracle Traffic Director Processes Other Objects

As a result of executing a NameTrans directive, Oracle Traffic Director might start processing directives in another object. This happens if the NameTrans directive that was successfully executed specifies a name or generates a partial path that matches the name or ppath attribute of another object.

If the successful NameTrans directive assigns a name by specifying a name argument, Oracle Traffic Director starts processing directives in the named object (defined with the object tag) before processing directives in the default object for the rest of the request-handling process.

4.4.3 PathCheck

After converting the logical URL of the requested resource to a physical path name in the NameTrans step, Oracle Traffic Director executes PathCheck directives to verify that the client is allowed to access the requested resource.

If there is more than one PathCheck directive, Oracle Traffic Director executes all directives in the order in which they appear, unless one of the directives denies access. If access is denied, Oracle Traffic Director switches to executing directives in the Error section.

If the NameTrans directive assigned a name or generated a physical path name that matches the name or ppath attribute of another object, Oracle Traffic Director first applies the PathCheck directives in the matching object before applying the directives in the default object.

4.4.4 ObjectType

Assuming that the PathCheck directives approve access, Oracle Traffic Director next executes the ObjectType directives to determine the MIME type of the request. The MIME type has three attributes: type, encoding, and language. When Oracle Traffic Director sends the response to the client, the type, language, and encoding values are transmitted in the headers of the response. The type also frequently helps Oracle Traffic Director to determine which Service directive to execute to generate the response to the client.

If there is more than one ObjectType directive, Oracle Traffic Director applies all directives in the order in which they appear. However, once a directive sets an attribute of the MIME type, further attempts to set the same attribute are ignored. The reason why all ObjectType directives are applied is that one directive can set one attribute, for example type, while another directive sets a different attribute, such as language.

As with the PathCheck directives, if another object was matched to the request as a result of the NameTrans step, Oracle Traffic Director executes the ObjectType directives in the matching object before executing the ObjectType directives in the default object.

4.4.5 Input

The Input directive selects filters that process incoming request data read by the Service step. Input directives are invoked when Oracle Traffic Director or plug-in first attempts to read entity body data from the client. You can add the NSAPI filters that process incoming data by invoking the insert-filter SAF in the Input stage of the request-handling process. NSAPI filters enable a function to intercept and potentially modify the content presented to or generated by another function. The Input directives are executed once per request.

The order of Input fn="insert-filter" and Output fn="insert-filter" directives in obj.conf is important if two or more filters are defined to be in the same location in the filter stack. Filters that were inserted later appear higher than filters that were inserted earlier.

4.4.6 Output

The Output directive selects filters that process outgoing response data generated by the Service step. The Output directive allows you to invoke the insert-filter SAF to install NSAPI filters that process outgoing data. NSAPI filters enable a function to intercept and potentially modify the content presented to or generated by another function. Output directives are executed when Oracle Traffic Director or a plug-in first attempts to write entity body data from the client. The Output directives are executed once per request.

The order of `Input fn="insert-filter"` and `Output fn="insert-filter"` directives in `obj.conf` is important if two or more filters are defined to be in the same location in the filter stack. Filters that were inserted later appear higher than filters that were inserted earlier.

4.4.7 Route

If a `Service` directive requires that the HTTP request be sent to another server, Oracle Traffic Director executes `Route` directives to determine how the request be routed. Routing a request can involve selecting Oracle Traffic Director that will service the request and selecting a proxy through which the request is sent.

4.4.8 Service

Oracle Traffic Director executes a `Service` directive to generate the response to send to the client. Oracle Traffic Director looks at each `Service` directive to find the first one that matches the type, method, and query string. If a `Service` directive does not specify type, method, or query string, then the unspecified attribute matches anything.

If there is more than one `Service` directive, Oracle Traffic Director applies the first one that matches the conditions of the request and ignores all remaining `Service` directives.

For the `PathCheck` and `ObjectType` directives, if another object was matched to the request as a result of the `NameTrans` step, Oracle Traffic Director checks the `Service` directives in the matching object before considering the ones in the `default` object. If Oracle Traffic Director successfully executes a `Service` directive in the matching object, it does not execute the `Service` directives in the `default` object, because it only executes one `Service` directive.

4.4.9 AddLog

After Oracle Traffic Director generates the response and sends it to the client, it executes `AddLog` directives to add entries to the log files. All `AddLog` directives are executed. Oracle Traffic Director can add entries to multiple log files.

4.4.10 Error

If an error occurs during the request-handling process, for example, if a `PathCheck` or `AuthTrans` directive denies access to the requested resource or the requested resource does not exist, the SAF sets the HTTP response status code and returns the value `REQ_ABORTED`. When this happens, Oracle Traffic Director stops processing the request. Instead, it searches for an `Error` directive matching the HTTP response status code or its associated reason phrase and executes the directive's function. If Oracle Traffic Director does not find a matching `Error` directive, it returns the response status code to the client.

4.5 Changes in Function Flow

There are times when the function flow changes from the normal request-handling process. This happens during internal redirection, restarts, and URI translation functions.

4.5.1 Restarted Requests

Requests can be restarted, for example, a `PathCheck` directive might restart a request for `http://`*server_name*`/` as a request for `http://`*server_name*`/index.html`.

4.5.2 Internal Requests

Oracle Traffic Director can generate internal requests, for example, an SHTML file or servlet might include a file. While processing the original request, Oracle Traffic Director makes an internal request to retrieve this file.

4.5.3 URI Translation

Oracle Traffic Director can execute `AuthTrans` and `NameTrans` directives to translate a URI to a physical path name without starting a new request.

4.6 Editing obj.conf

Use caution when making changes to this file. Simple mistakes can make Oracle Traffic Director fail to start or operate incorrectly.

4.6.1 Order of Directives

The order of directives is important, because Oracle Traffic Director executes them in the order in which they appear in `obj.conf`. The outcome of some directives affects the execution of other directives.

For `PathCheck` directives, the order within the `PathCheck` section is not important because Oracle Traffic Director executes all `PathCheck` directives. However, the order within the `ObjectType` section is very important, because if an `ObjectType` directive sets an attribute value, no other `ObjectType` directive can change that value.

Similarly, the order of directives in the `Service` section is very important. Oracle Traffic Director executes the first `Service` directive that matches the current request and does not execute the others.

4.6.2 Parameters

The number and names of parameters depend on the function. The order of parameters on the line is not important.

4.6.3 Case Sensitivity

Items in the `obj.conf` file are case-sensitive including function names, parameter names, parameter values, and path names.

4.6.4 Separators

Function names in the C language can be composed of letters, digits, and underscores. You can use the hyphen (-) in the configuration file in place of underscore (_) for your C code function names. This is only true for function names.

4.6.5 Quotation Marks

Quotation marks (") are only required around value strings when there is a space in the string. Otherwise, they are optional. Each open quotation mark must be matched by a closed quotation mark.

4.6.6 Spaces

- Spaces are not allowed at the beginning of a line except when continuing from the previous line.
- Spaces are not allowed before or after the equal sign (=) that separates the name and value.
- Spaces are not allowed at the end of a line or on a blank line.

4.6.7 Line Continuation

A long line may be continued on the next line by beginning the next line with a space or tab.

4.6.8 Path Names

Always use forward slashes (/), in path names. A backslash escapes the next character.

4.6.9 Comments

Comments begin with a pound sign (#). If you manually add comments to the `obj.conf` file, then use the Administration Console or CLI to make changes to your server: your comments are overwritten when the `obj.conf` file is updated.

5

Predefined Server Application Functions and Filters in obj.conf

This chapter describes the predefined server application functions (SAFs) and filters that are used in the `obj.conf` file. For details about the syntax and use of the `obj.conf` file, see Chapter 4, "Syntax and Use of obj.conf".

Each SAF has its own parameters that are passed to it by an `obj.conf` directive. SAFs can examine, modify, or create server variables. Each SAF returns a result code that indicates whether it has succeeded, did nothing, or has failed.

The SAFs in this chapter are grouped by the type of directive that calls them. For an alphabetical list of predefined SAFs and server configuration elements, see Appendix D, "Alphabetical List of Server Configuration Elements and Predefined SAFs".

This chapter includes the following topics:

5.1 The bucket Parameter

The `bucket` parameter is common to all SAFs. You can measure the performance of any SAF in the `obj.conf` file by adding a `bucket=`*bucket-name* parameter to the function, for example, `bucket="cache-bucket"`. The bucket statistics are displayed by the `perfdump` utility, which can be set up through the Administrator Console, CLI, or through the `service-dump` SAF.

The following performance buckets are predefined:

- The `default-bucket` records statistics for the functions not associated with any user-defined or built-in bucket.
- The `all-requests` bucket records `perfdump` statistics for all NSAPI SAFs, including those in the `default-bucket`.

5.2 AuthTrans

The `Authtrans` directive instructs Oracle Traffic Director to check for authorization before allowing a client to access resources. For more information, see Section 4.4.1, "AuthTrans".

The following `AuthTrans`-class functions are described in detail in this section:

- get-sslid
- qos-handler

In addition, the following common SAFs are valid for the `AuthTrans` directive:

- match-browser
- set-variable

5.2.1 get-sslid

The `get-sslid` function retrieves a string that is unique to the current SSL session and stores it as the `ssl-id` variable in the `Session->client` parameter block.

> **Note:** This function is provided for backward compatibility. The functionality of `get-sslid` was incorporated into the standard processing of an SSL connection.

If the variable `ssl-id` is present when a CGI is invoked, it is passed to the CGI as the `HTTPS_SESSIONID` environment variable. The `get-sslid` function has no parameters and always returns `REQ_NOACTION`. It has no effect if SSL is not enabled.

5.2.2 qos-handler

The `qos-handler` function examines the current quality of service (QoS) statistics for a virtual server, logs the statistics, and enforces the QoS parameters by returning an error. This function must be the first `AuthTrans` function configured in the `default` object.

Example

```
AuthTrans fn= "qos-handler"
```

> **See Also:** qos-error

5.2.3 webapp-firewall

The `webapp-firewall` function controls the enabling and disabling of the rule engine. If this function is present in a virtual server specific obj.conf, it indicates that the rule engine is enabled for that particular virtual server.

The webapp-firewall function is not configured by default and hence, the rule engine is not enabled. If the rule engine is not enabled, neither the directives nor the rules within the configuration files, specified by webapp-firewall-ruleset element, are applied.

Note:

- If the directive SecRuleEngine is specified within the configuration file(s) specified by the webapp-firewall-ruleset element, then it will be ignored. However, this condition is not applicable if SecRuleEngine is set to DetectionOnly mode.
- If there are other SAFs that could return REQ_PROCEED, then the SAF webapp-firewall must be on top of the list. If this is not the case, the execution of webapp-firewall might get skipped.
- For information about various web application firewall use cases, see the appendix, Web Application Firewall Examples and Use Cases in the *Oracle Traffic Director Administrator's Guide*.

Table 5–1 describes parameters for the webapp-firewall function. These parameters take precedence over the equivalent settings specified within the webapp-firewall-ruleset element.

Table 5–1 webapp-firewall Parameters

Parameter	Equivalent setting within webapp-firewall-ruleset	Description
detect-only	DetectionOnly	(optional) Indicates if the rule engine should enforce the rules or not. This is equivalent to setting SecRuleEngine directive to DetectionOnly. The value true indicates that the directives should be evaluated but the result of the evaluation should not be enforced. The value false indicates that the rules should be enforced. If this parameter is not specified and if SecRuleEngine is set to DetectionOnly mode (in the configuration file specified by webapp-firewall-ruleset), then the behavior would be same as setting detect-only to true.

Table 5–1 (Cont.) webapp-firewall Parameters

Parameter	Equivalent setting within webapp-firewall-ruleset	Description
process-request-body	SecRequestBodyAccess	(Optional) Indicates whether request bodies will be processed by web application firewall. When the body-buffer-size parameter in server.xml is configured to be a positive value, Oracle Traffic Director will buffer the request body in memory, up to the limit defined by body-buffer-size parameter. This parameter dictates whether web application firewall will access the buffered request body. The value on indicates that request bodies should be processed. The value off indicates that response bodies should not be processed. The default value is what is set by SecRequestBodyAccess directive (if any) in the configuration files specified by the webapp-firewall-ruleset element. If SecRequestBodyAccess directive is not present, the value is off.
process-response-body	SecResponseBodyAccess	(Optional) Indicates whether response bodies are buffered and processed by web application firewall. When response body processing is enabled, the server will buffer the entire response body in memory, up to the limit defined by the SecResponseBodyLimit directive (if any) in configuration files specified by the webapp-firewall-ruleset element. If SecResponseBodyLimit directive is not present, the value is 524288 (512 KB). The value on indicates that response bodies should be processed. The value off indicates that response bodies should not be processed. The default value is what is set by SecResponseBodyAccess directive (if any) in the configuration files specified by the webapp-firewall-ruleset directive. If SecResponseBodyAccess directive is not present, the value is off.

5.3 NameTrans

The NameTrans directive translates virtual URLs to physical directories on your server. The NameTrans directive must appear in the default object. For more information, see Section 4.4.2, "NameTrans".

The following NameTrans-class functions are described in detail in this section:

- assign-name
- map
- reverse-map
- rewrite
- strip-params
- sed-request-header

- sed-response-header
- sed-param-name
- sed-param-value

In addition, the following common SAFs are also valid for the `NameTrans` directive:

- match-browser
- redirect
- restart
- set-variable

5.3.1 assign-name

The `assign-name` function specifies the name of an object in the `obj.conf` file that matches the current request. Oracle Traffic Director then processes the directives in the named object in preference to the ones in the default object.

For example, if you have the following directive in the `default` object:

```
NameTrans fn="assign-name" name="personnel" from="/personnel"
```

Assume that Oracle Traffic Director receives a request for `http://`*server-name*`/personnel`. After processing this `NameTrans` directive, Oracle Traffic Director searches for an object named `personnel` in the `obj.conf` file and continues by processing the directives in the `personnel` object.

The `assign-name` function returns `REQ_NOACTION`.

Table 5–2 describes parameters for the `assign-name` function.

Table 5–2 assign-name Parameters

Parameter	Description
`from`	(Optional) Wildcard pattern that specifies the path to be affected. If you do not specify the `from` parameter, all paths are affected.
`name`	Specifies an additional named object in the `obj.conf` file whose directives are applied to this request.
`find-pathinfo-forward`	(Optional) Instructs Oracle Traffic Director to look for the `PATHINFO` forward in the path right after the `ntrans-base`, instead of backward from the end of path as Oracle Traffic Director function `assign-name` does by default. The `find-pathinfo-forward` parameter is ignored if the `ntrans-base` parameter is not set in `rq->vars`. By default, `ntrans-base` is set. This feature can improve performance for certain URLs by reducing the number of statistics performed.

Table 5–2 (Cont.) assign-name Parameters

Parameter	Description
nostat	(Optional) Prevents Oracle Traffic Director from performing a stat on a specified URL. The effect of nostat="*virtual-path*" in the NameTrans function assign-name is that Oracle Traffic Director assumes that a stat on the specified *virtual-path* will fail. Therefore, use nostat only when the path of the *virtual-path* does not exist on the system. For example, use nostat for NSAPI plug-in URLs to improve performance by avoiding unnecessary stats on those URLs. When the default PathCheck server functions are used, Oracle Traffic Director does not stat for the paths */ntrans-base/virtual-path* and */ntrans-base/virtual-path/** if *ntrans-base* is set (the default condition). It does not stat for the URLs */virtual-path* and */virtual-path/** if *ntrans-base* is not set.

Example

```
# This NameTrans directive is in the default object.
NameTrans fn="assign-name" name="proxy-cache" from="/.proxycache"
...
<Object name="proxy-cache">
...additional directives..
</Object>
```

5.3.2 map

The map function maps a request URI to a URL on another server, enabling you to specify that a request should be serviced by another server. To load balance a given URI across multiple servers, use the map function in conjunction with the set-origin-server function. The map function looks for a certain prefix in the URI that the client is requesting. If map finds the prefix, it replaces the prefix with the mirror site prefix.

Table 5–3 describes the parameters for the map function.

