

Oracle® Fusion Middleware

Administering Oracle Traffic Director



12c (12.2.1.4.0)

E96373-02

November 2019

The Oracle logo, consisting of a solid red square with the word "ORACLE" in white, uppercase, sans-serif font centered within it.

ORACLE®

Oracle Fusion Middleware Administering Oracle Traffic Director, 12c (12.2.1.4.0)

E96373-02

Copyright © 2016, 2019, Oracle and/or its affiliates. All rights reserved.

Primary Author: Trupthi N. T.

Contributors: Isvaran Krishnamurthy, Amit Gupta, Savija Vijayaraghavan, Prem Kumar Venkatasalapathy, Praveen Chandrashekar

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, then 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

Preface

Audience	xiv
Documentation Accessibility	xiv
Related Documents	xiv
Conventions	xv
What's New in this Release?	xv

1 Introduction

Overview	1-1
Features of Oracle Traffic Director	1-2
Administration Interfaces	1-5
Command Line Interface-WebLogic Scripting Tool	1-5
Usage Modes	1-6
Graphical User Interface-Fusion Middleware Control	1-6
Oracle Traffic Director Terminology	1-6
Overview of Administration Tasks	1-8

2 Typical Deployment Topology

3 Setting up an Administration Domain

Types of Administration Domain	3-1
Creating a Managed Domain	3-3
Creating a Managed Domain Using Restricted JRF Template	3-4
Creating a Domain using Full JRF Template	3-6
Creating a Repository using Repository Creation Utility in Graphical Mode	3-9
Creating a Repository in Silent Mode	3-9
Logging to the Administration Console	3-10
Creating the Load Balancer for a Managed Domain	3-10
Creating a Standalone Domain	3-13
Creating a Standalone Domain using the Configuration Wizard	3-13

Creating a Standalone Domain Using Offline WLST Commands	3-14
Creating the Load Balancer for a Standalone Domain	3-14
Verifying the Load-Balancing Behavior of the Oracle Traffic Director Instance	3-16

4 Configuring Oracle Traffic Director for High Availability

Overview	4-1
Failover configuration modes	4-1
Failover in Active-Passive Mode	4-1
Failover in Active-Active Mode	4-3
Preparing your System for High Availability	4-4
Configuring High Availability	4-5

5 Managing Configurations

Creating an Oracle Traffic Director Configuration	5-1
Creating a Configuration Using Fusion Middleware Control	5-2
Creating a Configuration Using WLST	5-2
Viewing a List of Configurations	5-3
Viewing a List of Configurations Using Fusion Middleware Control	5-3
Viewing a List of Configurations Using WLST	5-3
Activating Configuration Changes	5-4
Activate Configuration Changes Using Fusion Middleware Control	5-4
Activate Configuration Changes Using WLST	5-4
Modifying an Oracle Traffic Director Configuration	5-5
Modifying a Configuration Using Fusion Middleware Control	5-5
Modifying a Configuration Using WLST	5-6
Copying an Oracle Traffic Director Configuration	5-8
Copying a Configuration Using Fusion Middleware Control	5-8
Copying a Configuration Using WLST	5-8
Deleting an Oracle Traffic Director Configuration	5-8
Deleting a Configuration Using Fusion Middleware Control	5-9
Deleting a Configuration Using WLST	5-10

6 Managing Instances

Creating Oracle Traffic Director Instances	6-1
Creating Oracle Traffic Director Instances Using Fusion Middleware Control	6-1
Creating Oracle Traffic Director Instance Using WLST	6-2
Viewing a List of Oracle Traffic Director Instances	6-2
Viewing a List of Oracle Traffic Director Instances Using Fusion Middleware Control	6-3