Table 5–3 map Parameters

Parameter	Description
from	The URI prefix to map. The prefix must not contain trailing slashes.
to	The URL prefix to which the request should be mapped. The prefix must not contain trailing slashes. The Host and Port values specified in this parameter are silently ignored.
name	(Optional) Specifies an additional named object in the obj.conf file. The directives of the named object will be applied to this request.
rewrite-host	(Optional) Indicates if the Host HTTP request header is rewritten to match the host specified by the to parameter. In a reverse proxy configuration where the proxy server and origin server service the same set of virtual servers, you can specify rewrite-host="false". Default value: true. It indicates that the Host HTTP request header is rewritten.

Example

```
NameTrans fn="map" from="/" name="reverse-proxy" to="/"
```

See Also: set-origin-server

5.3.3 reverse-map

The `reverse-map` function rewrites the HTTP response headers when Oracle Traffic Director is functioning as a reverse proxy. `reverse-map` looks for the URL prefix specified by the `from` parameter in certain response headers. If the `from` prefix matches the beginning of the response header value, `reverse-map` replaces the matching portion with the `to` prefix.

Table 5–4 describes the parameters for the `reverse-map` function.

Table 5–4 reverse-map Parameters

Parameter	Description
`from`	URL prefix to be rewritten.
`to`	URL prefix that will be substituted in place of the `from` prefix.
`rewrite-location`	(Optional) Indicates whether the `location` HTTP response header should be rewritten. Default value: `true`. It indicates that the location header is rewritten.
`rewrite-content-location`	(Optional) Indicates whether the `Content-Location` HTTP response header should be rewritten. Default value: `true`. It indicates that the `Content-Location` header is rewritten.
`rewrite-`*`headername`*	(Optional) Indicates whether the *headername* HTTP response header should be rewritten, where *headername* is a user-defined header name. With the exception of the `Location` and `Content-Location` headers. Default value: `false`. It indicates that the *headername* header is not rewritten.

Example

```
NameTrans fn="reverse-map" from="http://download.oracle.com/app/docs" to="/docs"
```

See Also: map

5.3.4 rewrite

The `rewrite` function allows flexible mappings between URIs and file system paths.

The following table describes parameters for the `rewrite` function.

Table 5–5 rewrite Parameters

Parameter	Description
`from`	(Optional) Wildcard pattern that specifies the path of requests that should be rewritten. The default is to match all paths.
`root`	(Optional) File system path to the effective root document directory.
`name`	(Optional) Name of an object in `obj.conf` whose directives will be applied to this request.
`path`	(Optional) Rewritten partial path. If non-empty, the path must begin with a slash (/).

Example

The following `obj.conf` code maps requests for the URI `/~user/index.html` to the file system path `/home/user/public_html/index.html`:

```
<If $path =~ "^/~([^/]+)(|/.*)$">
NameTrans fn="rewrite"
          root="/home/$1/public_html"
          path="$2"
</If>
```

See Also: restart

5.3.5 strip-params

The `strip-params` function removes the embedded semicolon-delimited parameters from the path. For example, a URI of `/dir1;param1/dir2` would become a path of `/dir1/dir2`. When used, the `strip-params` function should be the first `NameTrans` directive listed.

Example

```
NameTrans fn="strip-params"
```

5.3.6 sed-request-header

The `sed-request-header` function applies the `sed` edit commands to rewrite the value of a specified HTTP request header.

Table 5–6 describes the parameters for `sed-request-header`.

Table 5–6 sed-request-header Parameters

Parameter	Description
`name`	Specifies the HTTP request header.
`sed`	Specifies a `sed` command script. When multiple `sed` parameters are provided, the `sed` edit commands are evaluated in the order they appear.

Example

```
NameTrans fn="sed-request-header"
     name="x-someheader"
     sed="s/abcd/123/g"
```

See Also: insert-filter, sed-request, sed-response, sed-response-header, sed-param-name, sed-param-value

5.3.7 sed-response-header

The `sed-response-header` function applies the `sed` edit commands to rewrite the value of a specified HTTP response header.

Table 5–7 describes the parameters for `sed-response-header`.

Table 5–7 sed-response-header Parameters

Parameter	Description
name	Specifies the HTTP response header.
sed	Specifies a sed command script. When multiple sed parameters are provided, the sed edit commands are evaluated in the order they appear.

Example

```
NameTrans fn="sed-response-header"
     name="server"
     sed="s/backend/frontend/g"
```

See Also: insert-filter, sed-request, sed-response, sed-request-header, sed-param-name, sed-param-value

5.3.8 sed-param-name

The sed-param-name function applies the sed edit commands to rewrite an arbitrary pblock parameter name.

Table 5–8 describes the parameters for sed-param-name.

Table 5–8 sed-param-name Parameters

Parameter	Description
pblock	Specifies the pblock for the parameter.
name	Specifies the name of the parameter to rewrite.
sed	Specifies a sed command script. When multiple sed parameters are provided, the sed edit commands are evaluated in the order they appear.

Example

```
NameTrans fn="sed-param-name"
     pblock="headers"
     name="content-length"
     sed="s/-length/-Length/g"
```

See Also: insert-filter, sed-request, sed-response, sed-request-header, sed-response-header, sed-param-value

5.3.9 sed-param-value

The sed-param-value function applies the sed edit commands to rewrite an arbitrary pblock parameter value (corresponding to a specified name).

Table 5–9 describes the parameters for sed-param-value.

Table 5–9 sed-param-value Parameters

Parameter	Description
pblock	Specifies the pblock for the parameter.

Table 5–9 (Cont.) sed-param-value Parameters

Parameter	Description
name	Specifies the name of the param value to rewrite.
sed	Specifies a sed command script. When multiple sed parameters are provided, the sed edit commands are evaluated in the order they appear.

Example

```
NameTrans fn="sed-param-name"
     pblock="reqpb"
     name="uri"
     sed="s/test/plan/g"
```

See Also: insert-filter, sed-request, sed-response, sed-request-header, sed-response-header, sed-param-name

5.4 PathCheck

The PathCheck directive checks the URL that is returned after the NameTrans step to verify that the client is allowed to access the specified origin server. For more information, see Section 4.4.3, "PathCheck".

The following PathCheck-class functions are described in detail in this section:

- check-request-limits
- deny-existence
- get-client-cert
- nt-uri-clean
- ssl-logout
- unix-uri-clean

In addition, the following common SAFs are valid for the PathCheck directive:

- match-browser
- restart
- set-variable

5.4.1 check-request-limits

The check-request-limits function monitors incoming requests that match a given attribute (for example, client IP address) and computes an average requests per second on a configurable time interval. When requests that match the monitored attribute exceed a threshold that you configure, subsequent matching requests are not serviced until the request rate drops. Use this function to detect possible denial-of-service attacks.

You must specify either max-rps or max-connections, otherwise check-request-limits does nothing. If you do not enter an attribute or attributes to monitor, the function monitors all requests.

By default, the function keeps entries on requests for 300 seconds (5 minutes) before purging them. To adjust this time, use the `init-request-limits` SAF in the `magnus.conf` file.

Table 5–10 describes the parameters for the `check-request-limits` function.

Table 5–10 check-request-limits Parameters

Parameter	Description
`max-rps`	(Optional) Threshold for matching requests per second. If this threshold is exceeded subsequent connections matching the criteria are not serviced. Because an acceptable threshold value can vary widely between sites, there is no default value for this parameter.
`max-connections`	(Optional) Maximum number of concurrent matching connections. If Oracle Traffic Director receives a request that matches the criteria while the number of matching requests currently being processed meets or exceeds this number, the request is denied. Note that this number is the current requests at any time, and is independent of the `interval`. parameter. As soon as the number of concurrent requests falls below this limit, new matching requests are processed. Because an acceptable value can vary widely between sites, there is no default value for this parameter.
`interval`	(Optional) In seconds, the time interval during which average requests per second is computed. The `max-rps` limit is not applied until the next request rate computation. Because potential attackers can have unlimited requests serviced during this interval, balance the length of this interval against the performance cost of recomputing the maximum requests per second. Default value: 30 seconds.
`continue`	(Optional) Determines what condition must be met in order for a blocked request type to become available again for servicing. Valid values are: ▪ `silence` - Refused requests must fall to zero in a subsequent interval for service to resume. ▪ `threshold` - Refused requests must fall below the `max-rps` value for service to resume. Default value: `threshold`.
`error`	(Optional) The HTTP status code to use for blocked requests. Default value: `503` (the `Service Unavailable` error).
`monitor`	(Optional) A request attribute to monitor. Request rates are tracked in a bucket named by the value of this parameter. If the `monitor` parameter is not specified, the matching requests are tracked in an unnamed (anonymous) bucket. Note that these buckets are different from the buckets you specify with the standard `obj.conf bucket` parameter. Although the value of the `monitor` parameter can be a fixed string, it is most useful when you use predefined variables, for example, `monitor="$ip"`. You can also specify multiple variables, separated by a colon. For example, monitor="$ip:$uri". For a list of predefined variables, see "Predefined Variables".

Example

The following example limits a client IP to a maximum request rate of 10 requests per second in the default interval of 30 seconds:

```
PathCheck fn="check-request-limit" monitor="$ip" max-rps="10"
```

The following example limits a client IP to a maximum request rate of 10 requests per second when accessing any Perl CGIs. Other types of requests are unlimited:

```
<If path = "*.pl">
PathCheck fn="check-request-limits" monitor="$ip" max-rps="10"
</If>
```

For more information on using the `If` tag, see "If, ElseIf, and Else".

The following example limits requests globally for Perl CGIs to 10 requests per second. No specific `monitor` parameter is specified:

```
<If path = "*.pl">
PathCheck fn="check-request-limits" max-rps="10"
</If>
```

The following example limits a client IP from generating more than 10 Perl CGI requests per second, or 5 JSP requests per second. To track the Perl and JSP totals separately, the specified `monitor` parameters contain both a fixed string identifier and the client IP variable:

```
<If path = "*.pl">
PathCheck fn="check-request-limits" max-rps="10" monitor="perl:$ip"
</If>
<If path = "*.jsp">
PathCheck fn="check-request-limits" max-rps="5" monitor="jsp:$ip"
</If>
```

The following example limits any one client IP to no more than 5 connections at a given time:

```
PathCheck fn="check-request-limits" max-connections="2" monitor="$ip"
```

5.4.2 deny-existence

The `deny-existence` function sends a `404 Not Found` message when a client tries to access a specified path.

Table 5–11 describes parameters for the `deny-existence` function.

Table 5–11 deny-existence Parameters

Parameter	Description
`path`	(Optional) Wildcard pattern of the file system path to hide. If the path does not match, the function does nothing and returns `REQ_NOACTION`. If the path is not provided, it is assumed to match.
`bong-file`	(Optional) Specifies a file to send rather than responding with the `404 Not Found` message. The value is a full file system path.

Example

```
PathCheck fn="deny-existence" path="/opt/oracle/webserver7/docs/private"

PathCheck fn="deny-existence" bong-file="/svr/msg/go-away.html"
```

5.4.3 get-client-cert

The `get-client-cert` function gets the authenticated client certificate from the SSL3 session. It can apply to all HTTP methods, or only to those that match a specified pattern. It only works when SSL is enabled on Oracle Traffic Director.

If the certificate is present or obtained from the SSL3 session, the function returns REQ_NOACTION and allows the request to proceed. Otherwise, it returns REQ_ABORTED and sets the protocol status to 403 forbidden, causing the request to fail.

The following table describes parameters for the get-client-cert function.

Table 5–12 get-client-cert Parameters

Parameter	Description
dorequest	(Optional) Controls whether to actually get the certificate, or just test for its presence. ■ 1 tells the function to redo the SSL3 handshake to get a client certificate, if Oracle Traffic Director does not already have the client certificate. This typically causes the client to present a dialog box to the user to select a client certificate. Oracle Traffic Director might already have the client certificate if it was requested on the initial handshake, or if a cached SSL session has been resumed. ■ 0 tells the function not to redo the SSL3 handshake if Oracle Traffic Director does not already have the client certificate. If a certificate is obtained from the client and verified successfully by Oracle Traffic Director, the ASCII base 64 encoding of the DER-encoded X.509 certificate is placed in the parameter auth-cert in the Request->vars pblock, and the function returns REQ_PROCEED, allowing the request to proceed. Default value: 0.
require	(Optional) Controls whether failure to get a client certificate will abort the HTTP request. ■ 1 tells the function to abort the HTTP request if the client certificate is not present after dorequest is handled. In this case, the HTTP status is set to PROTOCOL_FORBIDDEN, and the function returns REQ_ABORTED. ■ 0 tells the function to return REQ_NOACTION if the client certificate is not present after dorequest is handled. Default value: 1.
method	(Optional) Specifies a wildcard pattern for the HTTP methods for which the function will be applied. If method is absent, the function is applied to all requests.

Example

```
# Get the client certificate from the session.
# If a certificate is not already associated with the session, request one.
# The request fails if the client does not present a
#valid certificate.
PathCheck fn="get-client-cert" dorequest="1"
```

5.4.4 nt-uri-clean

(Windows only) The nt-uri-clean function denies access to any resource whose physical path contains \.\, \..\ or \\ (these are potential security problems).

Table 5–13 describes the parameters for the nt-uri-clean function.

Table 5–13 nt-uri-clean Parameters

Parameter	Description
`tildeok`	(Optional) If present, allows tilde (~) characters in URIs. This is a potential security risk on the Windows platform, where `longfi~1.htm` might reference `longfilename.htm` but does not go through the proper ACL checking. If present, "`//`" sequences are allowed.
`dotdirok`	(Optional) If present, `/./` sequences are allowed.

Example

```
PathCheck fn="nt-uri-clean"
```

See Also: unix-uri-clean

5.4.5 ssl-logout

The `ssl-logout` function invalidates the current SSL session in Oracle Traffic Director's SSL session cache. This does not affect the current request, but the next time that the client connects, a new SSL session is created. If SSL is enabled, this function returns `REQ_PROCEED` after invalidating the session cache entry. If SSL is not enabled, it returns `REQ_NOACTION`.

5.4.6 unix-uri-clean

(UNIX only) The `unix-uri-clean` function denies access to any resource whose physical path contains `/./` or `/../` or `//` (these are potential security problems).

The following table describes parameters for the `unix-uri-clean` function.

Table 5–14 unix-uri-clean Parameters

Parameter	Description
`dotdirok`	If present, `/./` sequences are allowed.

Example

```
PathCheck fn="unix-uri-clean"
```

See Also: nt-uri-clean

5.5 ObjectType

The `ObjectType` directives determine the MIME type of the file that has to be sent to the client in response to a request. For more information, see ObjectType.

The following `ObjectType`-class functions are described in detail in this section:

- block-auth-cert
- block-cache-info
- block-cipher
- block-ip
- block-issuer-dn
- block-jroute

- block-keysize
- block-proxy-agent
- block-secret-keysize
- block-ssl
- block-ssl-id
- block-user-dn
- block-via
- block-xforwarded-for
- forward-auth-cert
- forward-cache-info
- forward-cipher
- forward-ip
- forward-issuer-dn
- forward-jroute
- forward-keysize
- forward-proxy-agent
- forward-secret-keysize
- forward-ssl
- forward-ssl-id
- forward-user-dn
- forward-via
- forward-xforwarded-for
- http-client-config
- proxy-cache-config
- proxy-cache-override-http
- proxy-websocket-config
- reverse-block-date
- reverse-block-server
- reverse-forward-date
- reverse-forward-server
- set-basic-auth
- set-cache-control
- set-cookie
- ssl-client-config
- type-by-exp
- type-by-extension

In addition, the following common SAFs are valid for the `ObjectType` directive:

- match-browser
- set-variable

5.5.1 block-auth-cert

The `block-auth-cert` function instructs Oracle Traffic Director to **not** generate and forward its own `Proxy-auth-cert` header to the origin server. In addition, if the incoming request contains this header, then the SAF will allow Oracle Traffic Director to pass-through the incoming request containing this header to the origin server.

Example

```
ObjectType fn="block-auth-cert"
```

See Also: forward-auth-cert

5.5.2 block-cache-info

The `block-cache-info` function instructs Oracle Traffic Director to **not** generate and forward its own `Proxy-cache-info` header to the origin server. In addition, if the incoming request contains this header, then the SAF will allow Oracle Traffic Director to pass-through the incoming request containing this header to the origin server.

Example

```
ObjectType fn="block-cache-info"
```

See Also: forward-cache-info

5.5.3 block-cipher

The `block-cipher` function instructs Oracle Traffic Director to **not** generate and forward its own `Proxy-cipher` header to the origin server. In addition, if the incoming request contains this header, then the SAF will allow Oracle Traffic Director to pass-through the incoming request containing this header to the origin server.

Example

```
ObjectType fn="block-cipher"
```

See Also: forward-cipher

5.5.4 block-ip

The `block-ip` function instructs Oracle Traffic Director to **not** generate and forward its own `Client-ip` header (or `Wl-proxy-client-ip` header for WebLogic Server) to the origin server. In addition, if the incoming request contains this header, then the SAF will allow Oracle Traffic Director to pass-through the incoming request containing this header to the origin server.

Example

```
ObjectType fn="block-ip"
```

See Also: forward-ip

5.5.5 block-issuer-dn

The `block-issuer-dn` function instructs Oracle Traffic Director to **not** generate and forward its own `Proxy-issuer-dn` header to the origin server. In addition, if the incoming request contains this header, then the SAF will allow Oracle Traffic Director to pass-through the incoming request containing this header to the origin server.

Example

```
ObjectType fn="block-issuer-dn"
```

See Also: forward-issuer-dn

5.5.6 block-jroute

The `block-jroute` function instructs Oracle Traffic Director to **not** generate and forward its own `Proxy-jroute` header to the origin server. In addition, if the incoming request contains this header, then the SAF will allow Oracle Traffic Director to pass-through the incoming request containing this header to the origin server.

Example

```
ObjectType fn="block-jroute"
```

See Also: forward-jroute

5.5.7 block-keysize

The `block-keysize` function instructs Oracle Traffic Director to **not** generate and forward its own `Proxy-keysize` header (or `Wl-proxy-client-keysize` header for WebLogic Server) to the origin server. In addition, if the incoming request contains this header, then the SAF will allow Oracle Traffic Director to pass-through the incoming request containing this header to the origin server.

Example

```
ObjectType fn="block-keysize"
```

See Also: forward-keysize

5.5.8 block-proxy-agent

The `block-proxy-agent` function instructs Oracle Traffic Director to **not** generate and forward its own `Proxy-agent` header to the origin server. In addition, if the incoming request contains this header, then the SAF will allow Oracle Traffic Director to pass-through the incoming request containing this header to the origin server.

Example

```
ObjectType fn="block-proxy-agent"
```

See Also: forward-proxy-agent

5.5.9 block-secret-keysize

The `block-secret-keysize` function instructs Oracle Traffic Director to **not** generate and forward its own `Proxy-secret-keysize` header (or `Wl-proxy-client-secretkeysize` header for WebLogic Server) to the origin server. In addition, if the incoming request contains this header, then the SAF will allow Oracle Traffic Director to pass-through the incoming request containing this header to the origin server.

Example

```
ObjectType fn="block-secret-keysize"
```

See Also: forward-secret-keysize

5.5.10 block-ssl

The `block-ssl` function instructs Oracle Traffic Director to **not** generate and forward its own `Proxy-ssl` header (or `Wl-proxy-ssl` header for WebLogic Server) to the origin server. In addition, if the incoming request contains this header, then the SAF will allow Oracle Traffic Director to pass-through the incoming request containing this header to the origin server.

Example

```
ObjectType fn="block-ssl"
```

See Also: forward-ssl

5.5.11 block-ssl-id

The `block-ssl-id` function instructs Oracle Traffic Director to **not** generate and forward its own `Proxy-ssl-id` header to the origin server. In addition, if the incoming request contains this header, then the SAF will allow Oracle Traffic Director to pass-through the incoming request containing this header to the origin server.

Example

```
ObjectType fn="block-ssl-id"
```

See Also: forward-ssl-id

5.5.12 block-user-dn

The `block-user-dn` function instructs Oracle Traffic Director to **not** generate and forward its own `Proxy-user-dn` header to the origin server. In addition, if the incoming request contains this header, then the SAF will allow Oracle Traffic Director to pass-through the incoming request containing this header to the origin server.

Example

```
ObjectType fn="block-user-dn"
```

See Also: forward-user-dn

5.5.13 block-via

The `block-via` function instructs Oracle Traffic Director to **not** generate and forward its own `Via` header to the origin server. In addition, if the incoming request contains this header, then the SAF will allow Oracle Traffic Director to pass-through the incoming request containing this header to the origin server.

Example

```
ObjectType fn="block-via"
```

See Also: forward-via

5.5.14 block-xforwarded-for

The `block-xforwarded-for` function instructs Oracle Traffic Director to **not** generate and forward its own `X-forwarded-for` header to the origin server. In addition, if the incoming request contains this header, then the SAF will allow Oracle Traffic Director to pass-through the incoming request containing this header to the origin server.

Example

```
ObjectType fn="block-xforwarded-for"
```

See Also: forward-xforwarded-for

5.5.15 forward-auth-cert

The `forward-auth-cert` function instructs Oracle Traffic Director to generate information about client's SSL/TLS certificate within the header `Proxy-auth-cert` and forward it to origin server. If an incoming request includes the header `Proxy-auth-cert`, this SAF will cause OTD to remove the header from the request that is forwarded to the origin server.

Table 5–15 describes the parameters for the `forward-auth-cert` function.

Table 5–15 forward-auth-cert Parameters

Parameter	Description
`hdr`	(Optional) Name of the HTTP request header used to communicate the client's DER-encoded SSL/TLS certificate in Base 64 encoding. Default value: `Proxy-auth-cert`.

See Also: block-auth-cert

5.5.16 forward-cache-info

The `forward-cache-info` function instructs Oracle Traffic Director to generate information about local hits within the header `Cache-info` and forward it to the origin server. If an incoming request includes the header `Cache-info`, this SAF will cause OTD to remove the header from the request that is forwarded to the origin server.

Table 5–16 describes the parameters for the `forward-cache-info` function.

Table 5–16 forward-cache-info Parameters

Parameter	Description
hdr	(Optional) Name of the HTTP request header used to communicate information about local cache hits. Default value: Cache-info.

See Also: block-cache-info

5.5.17 forward-cipher

The forward-cipher function instructs Oracle Traffic Director to generate information about the client's SSL/TLS cipher suite within the header Proxy-cipher and forward it to origin server. If an incoming request includes the header Proxy-cipher, this SAF will cause OTD to remove the header from the request that is forwarded to the origin server.

Table 5–17 describes the parameters for the forward-cipher function.

Table 5–17 forward-cipher Parameters

Parameter	Description
hdr	(Optional) Name of the HTTP request header used to communicate the name of the client's SSL/TLS cipher suite. Default value: Proxy-cipher.

See Also: block-cipher

5.5.18 forward-ip

The forward-ip function instructs Oracle Traffic Director to generate the client's IP address within the header Client-ip (or WI-proxy-client-ip for WebLogic Server) and forward it to origin server. If an incoming request includes the header Client-ip (or WI-proxy-client-ip for WebLogic Server), this SAF will cause Oracle Traffic Director to remove the header from the request that is forwarded to the origin server. Subsequently, Oracle Traffic Director generates and inserts this header with the appropriate value before forwarding the request to the origin server.

Table 5–18 describes parameters for the forward-ip function.

Table 5–18 forward-ip Parameters

Parameter	Description
hdr	(Optional) Name of the HTTP request header used to communicate the client's IP address. Default value: Client-ip, when the origin server is non-WLS and WI-proxy-client-ip -> when origin server is WLS.

See Also: block-ip

5.5.19 forward-issuer-dn

The forward-issuer-dn function instructs Oracle Traffic Director to generate information about the client's SSL/TLS certificate within the header Proxy-issuer-dn

and forward it to origin server. If an incoming request includes the header `Proxy-issuer-dn`, this SAF will cause OTD to remove the header from the request that is forwarded to the origin server.

Table 5–19 describes the parameters for the `forward-issuer-dn` function.

Table 5–19 forward-issuer-dn Parameters

Parameter	Description
`hdr`	(Optional) Name of the HTTP request header used to communicate the distinguished name of the issuer of the client's SSL/TLS certificate. Default value: `Proxy-issuer-dn`.

See Also: block-issuer-dn

5.5.20 forward-jroute

The `forward-jroute` function instructs Oracle Traffic Director to generate information about request routing within the header `Proxy-jroute` and forward it to origin server. The `Proxy-jroute` header field is used by the `set-origin-server function` and some Servlet containers to implement session stickiness. If an incoming request includes the header `Proxy-jroute`, this SAF will cause OTD to remove the header from the request that is forwarded to the origin server.

Table 5–20 describes the parameters for the `forward-jroute` function.

Table 5–20 forward-jroute Parameters

Parameter	Description
`hdr`	(Optional) Name of the HTTP request header used to communicate the request routing information. Default value: `Proxy-jroute`.

See Also: block-jroute, set-origin-server

5.5.21 forward-keysize

The `forward-keysize` function instructs Oracle Traffic Director to generate information about the size of the client's SSL/TLS key within the header `Proxy-keysize` and forward it to origin server. If an incoming request includes the header `Proxy-keysize`, this SAF will cause OTD to remove the header from the request that is forwarded to the origin server.

Table 5–21 describes the parameters for the `forward-keysize` function.

Table 5–21 forward-keysize Parameters

Parameter	Description
`hdr`	(Optional) Name of the HTTP request header used to communicate the size of the client's SSL/TLS key. Default value: `Proxy-keysize`.

See Also: block-keysize

5.5.22 forward-proxy-agent

The `forward-proxy-agent` function instructs Oracle Traffic Director to generate its version information within the header `Proxy-agent` and forward it to origin server. If an incoming request includes the header `Proxy-agent`, this SAF will cause OTD to remove the header from the request that is forwarded to the origin server.

Table 5–22 describes the parameters for the `forward-proxy-agent` function.

Table 5–22 forward-proxy-agent Parameters

Parameter	Description
`hdr`	(Optional) Name of the HTTP request header used to communicate server version. Default value: `Proxy-agent`.

See Also: block-proxy-agent, http-client-config

5.5.23 forward-secret-keysize

The `forward-secret-keysize` function instructs Oracle Traffic Director to generate information about the size of the client's SSL/TLS secret key within the header `Proxy-secret-keysize` (or `Wl-proxy-client-secretkeysize` for WebLogic Server) and forward it to origin server. If an incoming request includes the header `Proxy-secret-keysize` (or `Wl-proxy-client-secretkeysize` for WebLogic Server), this SAF will cause OTD to remove the header from the request that is forwarded to the origin server.

Table 5–23 forward-secret-keysize Parameters

Parameter	Description
`hdr`	(Optional) Name of the HTTP request header used to communicate the client's SSL/TLS secret key. Default value: `Proxy-secret-keysize`, when the origin server is non-WLS and `Wl-proxy-client-secretkeysize` -> when origin server is WLS.

Example

```
ObjectType fn="forward-secret-keysize"
```

See Also: block-secret-keysize

5.5.24 forward-ssl

The `forward-ssl` function instructs the server to forward information to remote (origin) servers to check if the client sent the request to Oracle Traffic Director over an SSL connection. Accordingly, if the client connects to OTD using a non-SSL connection, this header is set with the value False. Similarly, if the client connects to OTD using an SSL connection, then this header is set with the value True. If an incoming request includes the header `Proxy-ssl` (or `WI-proxy-ssl` for WebLogic Server), this SAF will cause OTD to remove the header from the request that is forwarded to the origin server.

Table 5–24 forward-ssl

Parameter	Description
hdr	Name of the HTTP request header used to communicate that the connection between the client and OTD was over SSL. Default value: `Proxy-ssl`, when the origin server is non-WLS and `WI-proxy-ssl` -> when origin server is WLS.

Example

```
ObjectType fn="forward-ssl"
```

See Also: block-ssl

5.5.25 forward-ssl-id

The `forward-ssl-id` function instructs Oracle Traffic Director to generate information about the client's SSL/TLS session ID within the header `Proxy-ssl-id` and forward it to origin server. If an incoming request includes the header `Proxy-ssl-id`, this SAF will cause OTD to remove the header from the request that is forwarded to the origin server.

Table 5–25 describes the parameters for the `forward-ssl-id` function.

Table 5–25 forward-ssl-id Parameters

Parameter	Description
`hdr`	(Optional) Name of the HTTP request header used to communicate the client's SSL/TLS session ID. Default value: `Proxy-ssl-id`.

See Also: block-ssl-id

5.5.26 forward-user-dn

The `forward-user-dn` function instructs Oracle Traffic Director to generate information about the distinguished name of the subject of the client's SSL/TLS certificate within the header `Proxy-user-dn` and forward it to origin server. If an incoming request includes the header `Proxy-user-dn`, this SAF will cause OTD to remove the header from the request that is forwarded to the origin server.

Table 5–26 describes the parameters for the `forward-user-dn` function.

Table 5–26 forward-user-dn Parameters

Parameter	Description
`hdr`	(Optional) Name of the HTTP request header used to communicate the distinguished name of the subject of the client's SSL/TLS certificate. Default value: `Proxy-user-dn`.

See Also: block-user-dn

5.5.27 forward-via

The `forward-via` function instructs Oracle Traffic Director to generate information about request routing within the header `Via` and forward it to origin server using the HTTP/1.1 `Via` format. The HTTP/1.1 `Via` header field records the proxy servers and protocol versions that were involved in routing a request. If an incoming request includes the header `Via`, this SAF will cause OTD to remove the header from the request that is forwarded to the origin server.

Table 5–27 describes parameters for the `forward-via` function.

Table 5–27 forward-via Parameters

Parameter	Description
`hdr`	(Optional) Name of the HTTP request header used to communicate routing information. Default value: `Via`.

See Also: block-via

5.5.28 forward-xforwarded-for

The `forward-xforwarded-for` function instructs Oracle Traffic Director to generate information about user-specified `X-Forwarded-For` header values within the header `X-Forwarded-For` and forward it to origin server. If the function is enabled, Oracle Traffic Director sends the `X-Forwarded-For` header value to the origin server, where the value is a comma-separated list of IP addresses. This SAF is enabled by default. If an incoming request includes the header `X-forwarded-for`, this SAF will cause OTD to remove the header from the request that is forwarded to the origin server.

Table 5–28 forward-xforwarded-for

Parameter	Description
hdr	(Optional) Name of the HTTP request header used to communicate routing information. Default value: `X-forwarded-for`.

Example

```
ObjectType fn="forward-xforwarded-for"
```

See Also: block-xforwarded-for

5.5.29 http-client-config

The `http-client-config` function configures Oracle Traffic Director's HTTP client.

Table 5–29 describes the parameters for the `http-client-config` function.

Table 5–29 http-client-config Parameters

Parameter	Description
`always-use-keep-alive`	(Optional) Indicates if the HTTP client can reuse existing persistent connections for all types of requests. Default value: `true` for WLS origin servers. It indicates that `GET/HEAD/OPTIONS` requests uses keep-alive by default.

Table 5–29 (Cont.) http-client-config Parameters

Parameter	Description
exclude-escape-chars	(Optional) Specifies the list of characters that Oracle Traffic Director should not escape. Various applications deployed in the application server requires certain characters not to be escaped. If you do not specify this parameter, Oracle Traffic Director may escape those characters. For example: `ObjectType fn="http-client-config" exclude-escape-chars="%&"`
keep-alive	(Optional) Indicates if the HTTP client uses persistent connections. Default value: `true`.
keep-alive-timeout	(Optional) The maximum number of seconds to keep a persistent connection open. Default value: `29`.
log-headers	(Optional) Specifies whether to log request or response headers in server log that Oracle Traffic Director sends and receives from the origin server. This parameter is useful for diagnostics purposes. For example: `ObjectType fn="http-client-config" log-headers="true"`
protocol	(Optional) HTTP protocol version string. By default, the HTTP client uses either `HTTP/1.0` or `HTTP/1.1` based on the contents of the HTTP request. In general, do not use the protocol parameter unless you encounter specific protocol interoperability problems.
proxy-agent	(Optional) Value of the proxy-agent HTTP request header. The default is a string that contains the web server product name and version.
proxy-buffer-size	(Optional) Specifies the size of the buffer that is used by Oracle Traffic Director to store data before it is sent to the client. Higher the value of this buffer size, the lower the number of write system calls. By default, the value of proxy buffer size is 16 KB. To change the value to 32 KB, use the parameter as follows: `ObjectType fn="http-client-config" proxy-buffer-size="32768"`
retries	(Optional) The number of times to retry getting content from the origin web server before sending an error to the client. Only `GET/HEAD/OPTIONS` requests are retried. If `GET` requests has body associated, then those requests are also not retried. Acceptable values are `0` through `100`, with the value `0` indicating that no retries should be attempted. Default value: `3`.
wls-initiated-failover	Controls whether or not a 503 error response from Oracle WebLogic Server triggers a failover to another server. Default value: `true`.