Viewing a List of Oracle Traffic Director Instances Using WLST	6-3
Starting, Stopping, and Restarting Oracle Traffic Director Instances	6-3
Starting, Stopping, and Restarting Oracle Traffic Director Instances Using Fusion Middleware Control	6-4
Starting, Stopping, and Restarting Oracle Traffic Director Instances Using WLST	6-4
Updating Oracle Traffic Director Instances Without Restarting	6-5
Reconfiguring an Oracle Traffic Director Instance Using Fusion Middleware Control	6-5
Reconfiguring Oracle Traffic Director Instances Using WLST	6-6
Deleting Oracle Traffic Director Instances	6-6
Deleting Oracle Traffic Director Instances Using Fusion Middleware Control	6-6
Deleting Oracle Traffic Director Instances Using WLST	6-7
Controlling Oracle Traffic Director Instances Through Scheduled Events	6-7
Managing Events Using Fusion Middleware Control	6-7
Managing Events Using WLST	6-8

7 Managing Origin-Server Pools

Creating an Origin-Server Pool	7-1
Creating an Origin-Server Pool Using Fusion Middleware Control	7-2
Creating an Origin-Server Pool Using WLST	7-3
Viewing a List of Origin-Server Pools	7-3
Viewing a List of Origin-Server Pools Using Fusion Middleware Control	7-4
Viewing a List of Origin-Server Pools Using WLST	7-4
Modifying an Origin-Server Pool	7-4
Changing the Properties of an Origin-Server Pool Using Fusion Middleware Control	7-5
Changing the Properties of an Origin-Server Pool Using WLST	7-7
Deleting an Origin-Server Pool	7-7
Deleting an Origin-Server Pool Using Fusion Middleware Control	7-8
Deleting an Origin-Server Pool Using WLST	7-9
Configuring an Oracle WebLogic Server Cluster as an Origin-Server Pool	7-9
How Dynamic Discovery Works	7-9
Enabling Dynamic Discovery	7-10
Configuring a Custom Maintenance Page	7-12
Configuring Health-Check Settings for Origin-Server Pools	7-13
Configuring Health-Check Settings for Origin Servers Using the Fusion Middleware Control	7-14
Configuring Health-Check Settings for Origin Servers Using WLST	7-14
Using an External Health-Check Executable to Check the Health of a Server	7-15
Configuring Health-Check Settings to Use an External Executable	7-15
Parameters to the External Health Check Executable	7-16

8 Managing Origin Servers

Adding an Origin Server to a Pool	8-1
Adding an Origin Server to a Pool Using Fusion Middleware Control	8-2
Adding an Origin Server to a Pool Using WLST	8-3
Viewing a List of Origin Servers	8-3
Viewing a List of Origin Servers Using Fusion Middleware Control	8-3
Viewing a List of Origin Servers Using WLST	8-4
Modifying an Origin Server	8-4
Modifying an Origin Server Using Fusion Middleware Control	8-4
Changing the Properties of an Origin Server Using WLST	8-5
Managing Ephemeral Ports	8-6
Removing an Origin Server from a Pool	8-6
Removing an Origin Server from a Pool Using Fusion Middleware Control	8-7
Removing an Origin Server from a Pool Using WLST	8-7

9 Managing Virtual Servers

Creating Virtual Servers	9-1
Creating a Virtual Server Using Fusion Middleware Control	9-2
Creating a Virtual Server Using WLST	9-3
Viewing a List of Virtual Servers	9-3
Viewing a List of Virtual Servers Using Fusion Middleware Control	9-3
Viewing a List of Virtual Servers Using WLST	9-4
Modifying Virtual Server Settings	9-4
Modifying a Virtual Server Using Fusion Middleware Control	9-4
Modifying a Virtual Server Using WLST	9-6
Configuring Routes for a Virtual Server	9-7
Configuring Routes Using Fusion Middleware Control	9-8
Configuring Routes Using WLST	9-9
Copying a Virtual Server	9-12
Copying a Virtual Server Using Fusion Middleware Control	9-12
Copying a Virtual Server Using WLST	9-13
Deleting a Virtual Server	9-13
Deleting a Virtual Server Using Fusion Middleware Control	9-13
Deleting a Virtual Server Using WLST	9-14
Caching in Oracle Traffic Director	9-14
Reviewing Cache Settings and Metrics for an Instance	9-14
Tunable Caching Parameters	9-16