Example

```
ObjectType fn="http-client-config" keep-alive="false"
```

5.5.30 proxy-cache-config

The `proxy-cache-config` function configures reverse proxy cache settings.

Table 5–30 describes the parameters for the `proxy-cache-config` function.

Table 5–30 proxy-cache-config Parameters

Parameter	Type	Description	Default Value
enabled	Boolean	Enable or disable caching of reverse proxy content.	false
max-reload-interval	Integer	(Optional) Specifies the maximum time in seconds allowed between consecutive up-to-date checks. If set to 0 default value, a check is made every time the document is accessed, and the last-modified-factor value has no effect. **Note**: Setting a value for this element will cause the server to behave in a manner different from what the HTTP specification mandates.	3600
min-reload-interval	Integer	Specifies the minimum time in seconds allowed between consecutive up-to-date checks of a cached document.	0
last-modified-factor	Float	(Optional) Represents the factor used in estimating the expiry time, which defines how long a document will be up-to- date based on the time it was last modified. The time elapsed since the last modification is multiplied by this factor. The result gives the estimated time the document is likely to remain unchanged. Specifying a value of 0 turns off this function. The caching system then uses only explicit expiry information which is rarely available. Only explicit Cache-Control or Expires HTTP headers are used. This value has no effect if max-reload-interval is set to 0.	0
min-object-size	Integer	The minimum size, in bytes, of any document to be cached. You can prefer to cache only larger documents.	0
max-object-size	Integer	The maximum size, in bytes, of any document to be cached. This setting enables users to limit the maximum size of cached documents, so that no single document can take too much space. This value cannot exceed the maximum value specified in server.xml in proxy-cache->max-heap-object-size.	proxy-cache->max-heap-object-size
query-maxlen	Integer	Specifies the number of characters in the query string (the "?string" part at the end of the URL that are still cacheable). The same queries are rarely repeated exactly in the same form by more than one user, and so caching them is often not desirable.	0
compression	Boolean	If this parameter is set to true then the proxy compresses the document before storing. This consumes less cache space, therefore cache can accommodate more entries for a given cache size. For clients which accept compression, compressed content is sent. For other clients, server will dynamically uncompress the content. Cache will not store two copies of the same content.	false
cache-https	Boolean	If this parameter is set to true then responses from HTTPS connections are also cached. If cache-https is false then responses from HTTPS origin server connections are not cached irrespective of HTTP headers.	false

Example

```
ObjectType fn="proxy-cache-config" enable="1" max-reload-interval=300
min-reload-interval=60
```

See Also: proxy-cache-override-http, service-proxy-cache-dump

5.5.31 proxy-cache-override-http

The `proxy-cache-override-http` function configures reverse proxy cache parameters that override certain HTTP caching rules.

Table 5–31 describes the parameters for the `proxy-cache-override-http` function.

Table 5–31 proxy-cache-override-http Parameters

Parameter	Type	Description	Default Value
`ignore-server-no-cache`	Boolean	If this parameter is set to `true`, a `pragma: no-cache` or `cache-control: no-cache` header from the origin server is ignored and the response is cached. This behavior violates the HTTP standard.	`false`
`ignore-server-no-store`	Boolean	If this parameter is set to `true`, a `cache-control: no-store` header from the origin server is ignored and the response is cached. This behavior violates the HTTP standard.	`false`
`ignore-private`	Boolean	If this parameter is set to `true`, a `cache-control: private` header from the origin server is ignored and the response is cached. This behavior violates the HTTP standard.	`false`
`ignore-client-no-cache`	Boolean	If set to `true`, a `pragma: no-cache` or `cache-control: no-cache` header from the client is ignored and the request is served from the cache. This behavior violates the HTTP standard.	`false`
`override-expire`	Boolean	If this parameter is set to `true`, `min-reload-interval` is enforced over the value of an `Expires` header and `Cache-Control: max-age` value. This behavior violates the HTTP standard.	`false`
`override-lastmod`	Boolean	If this parameter is set to `true`, `min-reload-interval` is enforced over the value of a `Last-modified` header. This behavior violates the HTTP standard.	`false`
`reload-into-ims`	Boolean	If this parameter is set to `true`, reload request from clients are converted into conditional GET requests with an `If-modified-since` header. This behavior violates the HTTP standard.	`true`
`require-expires`	Boolean	If this parameter is set to `true`, a response without an Expires header will not be cached. This behavior violates the HTTP standard.	`false`
`without-lastmod`	Boolean	If this parameter is set to `true`, the absence of a `Last-modified` header is ignored and the response cached. This behavior violates the HTTP standard.	`false`

Example

```
<If uri =~ '^/images/'>
    ObjectType fn="proxy-cache-config" enable="1" max-reload-interval=600
    ObjectType fn="proxy-cache-override-http" ignore-client-no-cache="true"
</If>
<Else uri =~ '^/myapp/'>
    ObjectType fn="proxy-cache-config" enable="1" max-reload-interval=120
</Else>
```

See Also: proxy-cache-config, service-proxy-cache-dump

5.5.32 proxy-websocket-config

The `proxy-websocket-config` SAF disables WebSocket upgrade and modifies the idle-timeout for WebSocket connections. WebSocket upgrade is enabled by default. If WebSocket upgrade needs to be disabled, then proxy-websocket-config can be used with enabled=off. The proxy-websocket-config directive may be present in the route object for a route or the default object for the whole virtual server. This enables administrators to disable WebSocket traffic or set a different idle-timeout value for certain routes or for the whole virtual server.

Table 5–32 describes the parameters for the `proxy-cache-override-http` function.

Table 5–32 proxy-websocket-config Parameters

Parameter	Default Value
`enabled`	'on' or 'off'
`idle-timeout`	Default is the timeout value mentioned in the tcp-thread-pool element

Example

```
ObjectType fn="proxy-websocket-config"
```

5.5.33 reverse-block-date

The `reverse-block-date` SAF blocks the Date header sent from the origin server and causes Oracle Traffic Director to generate and insert its own Date header in the response.

Example

```
ObjectType fn="reverse-block-date"
```

See Also: reverse-forward-date

5.5.34 reverse-block-server

The `reverse-block-server` SAF blocks the Server header sent from the origin server and causes Oracle Traffic Director to insert its own Server header in the response.

Example

```
ObjectType fn="reverse-block-server"
```

See Also: reverse-forward-server

5.5.35 reverse-forward-date

The `reverse-forward-date` SAF forwards the Date header sent from the origin server. In Oracle Traffic Director, this is the default behavior.

Example

```
ObjectType fn="reverse-forward-date"
```

See Also: reverse-block-date

5.5.36 reverse-forward-server

The `reverse-forward-server` SAF forwards the Server header sent from the origin server. If origin server does not generate any Server header then Oracle Traffic Director generates and uses its own Server header. This is the default behavior.

Example

```
ObjectType fn="reverse-forward-server"
```

See Also: reverse-block-server

5.5.37 set-basic-auth

The `set-basic-auth` function allows you to set the HTTP basic authentication credentials, used by the server, when it sends an HTTP request. Use `set-basic-auth` to authenticate to a remote origin server or proxy server.

The following table describes parameters for the `set-basic-auth` function.

Table 5–33 set-basic-auth Parameters

Parameter	Description
`user`	Name of the user to authenticate.
`password`	Password of the user to authenticate.
`hdr`	(Optional) Name of the HTTP request header used to communicate the credentials.
`bucket`	(Optional) Common to all `obj.conf` functions. Adds a bucket to monitor performance. For more information, see The bucket Parameter.

Example

```
ObjectType fn="set-basic-auth" user="admin" password="secret"
hdr="proxy-authorization"
```

5.5.38 set-cache-control

The `set-cache-control` function allows you to specify the HTTP caching policy for the response being sent back to the client.

The following table describes parameters for the `set-cache-control` function.

Table 5–34 set-cache-control Parameters

Parameter	Description
`control`	HTTP cache control directives. Separate multiple directives by commas.

The following table describes some of the useful cache control directives defined by the HTTP/1.1 protocol.

Table 5–35 Cache Control Directives

Directive	Description
`public`	The response may be cached by any cache.
`private`	The response must not be cached by a shared cache (for example, a proxy server).
`no-cache`	Clients must ask Oracle Traffic Director for updated content on each access.
`max-age=`*n*	The response should not be cached for more than *n* seconds.

Example

```
ObjectType fn="set-cache-control" control="private,max-age=60"
```

5.5.39 set-cookie

The `set-cookie` function allows you to set a cookie in the response being sent back to the client.

The following table describes parameters for the `set-cookie` function.

Table 5–36 set-cookie Parameters

Parameter	Description
`name`	Name of the cookie.
`value`	(Optional) Value of the cookie. Default value: `null`.
`path`	(Optional) Base URI to which the cookie applies. Default value: `/` (slash).
`domain`	(Optional) The domain name of servers to which the cookie must be sent. If no domain is specified, web browsers send the cookie only to Oracle Traffic Director that sets the cookie.
`max-age`	(Optional) Maximum time (in seconds) after which the cookie expires. If `max-age` is not specified, web browsers delete the cookie when the user closes the web browser.

Example

```
<If not defined $cookie{'FIRSTVISITTIME'}>
ObjectType fn="set-cookie"
           name="FIRSTVISITTIME"
           value="$time"
           max-age="31536000"
</If>
```

5.5.40 ssl-client-config

The `ssl-client-config` function configures options used when Oracle Traffic Director connects to a origin server using SSL/TLS.

Table 5–37 describes the parameters for the `ssl-client-config` function.

Table 5–37 ssl-client-config Parameters

Parameter	Description
client-cert-nickname	(Optional) Nickname of the client certificate to present to the remote server. The default is not to present a client certificate.
validate-server-cert	(Optional) Boolean that indicates whether Oracle Traffic Director validates the certificate presented by the remote server. Default value: true. It indicates that remote servers must present valid certificates that were issued by a trusted certificate authority.

Example

```
ObjectType fn="ssl-client-config" validate-server-cert="false"
```

See Also: block-auth-cert, forward-auth-cert

5.5.41 type-by-exp

The type-by-exp function matches the current path with a wildcard expression. If they match, the type parameter information is applied to the file. This is the same as type-by-extension, except that you use wildcard patterns for the files or directories specified in the URLs.

Table 5–38 describes the parameters for the type-by-exp function.

Table 5–38 type-by-exp Parameters

Parameter	Description
exp	Wildcard pattern of paths for which this function is applied.
type	(Optional) Type assigned to a matching request (the Content-Type header).
enc	(Optional) Encoding assigned to a matching request (the Content-Encoding header).
lang	(Optional) Language assigned to a matching request (the Content-Language header).
charset	(Optional) The character set for the magnus-charset parameter in rq->srvhdrs. If a browser sends the Accept-Charset header or its User-Agent is Mozilla/1.1 or newer, then append "; charset=*charset*" to Content-Type, where *charset* is the value of the magnus-charset parameter in rq->srvhdrs.

Example

```
ObjectType fn="type-by-exp" exp="*.test" type="application/html"
```

See Also: type-by-extension

5.5.42 type-by-extension

The type-by-extension function instructs Oracle Traffic Director to look in a table of MIME type mappings to find the MIME type of the requested resource. The MIME type is added to the Content-Type header that is sent back to the client.

The table of MIME type mappings is created by a mime-file element in the server.xml file, which loads a MIME types file or list and creates the mappings.

For example, the following two lines are part of a MIME types file:

```
type=text/html exts=htm,html
type=text/plain exts=txt
```

If the extension of the requested resource is `htm` or `html`, the `type-by-extension` file sets the type to `text/html`. If the extension is `.txt`, the function sets the type to `text/plain`.

Example

```
ObjectType fn="type-by-extension"
```

See Also: type-by-exp

5.6 Input

The `Input` directives allow you to select filters that will process incoming request data read by the `Service` stage. For more information, see Input.

The following `Input`-class filter is described in detail in this section:

- sed-request
-

The following common SAFs are valid for the `Input` directive:

- insert-filter
- match-browser
- remove-filter
- set-variable

Every `Input` directive has the following optional parameters.

Table 5–39 Input Directive's Optional Parameters

Parameters	Description
`type`	(Optional) Specifies a wildcard pattern of MIME types for which this function is executed.
`method`	(Optional) Specifies a wildcard pattern of HTTP methods for which this function is executed. Common HTTP methods are `GET`, `HEAD`, and `POST`.
`query`	(Optional) Specifies a wildcard pattern of query strings for which this function is executed.

5.6.1 sed-request

The `sed-request` filter applies the `sed` edit commands to an incoming request entity body, for example, an uploaded file or submitted form.

Table 5–40 describes the parameters for `sed-request`.

Table 5–40 sed-request Parameters

Parameter	Description
`sed`	Specifies a `sed` command script. When multiple `sed` parameters are provided, the `sed` edit commands are evaluated in the order they appear.

Example

The following `obj.conf` code instructs `sed-request` to encode any (<) and (>) characters posted in an HTML form:

```
Input fn="insert-filter"
      method="POST"
      filter="sed-request"
      sed="s/</\\&lt;/g"
      sed="s/%3c/\\&lt;/g"
      sed="s/%3C/\\&lt;/g"
      sed="s/>/\\&gt;/g"
      sed="s/%3e/\\&gt;/g"
      sed="s/%3E/\\&gt;/g"
```

Because `POST` bodies are usually URL-encoded, it is important to check for URL-encoded forms when editing `POST` bodies. `%3C` is the URL-encoded form of (<) and `%3E` is the URI-encoded form of (>).

See Also: insert-filter, sed-response, sed-request-header, sed-response-header, sed-param-name, sed-param-value

5.7 Output

The `Output` stage allows you to select filters that will process outgoing data. For more information, see Output.

Every `Output` directive has the following optional parameters:

Table 5–41 Output Directive's Optional Parameters

Parameters	Description
`type`	(Optional) Specifies a wildcard pattern of MIME types for which this function is executed.
`method`	(Optional) Specifies a wildcard pattern of HTTP methods for which this function is executed. Common HTTP methods are `GET`, `HEAD`, and `POST`.
`query`	(Optional) Specifies a wildcard pattern of query strings for which this function is executed.

The following `Output`-class filters are described in detail in this section:

- sed-responsesed-request

The following common SAFs are valid for the Output directive:

- insert-filter
- match-browser
- redirect
- remove-filter
- restart
- set-variable

5.7.1 sed-response

The `sed-response` filter applies the `sed` edit commands to an incoming request entity body, for example, an uploaded file or submitted form.

Table 5–42 describes the parameters for `sed-response`.

Table 5–42 sed-response Parameters

Parameter	Description
`sed`	Specifies a `sed` command script. When multiple `sed` parameters are provided, the `sed` edit commands are evaluated in the order they appear.

Example

The following `obj.conf` code instructs `sed-response` to rewrite any occurrence of `http://127.0.0.1/` in an HTML response to `http://server.example.com/`:

```
Output fn="insert-filter"
       type="text/html"
       filter="sed-response"
       sed="s|http://127.0.0.1/|http://server.example.com/|g"
```

See Also: insert-filter, sed-request, sed-request-header, sed-response-header, sed-param-name, sed-param-value

5.8 Route

The `Route` directive specifies information as to where the Web Server should route requests. For more information, see Route.

The following `Route`-class functions are described in detail in this section:

- set-origin-server

In addition, the following common SAFs are valid for the Route directive:

- match-browser
- set-variable

5.8.1 set-origin-server

The `set-origin-server` function distributes the load across a set of homogeneous HTTP origin servers. This SAF chooses the origin server from a given origin server pool for this request. The `set-origin-server` SAF requires `origin-server-pool` as mandatory parameter.

Table 5–43 describes the parameters for the `set-origin-server` function.

Table 5–43 set-origin-server Parameters

Parameter	Description
`origin-server-pool`	(Mandatory) Name of the configured origin server pool. From this pool, one of the origin servers will be chosen based on the load balancing properties defined within the `origin-server-pool` element in server.xml.

Table 5–43 (Cont.) set-origin-server Parameters

Parameter	Description
`sticky-cookie`	(Optional) Name of a cookie that, when present in a response, will cause subsequent requests to stick to that origin server. Accordingly, subsequent requests with this cookie are sent to the same origin server. This parameter accepts * as value, which means that any cookie received from the origin server will be considered as sticky. Default value: `*`.
`sticky-param`	(Optional) Name of a URI parameter to inspect for route information. When the URI parameter is present in a request URI and its value contains a colon (:) followed by a route ID, the request will stick to the origin server identified by that route ID. Default value: `jsessionid`.
`route-hdr`	(Optional) Name of the HTTP request header used to communicate route IDs to origin servers. `set-origin-server` associates each origin server named by a server parameter with a unique route ID. Origin servers may encode this route ID in the URI parameter named by the `sticky-param` parameter to cause subsequent requests to stick to them. Default value: `Proxy-jroute`.
`route-cookie`	(Optional) Name of the cookie generated by Oracle Traffic Director when it encounters a `sticky-cookie` in a response. The `route-cookie` parameter stores a route ID that enables Oracle Traffic Director to direct subsequent requests back to the same origin server. Default value: `JROUTE`.
`rewrite-host`	(Optional) Indicates whether the host HTTP request header is rewritten to match the host specified by the `server` parameter. Default value: `false`. It indicates that the host header is not rewritten.
`rewrite-location`	(Optional) Indicates whether the `Location` HTTP response header that matches the `server` parameter should be rewritten. Default value: `true`. It indicates that the matching `Location` headers are rewritten.
`rewrite-content-location`	(Optional) Indicates whether the `Content-Location` HTTP response header that matches Oracle Traffic Director parameter should be rewritten. Default value: `true`. It indicates that the matching `Content-Location` headers are rewritten.
`rewrite-`*`headername`*	(Optional) Indicates whether the *headername* HTTP response headers that match Oracle Traffic Director parameter should be rewritten, where *headername* is a user-defined header name. *headername* is in lowercase. With the exception of the `Location` and `Content-Location` headers, Default value: `false`. It indicates that the `headername` header is not rewritten.

Example

```
Route fn="set-origin-server" origin-server-pool="origin-server-pool-1"
```

See Also: map

5.9 Service

The Service directives send the response data to the client. For more information, see Service.

Every Service directive has the following optional parameters to determine whether the function is executed. All optional parameters must match the current request for the function to be executed.

Table 5–44 Service Directive's Optional Parameters

Optional Parameters	Description
type	Specifies a wildcard pattern of MIME types for which this function is executed. The magnus-internal/* MIME types are used only to select a Service function to execute.
method	Specifies a wildcard pattern of HTTP methods for which this function is executed. Common HTTP methods are GET, HEAD, and POST.
query	Specifies a wildcard pattern of query strings for which this function is executed.
UseOutputStreamSize	Determines the default output stream buffer size (in bytes), for data sent to the client. If this parameter is not specified, the default is 8192 bytes. **Note:** Set this parameter to zero (0) to disable output stream buffering.
flushTimer	Determines the maximum number of milliseconds between write operations in which buffering is enabled. If the interval between subsequent write operations is greater than the flushTimer value for an application, further buffering is disabled. This is necessary for monitoring the status of CGI applications that run continuously and generate periodic status update reports. If this parameter is not specified, the default is 3000 milliseconds.
ChunkedRequestBufferSize	Determines the default buffer size, in bytes, for unchunking request data. If this parameter is not specified, the default is 8192 bytes.
ChunkedRequestTimeout	Determines the default timeout, in seconds, for unchunking request data. If this parameter is not specified, the default is 60 seconds.

If there is more than one Service-class function, the first one matching the optional wildcard parameters (type, method, and query) are executed.

The UseOutputStreamSize, ChunkedRequestBufferSize, and ChunkedRequestTimeout parameters also have equivalent magnus.conf directives. The obj.conf parameters override the magnus.conf directives.

By default, Oracle Traffic Director sends the requested file to the client by calling the send-file function. The directive that sets the default is:

```
Service method="(GET|HEAD)" type="*~magnus-internal/*" fn="send-file"
```

This directive usually comes last in the set of Service-class directives to give all other Service directives a chance to be invoked. This directive is invoked if the method of the request is GET, HEAD, or POST, and the type does not start with magnus-internal/. Note here that the pattern *~ means "does not match." For a list of characters that can be used in patterns, see Wildcard Patterns.

The functions used in the `Service` directive are described in the following sections:

- proxy-retrieve
- remove-filter
- service-proxy-cache-dump
- service-trace
- stats-xml

In addition, the following common SAFs are valid for the Service directive:

- match-browser
- remove-filter
- set-variable

5.9.1 proxy-retrieve

The `proxy-retrieve` function retrieves a document from a remote server and returns it to the client. This function also enables you to configure Oracle Traffic Director to allow or block arbitrary methods. This function only works on the HTTP protocol.

Table 5–45 describes the parameters for the `proxy-retrieve` function.

Table 5–45 proxy-retrieve Parameters

Parameter	Description
`type`	(Optional) Common to all `Service`-class functions. Specifies a wildcard pattern of MIME types for which this function is executed. For more information, see Service.
`method`	(Optional) Common to all `Service`-class functions. Specifies a wildcard pattern of HTTP methods for which this function is executed. For more information, see Service.
`query`	(Optional) Common to all `Service`-class functions. Specifies a wildcard pattern of query strings for which this function is executed. For more information, see Service.
`UseOutputStreamSize`	(Optional) Common to all `Service`-class functions. Determines the default output stream buffer size (in bytes), for data sent to the client. For more information, see Service.
`flushTimer`	(Optional) Common to all `Service`-class functions. Determines the maximum number of milliseconds between write operations in which buffering is enabled. For more information, see Service.
`ChunkedRequestBufferSize`	(Optional) Common to all `Service`-class functions. Determines the default buffer size, in bytes, for unchunking request data. For more information, see Service.
`ChunkedRequestTimeout`	(Optional) Common to all `Service`-class functions. Determines the default timeout, in seconds, for unchunking request data. For more information, see Service.

Example

```
# Normal proxy retrieve
Service fn="proxy-retrieve"
# Proxy retrieve with POST method disabled
Service fn="proxy-retrieve"
     method="(POST)"
```

See Also: set-origin-server

5.9.2 remove-filter

The `remove-filter` function is used to remove a filter from the filter stack. If the filter is inserted multiple times, only the topmost instance is removed. In general, it is not necessary to remove filters with remove-filter, as they are removed automatically at the end of a request.

The following table describes parameters for the `remove-filter` function.

Table 5–46 remove-filter Parameters

Parameter	Description
`type`	(Optional) Common to all `Service`-class functions. Specifies a wildcard pattern of MIME types for which this function is executed. For more information, see Service.
`method`	(Optional) Common to all `Service`-class functions. Specifies a wildcard pattern of HTTP methods for which this function is executed. For more information, see Service.
`query`	(Optional) Common to all `Service`-class functions. Specifies a wildcard pattern of query strings for which this function is executed. For more information, see Service.
`UseOutputStreamSize`	(Optional) Common to all `Service`-class functions. Determines the default output stream buffer size (in bytes), for data sent to the client. For more information, see Service.
`flushTimer`	(Optional) Common to all `Service`-class functions. Determines the maximum number of milliseconds between write operations in which buffering is enabled. For more information, see Service.
`ChunkedRequestBufferSize`	(Optional) Common to all `Service`-class functions. Determines the default buffer size, in bytes, for unchunking request data. For more information, see Service.
`ChunkedRequestTimeout`	(Optional) Common to all `Service`-class functions. Determines the default timeout, in seconds, for unchunking request data. For more information, see Service.

Example

```
Service fn="remove-filter"
```

5.9.3 service-proxy-cache-dump

The `service-proxy-cache-dump` function dumps the current reverse proxy caching statistics.

Table 5–47 describes the parameters for the `service-proxy-cache-dump` function.

Table 5–47 service-proxy-cache-dump Parameters

Parameter	Description
`list`	Lists the objects in the cache. The cache listing includes the URI, a set of flags, the current number of references to the cache entry and the size of the entry.
`refresh=n`	Setting this parameter to a value `n` causes the client to reload the page every `n` seconds.

Table 5–47 (Cont.) service-proxy-cache-dump Parameters

Parameter	Description
`restart`	Stops and restarts the cache.
`start`	Starts the cache.
`stop`	Stops the cache.

Example

```
<Object name="default"
NameTrans fn=assign-name name="proxy-cache" from="/.proxycache"
</Object>
<Object name="proxy-cache">
    Service fn="service-proxy-cache-dump"
</Object>
```

See Also: proxy-cache-config, proxy-cache-override-http

5.9.4 service-trace

The `service-trace` function services `TRACE` requests. `TRACE` requests are used to diagnose problems with web proxy servers located between a web client and web server.

Table 5–48 describes the parameters for the `service-trace` function.

Table 5–48 service-trace Parameters

Parameter	Description
`type`	(Optional) Common to all `Service`-class functions. Specifies a wildcard pattern of MIME types for which this function is executed. For more information, see Service.
`method`	(Optional) Common to all `Service`-class functions. Specifies a wildcard pattern of HTTP methods for which this function is executed. For more information, see Service.
`query`	(Optional) Common to all `Service`-class functions. Specifies a wildcard pattern of query strings for which this function is executed. For more information, see Service.
`UseOutputStreamSize`	(Optional) Common to all `Service`-class functions. Determines the default output stream buffer size (in bytes), for data sent to the client. For more information, see Service.
`flushTimer`	(Optional) Common to all `Service`-class functions. Determines the maximum number of milliseconds between write operations in which buffering is enabled. For more information, see Service.
`ChunkedRequestBufferSize`	(Optional) Common to all `Service`-class functions. Determines the default buffer size, in bytes, for unchunking request data. For more information, see Service.
`ChunkedRequestTimeout`	(Optional) Common to all `Service`-class functions. Determines the default timeout, in seconds, for unchunking request data. For more information, see Service.

Example

```
<Object name="default">
...
```

```
Service method="TRACE" fn="service-trace"
...
</Object>
```

5.9.5 stats-xml

The `stats-xml` function creates a performance report in XML format. If performance buckets are defined, this performance report includes them.

The report is generated at:

`http://`*`server_id`*`:`*`portURI`*

For example:

`http://example.com:80/stats-xml`

The following table describes parameters for the `stats-xml` function.

Table 5–49 stats-xml Parameters

Parameter	Description
`type`	(Optional) Common to all `Service`-class functions. Specifies a wildcard pattern of MIME types for which this function is executed. For more information, see Service.
`method`	(Optional) Common to all `Service`-class functions. Specifies a wildcard pattern of HTTP methods for which this function is executed. For more information, see Service.
`query`	(Optional) Common to all `Service`-class functions. Specifies a wildcard pattern of query strings for which this function is executed. For more information, see Service.
`UseOutputStreamSize`	(Optional) Common to all `Service`-class functions. Determines the default output stream buffer size (in bytes), for data sent to the client. For more information, see Service.
`flushTimer`	(Optional) Common to all `Service`-class functions. Determines the maximum number of milliseconds between write operations in which buffering is enabled. For more information, see Service.
`ChunkedRequestBufferSize`	(Optional) Common to all `Service`-class functions. Determines the default buffer size, in bytes, for unchunking request data. For more information, see Service.
`ChunkedRequestTimeout`	(Optional) Common to all `Service`-class functions. Determines the default timeout, in seconds, for unchunking request data. For more information, see Service.

Example

```
<Object name="default">
<If uri = "/stats-xml/*">
Service fn="stats-xml"
</If>
...
</Object>
```

5.10 AddLog

The `AddLog` directives are executed to record information about the transaction. For more information, see AddLog.

The following AddLog-class function is described in detail in this section:

- flex-log

In addition, the following common SAFs are valid for the AddLog directive:

- match-browser
- set-variable

5.10.1 flex-log

The flex-log function records request-specific data in a flexible log format. It can also record requests in the common log format. There is a log analyzer, flexanlg, in the /bin directory for Web Server. There are also a number of free statistics generators for the common log format.

Specify the log format using the format subelement of the access-log element in server.xml. For more information, see access-log. For more information about the log format, see "Using the Custom Access-Log File Format".

Table 5–50 describes the parameters for the flex-log function.

Table 5–50 flex-log Parameters

Parameter	Description
name	(Optional) Specifies the name of a log file. The name must previously been defined by an access-log element in server.xml. If no name is given, the entry is recorded in the default log file.
iponly	(Optional) Instructs Oracle Traffic Director to log the IP address of the remote client rather than looking up and logging the DNS name. This improves performance if DNS is turned off. The value of iponly has no significance, as long as it exists; you can use iponly=1.

Example

```
# Log all accesses to the default log file
AddLog fn="flex-log"
# Log accesses from outside our subnet (198.93.5.*) to
# nonlocallog
<Client ip="*~198.93.5.*">
AddLog fn="flex-log" name="nonlocallog"
</Client>
```

5.11 Error

If an SAF results in an error, Oracle Traffic Director stops executing all other directives and immediately starts executing the Error directives. For more information, see Error.

The following Error-class functions are described in detail in this section:

- qos-error
- send-error

In addition, the following common SAFs are valid for the Error directive:

- match-browser
- redirect
- remove-filter

- restart
- set-variable

5.11.1 qos-error

The `qos-error` function returns an error page stating the quality of service that caused the error and the value of the QoS statistic.

Table 5–51 describes the parameters for the `qos-error` function.

Table 5–51 qos-error Parameters

Parameter	Description
`code`	(Optional) Three-digit number representing the HTTP response status code, such as `401` or `407`. The recommended value is `503`. This can be any HTTP response status code or reason phrase according to the HTTP specification. A list of common HTTP response status codes and reason strings is as follows: ■ `401 Unauthorized` ■ `403 Forbidden` ■ `404 Not Found` ■ `500 Server Error`

Example

```
Error fn="qos-error" code="503"
```

Note: qos-handler

5.11.2 send-error

The `send-error` function sends an HTML file to the client in place of a specific HTTP response status. This allows the server to present a message describing the problem. The HTML page may contain images and links to the server's home page or other pages.

Table 5–52 describes the parameters for the `send-error` function.

Table 5–52 send-error Parameters

Parameter	Description
`path`	Specifies the absolute path of an HTML file to send to the client. If the file does not exist or is not accessible, the server returns a 404 or 403 error page. The file is sent as text/html regardless of its name or actual type.
`code`	(Optional) Three-digit number representing the HTTP response status code, such as `401` or `407`. This can be any HTTP response status code or reason phrase according to the HTTP specification. A list of common HTTP response status codes and reason strings is as follows: ■ `401 Unauthorized` ■ `403 Forbidden` ■ `404 Not Found` ■ `500 Server Error`

Table 5–52 (Cont.) send-error Parameters

Parameter	Description
`type`	(Optional) Common to all `Service`-class functions. Specifies a wildcard pattern of MIME types for which this function will be executed. For more information, see Service.
`method`	(Optional) Common to all `Service`-class functions. Specifies a wildcard pattern of HTTP methods for which this function will be executed. For more information, see Service.
`query`	(Optional) Common to all `Service`-class functions. Specifies a wildcard pattern of query strings for which this function will be executed. For more information, see Service.
`bucket`	(Optional) Common to all `obj.conf` functions. Adds a bucket to monitor performance. For more information, see The bucket Parameter.

Example

```
Error fn="send-error" code="401"
path="/opt/oracle/webserver7/docs/errors/401.html"
```

5.12 Common SAFs

This section lists SAFs that are common to multiple directives.

Table 5–53 Common SAFs

Server Application Functions	Directives
insert-filter	▪ Input ▪ Output
match-browser	▪ AuthTrans ▪ NameTrans ▪ PathCheck ▪ ObjectType ▪ Input ▪ Output ▪ Route ▪ Service ▪ AddLog ▪ Error
redirect	▪ NameTrans ▪ Output ▪ Error
remove-filter	▪ Input ▪ Output ▪ Service ▪ Error
restart	▪ NameTrans

Table 5–53 (Cont.) Common SAFs

Server Application Functions	Directives
set-variable	■ AuthTrans ■ NameTrans ■ PathCheck ■ ObjectType ■ Input ■ Output ■ Route ■ Service ■ AddLog ■ Error

5.12.1 insert-filter

The `insert-filter` SAF is used to add a filter to the filter stack to process incoming (client to server) data. The order of `Input fn="insert-filter"` and `Output fn="insert-filter"` directives is important.