Configuring Caching Parameters	9-17
Content Serving	9-20
Content Serving Using Fusion Middleware Control	9-21
Configuring Content Serving Using WLST	9-22

10 Managing TCP Proxies

Creating a TCP Proxy	10-1
Creating a TCP Proxy Using Fusion Middleware Control	10-2
Creating a TCP Proxy Using WLST	10-3
Viewing a List of TCP Proxies	10-3
Viewing a List of TCP Proxies Using Fusion Middleware Control	10-3
Viewing a List of TCP Proxies Using WLST	10-4
Modifying a TCP Proxy	10-4
Modifying a TCP Proxy Using Fusion Middleware Control	10-4
Modifying a TCP Proxy Using WLST	10-5
Deleting a TCP Proxy	10-6
Deleting a TCP Proxy Using Fusion Middleware Control	10-6
Deleting a TCP Proxy Using WLST	10-6

11 Managing Listeners

Creating a Listener	11-1
Creating a Listener Using Fusion Middleware Control	11-2
Creating a Listener Using WLST	11-3
Viewing a List of Listeners	11-4
Viewing a List of Listeners Using Fusion Middleware Control	11-4
Viewing a List of Listeners Using WLST	11-5
Modifying a Listener	11-5
Modifying a Listener Using Fusion Middleware Control	11-5
Modifying a Listener Using WLST	11-6
Deleting a Listener	11-7
Deleting a Listener Using Fusion Middleware Control	11-7
Deleting a Listener Using WLST	11-8
Configuring Status Listener	11-8
Configuring Status Listener using Fusion Middleware Control	11-8
Configuring Status Listener Using WLST	11-9

12 Managing Security

SSL/TLS Concepts	12-1
About SSL	12-1

About Ciphers	12-2
Cipher Suites Supported by Oracle Traffic Director	12-2
About Keys	12-3
About Certificates	12-4
RSA and ECC Certificates	12-4
Managing Certificates	12-4
Obtaining a Certificate	12-5
Generating a Keypair	12-6
Before You Begin	12-6
Generating a Keypair Using Fusion Middleware Control	12-6
Generating a Keypair using WLST	12-8
Generating a Certificate Signing Request (CSR)	12-9
Generating a CSR Using Fusion Middleware Control	12-9
Generating a CSR Using WLST	12-9
Importing a Certificate	12-10
Importing a CA Signed Certificate	12-10
Importing an Existing Certificate	12-11
Importing a Trusted Certificate	12-12
Viewing a List of Certificates	12-14
Deleting a Certificate	12-15
Deleting a Certificate using Fusion Middleware Control	12-15
Deleting a Certificate Using WLST	12-15
Configuring SSL/TLS on Oracle Traffic Director	12-16
Configuring SSL/TLS Between Oracle Traffic Director and Clients	12-16
Configuring SSL on a HTTP/TCP Listener	12-16
Configuring SSL On a Virtual Server	12-19
Configuring SSL/TLS Between Oracle Traffic Director and Origin Servers	12-21
About One-Way and Two-Way SSL/TLS	12-22
Configuring One-Way SSL/TLS Between Oracle Traffic Director and Origin Servers	12-22
Configuring Two-Way SSL/TLS	12-23
Configuring Ciphers On an Origin Server Pool	12-24
Configure SSL Termination At a Hardware Load Balancer front-ending Oracle Traffic Director	12-25
Configure WebLogic to receive SSL information from Web Tier / Traffic Director	12-27
Configure SSL Pass through on Oracle Traffic Director	12-28
Managing Certificate Revocation Lists	12-28
Installing and Deleting CRLs Manually	12-28
Installing CRLs Manually Using Fusion Middleware Control	12-28
Installing and Deleting CRLs Manually Using WLST	12-29
Update CRLs Automatically	12-29

Configuring Oracle Traffic Director to Install CRLs Automatically Using Fusion Middleware Control	12-30
Configuring Oracle Traffic Director to Install CRLs Automatically Using WLST	12-31