Returns

Returns `REQ_PROCEED` if the specified filter was inserted successfully or `REQ_NOACTION` if the specified filter was not inserted because it was not required. Any other return value indicates an error.

Parameters

The following table describes parameters for the `insert-filter` function.

Table 5–54 `insert-filter` Parameters

Parameter	Description
`filter`	Specifies the name of the filter to insert. For more information about predefined filters, see Input and Output.
`type`	(Optional) Common to all `Input`-class and `Output`-class functions. Specifies a wildcard pattern of MIME types for which this function is executed.
`method`	(Optional) Common to all `Input`-class and `Output`-class functions. Specifies a wildcard pattern of HTTP methods for which this function is executed. Common HTTP methods are `GET`, `HEAD`, and `POST`.
`query`	(Optional) Common to all `Input`-class and `Output`-class functions. Specifies a wildcard pattern of query strings for which this function is executed.

5.12.1.1 Example

```
Input fn="insert-filter" filter="http-decompression"
```

This directive instructs the `insert-filter` function to add a custom filter, that is, `http-decompression` to the filter stack. The `http-decompression` filter will decompress the incoming HTTP request data before it goes to the service stage. For more information about predefined filters, see Input and Output.

5.12.2 match-browser

The `match-browser` function matches specific strings in the `User-Agent` string supplied by the browser. It then modifies the behavior of Web Server based on the results by setting values for specified variables. This function is applicable in all directives.

Syntax

stage `fn="match-browser" browser=`*"string" name="value"* [*name="value"* . . .]

Parameters

The following table describes parameter values for the `match-browser` function.

Table 5–55 match-browser Parameters

Value	Description
stage	Stage directive used in `obj.conf` processing. The `match-browser` function is applicable in all stage directives.
string	Wildcard pattern to compare with the `User-Agent` header (for example, `"*Mozilla*"`).
name	Variable to be changed. The `match-browser` function indirectly invokes the `set-variable` function.
value	New value for the specified variable.

Example

```
AuthTrans fn="match-browser"
          browser="*[Bb]roken*"
          ssl-unclean-shutdown="true"
          keep-alive="disabled"
          http-downgrade="1.0"
```

If a browser's `User-Agent` header contains the string `Broken` or `broken`, the above `AuthTrans` directive instructs Oracle Traffic Director to do the following:

- Not send the SSL3 and TLS `close_notify` packet
- Not honor requests for HTTP Keep-Alive
- Use the HTTP/1.0 protocol rather than HTTP/1.1

For more information on the variables used in this example, such as`ssl-unclean-shutdown`, see set-variable.

See Also: set-variable

5.12.3 redirect

The `redirect` function lets you change URLs and send the updated URL to the client. When a client accesses your server with an old path, Oracle Traffic Director treats the request as a request for the new URL.

The `redirect` function inspects the URL to which the client will be redirected. If the URL matches the URL the client has requested (same scheme, hostname, port, and path), this function does not perform the redirect and instead returns `REQ_NOACTION`.

Table 5–56 describes the parameters for the `redirect` function.

Table 5–56 redirect Parameters

Parameter	Description
`from`	(Optional) Specifies the prefix of the requested URI to match. If `from` is not specified, it defaults to `""`.
`url`	(Optional) Specifies a complete URL to return to the client. If you use this parameter, do not use `url-prefix`.
`url-prefix`	(Optional) The new URL prefix to return to the client. The `from` prefix is replaced by this URL prefix. If you use this parameter, do not use `url`.
`escape`	(Optional) Indicates whether the value of the `url` or `url-prefix` parameter needs to be escaped. The default is `yes`, indicating that Oracle Traffic Director will escape the value. The value `no` indicates that the URL or URL prefix value has already been escaped. An example of an escaped value is one where any % characters have been replaced with %25 and any spaces have been replaced with %20.
`status`	(Optional) Customizes the HTTP status code. If `status` is not specified, it defaults to `302`.
`type`	(Optional) Common to all `Output`-class functions. Specifies a wildcard pattern of MIME types for which this function is executed.
`method`	(Optional) Common to all `Output`-class functions. Specifies a wildcard pattern of HTTP methods for which this function is executed. Common HTTP methods are `GET`, `HEAD`, and `POST`.
`query`	(Optional) Common to all `Output`-class functions. Specifies a wildcard pattern of query strings for which this function is executed.

Example

In the first example, any request for `http://`*server-name*`/`*whatever* is translated to a request for `http://tmpserver/`*whatever*.

```
NameTrans fn="redirect" from="/" url-prefix="http://tmpserver/"
```

In the second example, any request for `http://`*server-name*`/toopopular/`*whatever* is translated to a request for `http://bigger/better/stronger/morepopular/`.

```
NameTrans fn="redirect" from="/toopopular"
     url="http://bigger/better/stronger/morepopular"
```

See Also: restart

5.12.4 remove-filter

The `remove-filter` SAF is used to remove a filter from the filter stack. If the filter is inserted multiple times, only the topmost instance is removed. In general, it is not necessary to remove filters with `remove-filter`, as they are removed automatically at the end of a request.

Returns

Returns `REQ_PROCEED` if the specified filter was removed successfully, or `REQ_NOACTION` if the specified filter was not part of the filter stack. Any other return value indicates an error.

Parameters

The following table describes parameters for the `remove-filter` function.

Table 5–57 remove-filter Parameters

Parameter	Description
filter	Specifies the name of the filter to remove.
type	(Optional) Common to all Input-class, Output-class, and Service-class functions. Specifies a wildcard pattern of MIME types for which this function is executed. The magnus-internal/* MIME types are used only to select a Service function to execute.
method	(Optional) Common to all Input-class, Output-class, and Service-class functions. Specifies a wildcard pattern of HTTP methods for which this function is executed. Common HTTP methods are GET, HEAD, and POST.
query	(Optional) Common to all Input-class, Output-class, and Service-class functions. Specifies a wildcard pattern of query strings for which this function is executed.
UseOutputStreamSize	(Optional) Common to all Service-class functions. Determines the default output stream buffer size (in bytes), for data sent to the client. For more information, see Service.
flushTimer	(Optional) Common to all Service-class functions. Determines the maximum number of milliseconds between write operations in which buffering is enabled. For more information, see Service.
ChunkedRequestBufferSize	(Optional) Common to all Service-class functions. Determines the default buffer size, in bytes, for unchunking request data. For more information, see Service.
ChunkedRequestTimeout	(Optional) Common to all Service-class functions. Determines the default timeout, in seconds, for unchunking request data. For more information, see Service.

5.12.4.1 Example

```
Input fn="remove-filter" filter="http-compression"
```

5.12.5 restart

The restart function allows URL rewriting within Oracle Traffic Director without sending an HTTP redirect to the client. The restart function replaces the uri and query values inrq->reqpb with the URI and query string specified by the uri parameter and restarts the request by returning REQ_RESTART.

If the uri parameter contains a ? character, the value following ? is used as the query string. Otherwise, the restarted request will not have a query string. Because the new request URI will be passed through the AuthTrans and NameTrans stages again, avoid creating infinite loops.

The following table describes parameters for the restart function.

Table 5–58 restart Parameters

Parameter	Description
from	(Optional) Wildcard pattern that specifies the path of requests that should be restarted. The default is to match all paths.
uri	URI and query string to use for the restarted request.

Example

The following `obj.conf` code directs Oracle Traffic Director to service requests for `/index.html` as though they were requests for `/index.jsp`:

```
NameTrans fn="restart" from="/index.html" uri="/index.jsp"
```

5.12.6 set-variable

The `set-variable` function enables you to change Oracle Traffic Director settings based upon conditional information in a request. This function is applicable in all directives.

It can also be used to manipulate variables in parameter blocks with the following commands:

- `insert-`*`pblock`*`="`*`name=value`*`"`

  Adds a new value to the specified *pblock*.

- `set-`*`pblock`*`="`*`name=value`*`"`

  Sets a new value in the specified *pblock,* replacing any existing values with the same name.

- `remove-`*`pblock`*`="`*`name`*`"`

  Removes all values with the given name from the specified *pblock*.

The `set-variable` function recognizes many predefined variables as parameters. Additionally, when a `set-variable` parameter name begins with `$` but is not the name of a predefined variable, the parameter and its value are stored in the `rq->vars` pblock. This functionality allows you to define or override the `$variable` values at the request time.

`set-variable` accepts both the `$variable` and `${variable}` forms, but the name of the parameter stored in the `rq->vars` pblock is always in the `$variable` form.

Syntax

`stage` `fn="set-variable" [{insert|set|remove}-`*`pblock="name=value"`* `...][`*`name="value"`* `...]`

Parameters

The following table describes parameter values for the `set-variable` function.

Table 5–59 set-variable Parameters

Value	Description
pblock	Specifies one of the following session or request parameter block names: ■ `client`: Contains the IP address of the client machine and the DNS name of the remote machine. ■ `vars`: Contains the server's working variables, which includes anything not specifically found in the `reqpb`, `headers`, or `srvhdrs` pblocks. The contents of this pblock differ, depending on the specific request and the type of SAF. ■ `reqpb`: Contains elements of the HTTP request, which includes the HTTP method such as GET or POST, the URI, the protocol (generally HTTP/1.0), and the query string. This pblock does not change during the request-response process. ■ `headers`: Contains all the request headers (such as `User-Agent`, `If-Modified-Since`, and so on) received from the client in the HTTP request. This pblock does not change during the request-response process. ■ `srvhdrs`: Contains the response headers (such as `Server`, `Date`, `Content-Type`, `Content-length`, and so on) that are to be sent to the client in the HTTP response.
name	The variable to set.
value	The string assigned to the variable specified by *name*.

Variables

The following tables lists variables supported by the `set-variable` SAF.

Table 5–60 Supported Variables

Variable	Description
`abort`	A value of `true` indicates that the result code should be set to `REQ_ABORTED`. Setting the result code to `REQ_ABORTED` will abort the current request and send an error to the browser. For information about result codes, see "Creating Custom Server Application Functions" in *Oracle Traffic Director Administration Guide*.

`error`	Sets the HTTP status code and exits the request by returning `REQ_ABORTED`. To set the HTTP status code without exiting the request, use the set-variable error parameter along with the `noaction` parameter. To rewrite an HTTP status code, use a `Client` tag to match the original status code and an `Output` directive to set the new status code.
	For example, the following code will rewrite all `302 Moved Temporarily` responses to `301 Moved Permanently` responses:
	`<Client code="302">`
	`Output fn="set-variable" error="301 Moved Permanently"`
	`noaction="true"`
	`</Client>`
	Sets the error code to be returned in the event of an aborted browser request.

`escape`	A Boolean value signifying whether a URL should be escaped using `util_uri_escape`. For information about `util_uri_escape`, see *Oracle Traffic Director Administration Guide*.
`find-pathinfo-forward`	Path information after the file name in a URI.

Table 5–60 (Cont.) Supported Variables

Variable	Description
http-downgrade	HTTP version number (for example, 1.0).
http-upgrade	HTTP version number (for example, 1.0).
keep-alive	A Boolean value that establishes whether a keep-alive request from a browser will be honored.
name	Specifies an additional named object in the obj.conf file whose directives will be applied to this request. See also assign-name.
noaction	A value of true indicates the result code should be set to REQ_NOACTION. For AuthTrans, NameTrans, Service, and Error stage SAFs, setting the result code to REQ_NOACTION indicates that subsequent SAFs in that stage should be allowed to execute. For information about result codes, see "Creating Custom Server Application Functions" in *Oracle Traffic Director Administration Guide*.
nostat	Causes the server *not* to perform the stat() function for a URL when possible. See also assign-name.
senthdrs	A Boolean value that indicates whether HTTP response headers have been sent to the client.
ssl-unclean-shutdown	A Boolean value that can be used to alter the way SSL3 connections are closed. **Caution:** As this violates the SSL3 RFCs, you should only use this with great caution if you know that you are experiencing problems with SSL3 shutdowns.
stop	A value of true indicates the result code should be set to REQ_PROCEED. For AuthTrans, NameTrans, Service, and Error stage SAFs, setting the result code to REQ_PROCEED indicates that no further SAFs in that stage should be allowed to execute.
url	Redirect requests to a specified URL.

Examples

- To deny HTTP keep-alive requests for a specific server class (while still honoring keep-alive requests for the other classes), add this AuthTrans directive to the obj.conf for the server class, and set the variable keep-alive to disabled:

  ```
  AuthTrans fn="set-variable" keep-alive="disabled"
  ```

- To set the same server class to use HTTP/1.0 while the rest of Oracle Traffic Director classes use HTTP/1.1, the AuthTrans directive is:

  ```
  AuthTrans fn="set-variable" keep-alive="disabled" http-downgrade="1.0"
  ```

- To insert an HTTP header into each response, add a NameTrans directive to obj.conf using the insert-*pblock* command and specify srvhdrs as your Session or Request parameter block.

  For example, to insert the HTTP header P3P, add the following line to each request:

  ```
  NameTrans fn="set-variable" insert-srvhdrs="P3P"
  ```

- To terminate processing a request based on certain URIs, use a Client tag to specify the URIs and an AuthTrans directive that sets the variable abort to true when there is a match. Your Client tag would be as follows:

  ```
  <Client uri="*(system32|root.exe)*">
  ```

```
AuthTrans fn="set-variable" abort="true"
</Client>
```

- To use predefined variables so that Oracle Traffic Director rewrites redirects to host *badname* as redirects to host *goodname*:

```
<If $srvhdrs{'location'} =~ "^(http|https)://badname/(.*)$"
Output fn="set-variable" $srvhdrs{'location'}="$1://goodname/$2"
</If>
```

- To set a `$variable` value at request time:

```
<If "$time_hour:$time_min" < "8:30" || "$time_hour:$time_min" > "17:00">
AuthTrans fn="set-variable" $docroot="/var/www/docs/closed"
</If>
...
NameTrans fn="document-root" root="$docroot"
```

Regardless of whether the `$docroot` variable has been defined in `server.xml`, its value is set to `/var/www/docs/closed` when Oracle Traffic Director is accessed after 5:00 p.m. and before 8:00 a.m. local time.

See Also: match-browser

A

Using Variables, Expressions, Wildcards, and String Interpolation

This appendix contains the following topics:

A.1 If, ElseIf, and Else Tags

The `If`, `ElseIf`, and `Else` tags enable you to define the conditions under which a set of directives will be executed. These tags can only appear inside an `Object` tag. In addition, these tags can evaluate an expression, then conditionally execute one or more contained directives. The usage of these tags is summarized below:

- `If` and `ElseIf` tags offer a richer expression syntax, including support for regular expressions. For more information about the `If` and `ElseIf` expression syntax, see Expressions.
- `If`, `ElseIf`, and `Else` tags can contain other tags.
- `If` and `ElseIf` expressions are evaluated once per request, not once per contained directive.
- `If`, `ElseIf`, and `Else` tags cannot contain multiple types of directives.
- Directives within the `If` and `ElseIf` tags can contain regular expression backreferences.

When used, an `ElseIf` or `Else` tag must immediately follow an `If` or `ElseIf` tag. `ElseIf` and `Else` tags are skipped if the preceding `If` or `ElseIf` expression evaluates to logical true.

The following example shows `If`, `ElseIf`, and `Else` tag syntax:

```
<If $path eq "/">
<If $browser =~ "MSIE">
NameTrans fn="rewrite" path="/msie.html"
</If>
<ElseIf $browser =~ "Mozilla">
NameTrans fn="rewrite" path="/mozilla.html"
</ElseIf>
<Else>
```

```
NameTrans fn="rewrite" path="/unknown.html"
</Else>
</If>
```

This example displays a different page based on whether the browser is Microsoft Internet Explorer, Mozilla Firefox, or another browser.