13 Managing Logs

About the Oracle Traffic Director Logs	13-1
Access Log	13-1
Server Log	13-2
Viewing Logs	13-2
Viewing Logs Using Fusion Middleware Control	13-3
Viewing Logs Using WLST	13-3
Configuring Log Preferences	13-3
Configuring Log Preferences Using Fusion Middleware Control	13-4
Configuring Log Preferences Using WLST	13-4
About Log Rotation	13-7
Rotating Logs Manually	13-7
Rotating Logs Manually Using Fusion Middleware Control	13-7
Rotating Logs Manually Using WLST	13-8
Configuring Oracle Traffic Director to Rotate Logs Automatically	13-9
Creating Log-Rotation Events Using Fusion Middleware Control	13-9
Creating Log-Rotation Events Using WLST	13-10

14 Managing Event Notifications

Origin server status change event	14-1
Subscribing to origin server status event using Fusion Middleware Control	14-1
Subscribing to origin server status change event Using WLST	14-2
Notification format	14-3
JSON Schema	14-3
Example	14-5
Error handling	14-5
Request limit exceeded event	14-6
Subscribing to Request Limit Exceeded Event Using WLST	14-6
Notification format	14-7
JSON schema	14-8
Example	14-9

15 Managing Failover Groups

Creating Failover Groups	15-1
Creating Failover Groups Using Fusion Middleware Control	15-2
Creating Failover Groups Using WLST	15-2
Managing Failover Groups	15-3

16 Monitoring Oracle Traffic Director Instances

Methods for Monitoring Oracle Traffic Director Instances	16-1
Configuring Statistics-Collection Settings	16-2
Configuring URI Access to Statistics Reports	16-3
Viewing Statistics Using WLST	16-6
Viewing stats-xml and perfdump Reports Through a Browser	16-6
Monitoring Using SNMP	16-8
Configuring Oracle Traffic Director Instances for SNMP Support	16-8
Configuring the SNMP Subagent	16-9
SNMP v3 User configuration	16-10
Starting and Stopping the SNMP Subagent	16-12
Viewing Statistics Using snmpwalk	16-13
Monitoring Using DMS	16-16
Sample XML (stats-xml) Report	16-17
Sample Plain-Text (perfdump) Report	16-19

17 Tuning Oracle Traffic Director for Performance

General Tuning Guidelines	17-1
Tuning the File Descriptor Limit	17-2
Tuning the Thread Pool and Connection Queue	17-4
About Threads and Connections	17-4
Reviewing Thread Pool Metrics for an Instance	17-5
Reviewing Connection Queue Metrics for an Instance	17-6
Tuning the Thread Pool and Connection Queue Settings	17-7
Tuning HTTP Listener Settings	17-8
Tuning Keep-Alive Settings	17-9
About Keep-Alive Connections	17-9
Reviewing Keep-Alive Connection Settings and Metrics	17-10
Tuning Keep-Alive Settings	17-12
Changing Keep-Alive Settings Using Fusion Middleware Control	17-12
Changing Keep-Alive Settings Using WLST	17-12
Tuning HTTP Request and Response Limits	17-13
Tuning DNS Caching Settings	17-14

Viewing DNS Cache Settings and Metrics	17-14
Configuring DNS Cache Settings	17-15
Tuning SSL/TLS-Related Settings	17-16
SSL/TLS Session Caching	17-16
Configuring SSL/TLS Session Cache Settings Using Fusion Middleware Control	17-16
Configuring SSL/TLS Session Caching Settings Using WLST	17-17
Ciphers and Certificate Keys	17-17
Configuring Access-Log Buffer Settings	17-18
Enabling and Configuring Content Compression	17-19
Tuning Connections to Origin Servers	17-23
Solaris-specific Tuning	17-25
Files Open in a Single Process (File Descriptor Limits)	17-26
Failure to Connect to HTTP Server	17-26
Tuning TCP Buffering	17-27
Reduce File System Maintenance	17-27
Long Service Times on Busy Volumes or Disks	17-27
Short-Term System Monitoring	17-27
Long-Term System Monitoring	17-28
Tuning for Performance Benchmarking	17-28