A.2 Variables

The Oracle Traffic Director includes a set of variables predefined by the server, and the capability for you to define custom variables. This section includes the following topics:

- Predefined Variables
- Custom Variables
- Resolving Variables

A.2.1 Predefined Variables

Predefined variables are implicitly defined by the server. The following table lists the predefined variables and their descriptions:

Variable	Description
`$`*n*	Regular expression backreference (value of the *n*th capturing subpattern, *n* = 1...9), for example, `$1`.
	Regular expression backreferences are only available within the body of `If` and `ElseIf` containers, and only if the container expressions includes one or more regular expressions.
`$&`	Value that matched a regular expression.
	Regular expression backreferences are only available within the body of `If` and `ElseIf` containers, and only if the container expressions includes one or more regular expressions.
`$`*n*	Regular expression backreference (value of the *n*th capturing subpattern, *n* = 1...9), for example, `$1`.
	Regular expression backreferences are only available within the body of `If` and `ElseIf` containers, and only if the container expressions includes one or more regular expressions.
`$&`	Value that matched a regular expression.
	Regular expression backreferences are only available within the body of `If` and `ElseIf` containers, and only if the container expressions includes one or more regular expressions.
`$body`	Contains the entity-body (if any) of the current HTTP request that OTD receives from the client (browser). The size of the data stored in this variable is configured using the `body-buffer-size` sub-element of the http element in server.xml.
`$browser`	Web browser version (alias for `$headers{'user-agent'}` if the client sent a `User-Agent` header or an empty string).
`$chunked`	Boolean variable that indicates if the request body was sent using chunked encoding.
`$code`	Response status code.
`$cookie{'`*name*`'}`	Value of the cookie *name* from the request.
`$dns`	Alias for `$client{'dns'}`.

Variable	Description
`$env{'name'}`	Value of the environment variable *name* (includes CGI/SHTML environment variables).
`$headers{'name'}`	Value of *name* from `rq->headers`, that is, value of the request header name where *name* is a lowercase string.
`$id`	Virtual server ID as specified by the *name* subelement of the `virtual-server` element in the `server.xml` file.
`$internal`	Boolean value that indicates whether request was internally generated.
`$ip`	Alias for `$client{'ip'}`.
`$keep_alive`	Boolean value that indicates if the connection is kept open.
`$keysize`	Alias for `$client{'keysize'}`.
`$method`	Request method (alias for `$reqpb{'method'}`).
`$path`	Requested path (either URI, partial path, or file system path depending on stage). The predefined variable `path` is the value of `path` from `rq->vars`. If `path` is not set in `rq->vars` (for example, if NameTrans has not completed), `path` gets the value of `ppath` from `rq->vars`.
`$path_info`	Alias for `$vars{'path-info'}`.
`$ppath`	Alias for `$vars{'ppath'}`.
`$protocol`	Request protocol (alias for `$reqpb{'protocol'}`).
`$query`	Request query string (alias for `$reqpb{'query'}`).
`$reason`	Response reason phrase.
`$referer`	Alias for `$headers{'referer'}`.
`$reqpb{'name'}`	Value of *name* from `rq->reqpb`.
`$restarted`	Boolean value that indicates if the request was restarted.
`$secret_keysize`	Alias for `$client{'secret-keysize'}`.
`$server_url`	Prefix for self-referencing URLs.
`$time`	Time the request was received as the number of seconds since 00:00:00 UTC, January 1, 2006.
`$time_day`	Day of the month when the request was received. Value can be from 01 to 31.
`$time_hour`	Hours since midnight when the request was received. Value can be from 00 to 23.
`$time_min`	Minutes after the hour when the request was received. Value can be from 00 to 59.
`$time_mon`	Month of the year when the request was received. Value can be from 01 to 12.
`$time_sec`	Seconds after the minute when the request was received. Value can be from 00 to 61.
`$time_wday`	Day of the week when the request was received. Value can be from 0 to 6, where 0 corresponds to Sunday.
`$time_year`	Four-digit year when the request was received.
`$type`	Alias for `$srvhdrs{'content-type'}`.
`$uri`	URI of the requested resource (alias `for $reqpb{'uri'}`).

Variable	Description
`$url`	URL of the requested resource.
`$urlhost`	Host name to which the client connected.
`$vars{'name'}`	Value of *name* from `rq->vars`.
`$security`	Boolean value that indicates if a secure transport was used.
`$senthdrs`	Boolean value that indicates if a response headers were been sent.
`$srvhdrs{'name'}`	Value of *name* from `rq->srvhdrs`, that is, value of response header name where *name* is a lowercase string.

A.2.2 Custom Variables

You can define custom variables in the `server.xml` file using the `variables` element. These variables can then be used in function parameters in `obj.conf` functions. You can also define variables at request time using the `set-variables` function in the `obj.conf` file.

> **Note:** The predefined variables take precedence over custom variables. It is a best practice to use uppercase names for custom variables. Using uppercase avoids conflicts with the lowercase predefined variables, if the list of predefined variables is extended in the future.

A.2.3 Resolving Variables

The server uses the following order when resolving a `$variable`:

1. Predefined variables
2. Variables defined at request time using `set-variable` in `obj.conf`
3. Variables defined by the `virtual-server` element's `variable` subelement in `server.xml`
4. Variables defined by the `server` element's `variable` subelement in `server.xml`

When you define a `$variable` at request time, it is stored as a name-value pair in the `rq->vars pblock`. These variables are given a higher precedence than `server.xml` variables so that `server.xml` variables can be overridden at request time.

A.3 Expressions

Expressions enable you to dynamically construct server application function (SAF) parameters and to select which SAFs to execute on a request-by-request basis. Expressions are constructed from literals, variables, functions, and operators. Use expressions in `If` and `ElseIf` tags, in log format strings, and SAF parameters.

This section contains the following topics:

- Expression Syntax
- Expression Results as Boolean Values
- Expression Literals
- Expression Variables
- Expression Operators

- Expression Functions
- Regular Expressions

A.3.1 Expression Syntax

The expression syntax is similar to the syntax used in Perl. Expressions are constructed from literals, variables, functions, and operators.

The following example illustrates how to use expressions in an `If` tag:

```
<If not $internal
    and $uri =~ "^/private/(.*)$"
    and $referer !~ "^https?://example.com/">
NameTrans fn="redirect"
          url="http://example.com/denied.jsp?file=$1"
</If>
```

This example expression checks to see if a request meets certain criteria, for example if it is an internal request. If it does not meet the criteria, the server redirects the request to a request denied URL.

The expression contains the following components:

- Literals - `"^/private/(.*)$"` and `"^https?://example.com/"`
- Variables - `$internal`, `$uri`, and `$referer`
- Operators - `not`, `and`, `=~`, and `!~`

For more information about `If` and `ElseIf` tags, see "If, ElseIf, and Else Tags".

A.3.2 Expression Results as Boolean Values

In some circumstances, for example, after evaluating an `If` or `ElseIf` expression, the server must treat the result of an expression as a Boolean value. The server uses the following rules when converting a numeric value to a Boolean value:

- Zero evaluates to false.
- All other numeric values evaluate to true.

The server uses the following rules when converting a string to a Boolean value:

- Zero-length strings evaluate to false.
- The string 0 (zero) evaluates to false.
- All other strings evaluate to true.

A.3.3 Expression Literals

Expression literals are divided into string and numeric literals.

A.3.3.1 String Literals

A string literal is enclosed by either single quotation marks (`'`) or double quotation marks (`"`). When single quotation marks enclose a string literal, the value of the literal is the value within the quotation marks. When double quotation marks are used, any references to variables or expressions within the quotation marks are interpolated. For more information, see String Interpolation.

The following expression examples show the use of single and double quotation marks.

```
# This expression evaluates to true.
('foo' eq "foo")

# This expression evaluates to false.
('foo' eq "bar")

# This expression evaluates to true.
('foo' eq "f$(lc('O'))o")

# This expression may evaluate to true or false,
# depending on the value of the variable $foo
('$foo' eq "$foo")
```

To include an uninterpolated dollar sign `$` in a string enclosed in a double quotation marks, use the two dollar sign or a backslash dollar sign `$$` or `\$` escape sequences.

When a double quotation marks appears within a literal enclosed by double quotation marks, it must be prefixed with a backslash. When a single backslash (`\`) appears within a literal bracketed by double quotes, it must be prefixed with a backslash. When a single quote character appears within a literal bracketed by single quotes, it must be prefixed with a backslash.

The following examples show valid and invalid literals:

```
# The following are examples of valid literals
'this string literal is bracketed by single quotes'
"this string literal is bracketed by double quotes"
"the backslash, \\, escapes characters in double quote string literals"
'it\'s easy to use strings literals'

# The following are examples of invalid literals
'it's important to escape quote characters'
"any \ characters in double quote string literals must be escaped"
```

A.3.3.2 Numeric Literals

A numeric literal can consist of decimal digits and an optional decimal point, a leading zero followed by octal digits, or a leading `0x` prefix followed by hexadecimal digits. Hexadecimal and octal numbers are automatically converted to decimal form.

The following examples show expressions that use numeric literals:

```
# The following expressions evaluate to true
(1 < 2)
(0x10 == "16")
(1 == 1.00)

# The following expressions evaluate to false
(1 > 2)
("0x10" == 16)
(1 != 1.00)
```

A.3.4 Expression Variables

Any `$variable` can be used as a variable in an expression. To mirror the `Client` tag syntax, the dollar sign `$` prefix is optional for predefined variable names in expressions. For example, the following three portions of the `obj.conf` file are semantically equivalent:

```
<If $uri = "*.html">
```

```
...
</If>

<If uri = "*.html">
...
</If>

<Client uri = "*.html">
...
</Client>
```

Any variable names you define must use the `$` prefix. For example, the following expression is invalid even if `somecustomvariable` is defined in a `server.xml` `variable` element:

```
<If somecustomvariable = "foo">
...
</If>
```

To make this expression valid, add the dollar sign prefix:

```
<If $somecustomvariable = "foo">
...
</If>
```

A.3.5 Expression Operators

The following table lists the operators that are used in expressions.

Operator Symbol	Operator Name
`!`	C-style logical not
`=`	Wildcard pattern match
`=~`	Regular expression match
`!~`	Regular expression mismatch
`+`	Addition or unary plus
`-`	Subtraction or unary minus
`.`	String concatenation
defined	Value is defined
`-d`	Directory exists
`-e`	File or directory exists
`-f`	File exists
`-l`	Symbolic link exists
`-r`	File is readable
`-s`	File size
`-U`	URI maps to accessible file or directory
`<`	Numeric less than
`<=`	Numeric less than or equal to
`>`	Numeric greater than

Operator Symbol	Operator Name
>=	Numeric greater than or equal to
lt	String less than
le	String less than or equal to
gt	String greater than
ge	String greater than or equal to
==	Numeric equal
!=	Numeric not equal
eq	String equal
ne	String not equal
^	C-style exclusive or
&&	C-style logical and
||	C-style logical or
not	Logical not
and	Logical and
or	Logical or
xor	Logical exclusive or

The following table lists the precedence of operators within expressions from highest to lowest precedence.

Symbol	Operands	Associativity	Description
( ), [ ]	0	Left to right	Parentheses
!, unary +, unary -	1	Right to left	Sign operators
=, =~, !~	2	Non-associative	Pattern matching operators
+, -, .	2	Non-associative	Additive operators
defined, -d, -f, -l, -r, -s, -U	1	Right to left	Named operators
<, lt, <=, le, >, gt, >=, ge	2	Non-associative	Relational operators
==, eq, !=, ne	2	Non-associative	Equality operators
^	2	Left to right	C-style exclusive or operator
&&	2	Left to right	C-style logical and operator
||	2	Left to right	C-style logical or operator
not	1	Right to left	Logical not operator
and	2	Left to right	Logical and operator
or, xor	2	Left to right	Logical or operators

The numeric operators (<, <=, >, >=, ==, and !=) are intended to operate on numbers and not strings. To facilitate comparing numbers, dates, and timestamps, the numeric operators ignore any white space, colons, slashes, and commas in their arguments. Dashes after the first digit are also ignored.

> **Note:** It is generally incorrect to use the numeric operators on non-numeric values.

For example, the following expression evaluates to true:

```
# The following expression evaluates to true because both
# "foo" and "bar" are numerically equivalent to 0
("foo" == "bar")
```

A.3.6 Expression Functions

Expression functions manipulate data for use in expressions. Expression functions are different from SAFs. While SAFs perform the actual work associated with an HTTP request, expression functions are used to select which SAFs run and what parameters to pass to the SAFs.

Some expression functions require one or more arguments. An expression function's argument list is enclosed in parentheses (()) and the individual arguments are separated by commas (,).

The individual expression functions are listed in the following sections:

- atime
- choose
- ctime
- escape
- external
- httpdate
- lc
- length
- lookup
- lookupregex
- mtime
- owner
- uc
- unescape
- uuid

A.3.6.1 atime

The atime function returns the time of the last access for the specified file or directory.

Syntax

`atime` (path)

Arguments

The following table describes the argument for the expression function.

Argument	Description
path	The absolute path to the directory or file name for which you are requesting the last access.

See Also: ctime, mtime

A.3.6.2 choose

The `choose` function parses pipe-separated values from *values* and returns one at random.

Syntax

```
choose (values)
```

Arguments

The following table describes the argument for the expression function.

Argument	Description
values	The list of values to choose from, separated by the pipe character (`|`)

Example

The following `obj.conf` code demonstrates the use of `choose` to randomly select one of three images:

```
NameTrans fn="rewrite"
          from="/images/random"
          path="/images/$(choose('iwsvi.jpg|0061.jpg|webservervii.jpg'))"
```

A.3.6.3 ctime

The `ctime` function returns the time of the last status change for the specified file or directory.

Syntax

```
ctime (path)
```

Arguments

The following table describes the argument for the expression function.

Argument	Description
path	The absolute path to the directory or file name for which you are requesting the last status change

See Also: atime, mtime

A.3.6.4 escape

The `escape` function encodes the URI using `util_uri_escape`, converting special octets to their percentage encoded equivalent and returns the result.

Syntax

```
escape(uri)
```

Arguments

The following table describes the argument for the expression function.

Argument	Description
uri	The URI that the expression function converts

See Also: unescape

A.3.6.5 external

The `external` function passes a value to an external rewriting program and returns the result.

Each invocation of `external` results in a single newline-terminated line being written to the external rewriting program's `stdin`. For each line of input, the program must produce a single line of output. When developing an external rewriting program, it is important to avoid buffering `stdout`. In Perl, for example, `$| = 1;` is used to disable buffering. Because the server expects the external rewriting program to produce one line of output for each line of input, the server can stop responding if the external rewriting program buffers its output.

Syntax

```
external (program, value)
```

Arguments

The following table shows the arguments for the `external` function.

Argument	Description
program	The program argument is the file name of an external rewriting program. Because program is executed using the operating system's default shell (`/bin/sh on Unix/Linux`) or the command interpreter (`CMD.EXE` on Windows), *program* should be an absolute path or the name of a program in the operating system's `PATH`. The server starts the external rewriting program on demand. A given server process never executes more than one instance of the program at a time.
value	The value passed to the rewriting program.

Example

The following is an example of an external rewriting program rewrite.pl, used to change the prefix `/home/ to /u/`:

```
#!/usr/bin/perl
$| = 1;
while (<STDIN>) {
     s|^/home/|/u/|;
     print $_;
}
```

In this example, the external expression function used to invoke `rewrite.pl` is as follows:

```
NameTrans fn="rewrite" path="$(external('rewrite.pl', $path))"
```

A.3.6.6 httpdate

The `httpdate` function returns an RFC 1123 date time stamp for use in HTTP header fields such as `Expires`.

Syntax

```
httpdate (time)
```

Argument

The following table describes the argument for the `httpdate` function.

Argument	Description
time	The time value

Example

The following `obj.conf` code could be used to set an `Expires` header that indicates a response is not cached for more than one day:

```
ObjectType fn="set-variable"
           insert-srvhdrs="$(httpdate($time + 86400))"
```

A.3.6.7 lc

The `lc` function converts all the US ASCII characters in the string to lowercase and returns the result.

Syntax

```
lc(string)
```

Argument

The following table describes the argument for the `lc` function.

Argument	Description
string	The string the expression function converts to lowercase

Example

The following obj.conf code can be used to redirect clients, who erroneously used uppercase characters in the request URI to the equivalent lowercase URI:

```
<If code == 404 and not -e path and -e lc(path)>
Error fn="redirect" uri="$(lc($uri))"
</If>e($time + 86400))"
```

See Also: uc

A.3.6.8 length

The `length` function returns the length of its argument, that is, a number representing the length of the string.

Syntax

```
length (string)
```

Arguments

The following table describes the argument for the expression function.

Argument	Description
string	The string for which the expression function computes the length.

Example

The following obj.conf code can be used to send a **404 Not found** error to clients that request URIs longer than 255 bytes:

```
<If length($uri) > 255)>
PathCheck fn="deny-existence"
</If>
```

A.3.6.9 lookup

The `lookup` function inspects a text file for a name-value pair with name *name* and returns the corresponding value. The name-value pairs in the file are separated by white space.

If the file does not contain a name-value pair with the specified name, this function returns the value of *defaultvalue,* if specified, or returns an empty string.

Syntax

```
lookup(filename, name, defaultvalue)
```

Arguments

The following table describes the argument for the expression function.

Argument	Description
filename	Indicates the name of a text file that contains one name-value pair per line. *filename* can be an absolute path or a path relative to the server's `config` directory. Names and values are separated by white space. Lines beginning with a pound sign (#) are ignored.
name	The name of the name-value pair for which the function looks in the text file.
defaultvalue	The value returned by the function if *filename* exists but does not contain a name-value pair with a name matching the value of name. If *defaultvalue* is not specified, it defaults to an empty string.

Example

The following example shows a text file called urimap.conf that could be used with the lookup function to map shortcut URIs to URIs:

```
# This file contains URI mappings for Oracle Traffic Director.
# Lines beginning with # are treated as comments.
# All other lines consist of a shortcut URI, whitespace, and canonical URI.
/webserver /software/products/web_srvr/home_web_srvr.html
/solaris   /software/solaris/
/java      /software/java/
```

Using the previous sample text file, you could use the following lookup expression to implement shortcut URIs for commonly accessed resources:

```
<If lookup('urimap.conf', uri)>
NameTrans fn="redirect" url="$(lookup('urimap.conf', uri))"
</If>
```

A.3.6.10 lookupregex

The `lookupregex` function inspects a text file for a regular expression-value pair. This function takes a string as an input and matches it with the regular expression in each line. It returns the corresponding value only if it matches. If the file does not contain a match, this function returns the default value, if specified, or returns an empty string.

Syntax

```
lookupregex (filename,string,defaultvalue)
```

Arguments

The following table describes the argument for the expression function.

Argument	Description
filename	The filename is the name of a text file that contains one regular expression-value pair per line. filename can be an absolute path or a path relative to the server's configuration directory.
string	The string to match with every regular expression in the text file.
defaultvalue	The value returned by the function if the filename exists but does not contain a matching regular expression-value pair. If `defaultvalue` is not specified, it defaults to an empty string.

A.3.6.11 mtime

The `mtime` function returns the time of the last data modification for the specified file or directory.

Syntax

```
mtime(path)
```

Arguments

The following table describes the argument for the expression function.

Argument	Description
path	The absolute path to the directory or file name for which you are requesting the last data modification.

See Also: atime, ctime

A.3.6.12 owner

The `owner` function returns the owner of a file.

Syntax

```
owner(path)
```

Arguments

The following table describes the argument for the expression function.

Argument	Description
path	The absolute path to the directory or file name for which you are requesting the last data modification.

A.3.6.13 uc

The `uc` function converts all the US ASCII characters in string to uppercase and returns the result.

Syntax

```
uc(string)
```

Arguments

The following table describes the argument for the expression function.

Table A–1 uc Argument

Argument	Description
string	The string that the expression function converts to uppercase.

See Also: lc

A.3.6.14 unescape

The unescape function decodes the URI using util_uri_unescape, converting percent-encoded octets to their unencoded form, and returns the result.

Syntax

```
unescape(uri)
```

Arguments

The following table describes the argument for the expression function.

Table A–2 unescape Argument

Argument	Description
uri	The URI that the function converts.

See Also: escape

A.3.6.15 uuid

The `uuid` function returns a UUID as a string. No two calls to `uuid` return the same UUID. Because they are guaranteed to be unique, UUIDs are useful for constructing client-specific cookie values.

Syntax

```
uuid()
```

A.3.7 Regular Expressions

The `If` and `ElseIf` expressions can evaluate regular expressions using the equal sign and tilde `(=~)` and exclamation point and tilde `(!~)` regular expression matching operators. These regular expressions use the Perl-compatible syntax implemented by Perl-compatible Regular Expressions (PCRE).

By default, regular expressions are case sensitive. The `(?i)` option flag can be added to the beginning of a regular expression to request case insensitivity, for example:

```
$uri =~ '^/[Ff][Ii][Ll][Ee][Nn][Aa][Mm][Ee]$'

$uri =~ '(?i)^/filename$'
```

When an `If` or `ElseIf` expression contains a regular expression, regular expression backreferences can appear within arguments in the container body. Regular expression backreferences are of the form `$`*n* where *n* is an integer between 1 and 9 corresponding to the capturing subpattern of the regular expression, for example:

```
<If $path =~ '^(.*)(\.html|\.htm)$'>
NameTrans fn="rewrite" path="$1.shtml"
</If>
```

In the preview example, two subpatterns are used in the `If` expression, so `$1` and `$2` can be used as backreferences. In the example, the value of the first capturing subpattern is used within a `NameTrans fn="rewrite"` parameter. The value of the second capturing subpattern is ignored.

An `If` or `ElseIf` expression can contain backreferences to earlier regular expressions in that same `If` or `ElseIf` expression, for example:

```
<If "foo" =~ "(.*)" and $1 eq "foo">
# Any contained directives will be executed
# since $1 will evaluate to "foo"
...
</If>
```

The contents of the preview `If` expression are executed, because the given `If` expression always evaluates to true.

However, `If` and `Elseif` expressions, and contained directives, cannot contain backreferences to regular expressions in parent containers. For example, the following `obj.conf` entry is invalid:

```
<If $path =~ '(.*)\.css'>
<If $browser = "*MSIE*">
# This example is invalid as $1 is not defined
AuthTrans fn="rewrite" path="$1-msie.css"
</If>
</If>
```

You can use the dollar sign and ampersand `$&` to obtain the value that last successfully matched a regular expression. Use the following `obj.conf` entry to redirect requests for HTML files to another server:

```
<If $path =~ '\.html$' or $path =~ '\.htm$' >
NameTrans fn="redirect" url="http://docs.example.com$&"
</If>
```

A.4 String Interpolation

Strings that contain references to variables or expressions are called interpolated strings. When you use interpolated strings, the embedded expressions and variables are evaluated, and the result is inserted into the string. The act of inserting data into a string is called string interpolation.

Use interpolated strings in expressions, log formats, and `obj.conf` parameters. In expressions, only string literals enclosed in double quotation marks are interpolated. For more information, see Expression Literals.

A.4.1 Using Variables in Interpolated Strings

To include the value of a variable in a string, prefix the name of the variable with the dollar-sign (`$`). For example, the following `format` element in `server.xml` logs the client IP address, requested URI, and corresponding file system path for each HTTP request:

```
<access-log>
  <file>access</file>
  <format>$ip "$uri" $path</format>
</access-log>
```

In this example, $ip, $uri, and $path are predefined variables. For more information, see Variables.

For more information about access logs and log format, see "Using the Custom Access-Log File Format".

If the name of the variable is ambiguous, add braces {} to the name. For example, the following string contains a reference to the predefined $path variable:

```
"${path}html"
```

Without the braces, the string contains a reference to a hypothetical variable named pathhtml.

A.4.2 Using Expressions in Interpolated Strings

To include the result of an expression in a string, prefix the expression with a dollar sign and a left parenthesis ($(and follow it with a right parenthesis). For example, the following two strings are identical after interpolation:

```
"$(2 + 2)"

"4"
```

When an interpolated string is used as an obj.conf parameter, the string is interpolated each time the corresponding instruction is executed. For example, the following lines could be used in the obj.conf file to redirect clients based on the requested URI and the contents of the file redirect.conf:

```
<Object ppath="/redirect/*">
NameTrans fn="redirect" url="$(lookup('redirect.conf', $uri, '/'))"
</Object>
```

In this example, the expression lookup('redirect.conf', $uri, '/') is evaluated each time the NameTrans directive is invoked, and the result is passed to the redirect SAF in its url parameter.

A.5 Wildcard Patterns

Oracle Traffic Director supports wildcard pattern matching in expressions. To use a wildcard without any special meaning, precede it with a backslash (\).

The following table describes various wildcard patterns and their uses.

Wildcard	Use
*	Matches zero or more characters.
?	Matches one occurrence of any character.
|	An or expression. The substrings used with this operator can contain other special characters such as an asterisk * or dollar sign $. The substrings must be enclosed in parentheses, for example, (a|b|c), but the parentheses cannot be nested.
$	Matches the end of the string. This is useful in or expressions.
[abc]	Matches one occurrence of the characters a, b, or c. Within these expressions, the only character that must be treated as a special character is the right bracket]; all others are not special.
[a-z]	Matches one occurrence of a character between a and z.

Wildcard	Use
`[^az]`	Matches any character except `a` or `z`.
`*~`	This expression, followed by another expression, removes any pattern matching the second expression.

The following table lists wildcard examples with pattern and the result.

Wildcard	Result
`*.example.com`	Matches any string ending with the characters `.example.com`.
`(quark|energy).example.com`	Matches either `quark.example.com` or `energy.example.com`.
`198.93.9[23].???`	Matches a numeric string starting with either `198.93.92` or `198.93.93` and ending with any 3 characters.
`*.*`	Matches any string with a period in it.
`*~example-*`	Matches any string except those starting with `example-`.
`*.example.com~quark.example.com`	Matches any host from domain `example.com` except for a single host `quark.example.com`.
`*.example.com~(quark|energy|neutrino).example.com`	Matches any host from domain `.example.com` except for hosts `quark.example.com`, `energy.example.com`, and `neutrino.example.com`.
`*.com~*.example.com`	Matches any host from domain `.com` except for hosts from sub domain `example.com`.
`*~*.gif*`	Matches any string except those including `gif`.

B

Using the Custom Access-Log File Format

This topic contains information about the log format used by Oracle Traffic Director. Use these format options to customize the format of your log files. You can enter them through the Admin Console, or edit the `format` subelement of the `access-log` element in the `server.xml` file.

You can use variables and expressions in log formats with the syntax `$`*variable* and `$(`*expression*`)`.

When creating a custom log format, anything between percent signs (%) is recognized as the name portion of a name-value pair stored in a parameter block in the server. Any additional text is treated as literal text, so you can add to the line to make it more readable. The one exception to the percent sign rule is the `%SYSDATE%` component, which delivers the current system date. `%SYSDATE%` is formatted using the time format `%d/%b/%Y:%H:%M:%S` and the offset from GMT.

If no format parameter is specified for a log file, the common log format is used:

```
"%Ses->client.ip% - %Req->vars.auth-user% [%SYSDATE%]
\"%Req->reqpb.clf-request%\" %Req->srvhdrs.clf-status%
%Req->srvhdrs.content-length%"
```

Typical components of log file format are listed in the following table. Because certain components could resolve to values that contain spaces, they are enclosed in escape quotes (`\"`).

Option	Component
Client host name (unless `iponly` is specified in `flex-log` or DNS name is not available) or IP address	`%Ses->client.ip%`
Client DNS name	`%Ses->client.dns%`
System date	`%SYSDATE%`
Full HTTP request line	`\"%Req->reqpb.clf-request%\"`
Status	`%Req->srvhdrs.clf-status%`
Response content length	`%Req->srvhdrs.content-length%`
Response content type	`%Req->srvhdrs.content-type%`
Referer header	`\"%Req->headers.referer%\"`
User-agent header	`\"%Req->headers.user-agent%\"`
HTTP method	`%Req->reqpb.method%`
HTTP URI	`%Req->reqpb.uri%`

Option	Component
HTTP query string	`%Req->reqpb.query%`
HTTP protocol version	`%Req->reqpb.protocol%`
Accept header	`%Req->headers.accept%`
Date header	`%Req->headers.date%`
If-Modified-Since header	`%Req->headers.if-modified-since%`
Authorization header	`%Req->headers.authorization%`
Any header value	`%Req->headers.headername%`
Name of authorized user	`%Req->vars.auth-user%`
Value of a cookie	`%Req->headers.cookie.name%`
Value of any variable in `Req->vars`	`%Req->vars.varname%`
Virtual server ID	`%vsid%`
ECID	`%Req->vars.ecid%`. For more information, refer the elements of http.
RID	`%Req->vars.rid%`. For more information, refer the elements of http.
Duration	`%duration%` Records the time in microseconds the server spent handling the request. Statistics must be enabled before `%duration%` can be used.
System time	`%Time` System time in seconds since 00:00:00 UTC, January 1, 1970.
Relative time	`%RELATIVETIME%` System time in seconds since logging started.
Method number	`%Req->method_num%` A number representing the HTTP method as used in NSAPI.
HTTP Protocol Version	`%Req->protv_num%` A number representing the HTTP protocol version as used in NSAPI.
HTTP request line	`%Req->reqpb.clf-request.method%` The method from the HTTP request line.
HTTP URI	`%Req->reqpb.clf-request.uri%` The URI from the HTTP request line.
URI path	`%Req->reqpb.clf-request.uri.abs_path%` The absolute path component of the URI
URI query	`%Req->reqpb.clf-request.uri.query%` The query component of the URI.
user_dn	`%Ses->client.user_dn%` The SSL client certificate authentication for web security.

Option	Component
HTTP protocol	`%Req->reqpb.clf-request.protocol%` The protocol from the HTTP request line.
Protocol name	`%Req->reqpb.clf-request.protocol.name%` The name of the protocol.
Protocol version	`%Req->reqpb.clf-request.protocol.version%` The version of the protocol.
Origin server	`%Req->vars.origin-server%` The origin server that served the request.

Additional log file parameters that can be configured are listed in the table below.

Option	Component
Cipher name	`%Ses->client.cipher%`
Size in bits of cipher key	`%Ses->client.keysize%`
Size in bits of private key	`%Ses->client.secret-keysize%`
DN for certificate issuer	`%Ses->client.issuer_dn%`
DN for certificate user	`%Ses->client.user_dn%`
SSL session identifier	`%Ses->client.ssl-id%`

C

Using Time Formats

This appendix describes the format strings used for dates and times in the server log. These formats are used by the NSAPI function `util_strftime`, by some built-in SAFs. The formats are similar to those used by the `strftime` C library routine, but not identical.

Table C–1 Format Strings for Date and Time

Attribute	Allowed Values
%a	Abbreviated day of the week (3 chars)
%d	Day of month as a decimal number (01-31)
%S	Second as a decimal number (00-59)
%M	Minute as a decimal number (00-59)
%H	Hour in 24-hour format (00-23)
%Y	Year with century, as a decimal number, up to 2099
%b	Abbreviated month name (3 chars)
%h	Abbreviated month name (3 chars)
%T	Time in HH:MM:SS format
%X	Time in HH:MM:SS format
%A	Day of the week, full name
%B	Month, full name
%C	"%a %b %e %H:%M:%S %Y"
%c	Date and time in "%m/%d/%y %H:%M:%S" format
%D	Date in "%m/%d/%y" format
%e	Day of month as decimal number (1-31) without leading zeros
%I	Hour in 12-hour format (01-12)
%j	Day of year as a decimal number (001-366)
%k	Hour in 24-hour format (0-23) without leading zeros
%l	Hour in 12-hour format (1-12) without leading zeros
%m	Month as a decimal number (01-12)
%n	Line feed
%p	a.m./p.m. indicator for 12-hour clock
%R	Time in "%H:%M" format

Table C–1 (Cont.) Format Strings for Date and Time

Attribute	Allowed Values
%r	Time in "%I:%M:%S %p" format
%t	Tab
%U	Week of year as a decimal number, with Sunday as first day of week (00-51)
%w	Weekday as a decimal number (0-6; Sunday is 0).
%W	Week of year as decimal number, with Monday as first day of week (00-51)
%x	Date in "%m/%d/%y" format
%y	Year without century, as decimal number (00-99)
%%	Percent sign

D

Alphabetical List of Server Configuration Elements and Predefined SAFs

This appendix provides an alphabetical list of server configuration elements, including `server.xml` elements and predefined SAFs in `obj.conf` file.

O

origin-server-pool

origin-server

P

pkcs11

pkcs11 bypass

property

proxy-cache

proxy-cache-config

proxy-cache-override-http

proxy-retrieve

Q

qos-error

qos-handler

R

remove-filter

reverse-map

rewrite

S

server

service-proxy-cache-dump

service-trace

set-cache-control

set-cookie

set-origin-server

snmp

ssl

ssl-client-config

ssl-logout

ssl-session-cache

stats

stats-xml

strip-params

T

thread-pool

time

token

type-by-exp