18 Diagnosing and Troubleshooting Problems

Roadmap for Troubleshooting Oracle Traffic Director	18-1
Troubleshooting High Availability Configuration Issues	18-2
Solutions to Common Errors	18-2
Startup failure: could not bind to port	18-2
Unable to start server with HTTP listener port 80	18-3
Oracle Traffic Director consumes excessive memory at startup	18-4
Operating system error: Too many open files in system	18-4
Oracle Traffic Director does not maintain session stickiness	18-4
Frequently Asked Questions	18-5
What is a "configuration"?	18-6
How do I access Fusion Middleware Control?	18-6
Why do I see a certificate warning when I access Fusion Middleware Control for the first time?	18-6
Can I manually edit configuration files?	18-6
In Fusion Middleware Control, what is the difference between saving a configuration and deploying it?	18-6
Why is the "Deployment Pending" message displayed in Fusion Middleware Control?	18-6
Why is the "Instance Configuration Deployed" message is displayed in Fusion Middleware Control?	18-7

Why does Fusion Middleware Control session end abruptly?	18-7
How do I access the WLST?	18-7
Why is a certificate warning message displayed when I tried to access the WLST for the first time?	18-7
How do I find out the short names for the options of a WLST command?	18-7
Why am I unable to select TCP as the health-check protocol when dynamic discovery is enabled?	18-7
After I changed the origin servers in a pool to Oracle WebLogic Servers, they are not discovered automatically, though dynamic discovery is enabled. Why?	18-8
How do I view the request and response headers sent and received by Oracle Traffic Director?	18-8
How do I enable SSL/TLS for an Oracle Traffic Director instance?	18-9
How do I find out which SSL/TLS cipher suites are supported and enabled?	18-10
How do I view a list of installed certificates?	18-10
How do I issue test requests to an SSL/TLS-enabled Oracle Traffic Director instance?	18-10
How do I analyze SSL/TLS connections?	18-10
How do I view details of SSL/TLS communication between Oracle Traffic Director instances and Oracle WebLogic Server origin servers?	18-13
Why are certain SSL/TLS-enabled origin servers marked offline after health checks, even though the servers are up?	18-13
Does Oracle Traffic Director rewrite the source IP address of clients before forwarding requests to the origin servers?	18-14
Why does Oracle Traffic Director return a 405 status code?	18-14
Contacting Oracle for Support	18-14

A Metrics Tracked by Oracle Traffic Director

Instance Metrics	A-1
Process Metrics	A-3
Connection Queue Metrics	A-4
Thread Pool Metrics	A-5
DNS Cache Metrics	A-6
Keep-Alive Metrics	A-6
Thread Metrics	A-7
Compression and Decompression Metrics	A-8
Virtual Server Metrics	A-8
CPU Metrics	A-10
Origin Server Metrics	A-11
Failover Instance Metrics	A-12
Cache Metrics	A-13
DMS Metrics Tables	A-13

B Web Application Firewall Examples and Use Cases

Basics of Rules	B-1
Rules Against Major Attacks	B-2
Brute Force Attacks	B-2
SQL Injection	B-4
XSS Attacks	B-5

Preface

This guide provides an overview of Oracle Traffic Director, and describes how to create, administer, monitor, and troubleshoot Oracle Traffic Director instances.

Audience

This guide is intended for users who are responsible for installing, configuring, administering, monitoring, and troubleshooting web-tier components such as web servers, reverse proxy servers, and load balancers.

It is assumed that readers of this guide are familiar with the following:

- Using web browsers
- Working in a terminal window
- Executing operating system commands on UNIX-like platforms

In addition, a basic understanding HTTP and SSL/TSL protocols is desirable, though not mandatory.

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 Release Notes*
- *Oracle Traffic Director Installation Guide*
- *Oracle Traffic Director WebLogic Server Scripting Tool Reference*
- *Oracle Traffic Director Configuration Files Reference*

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.
<i>italic</i>	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.

What's New in this Release?

This preface introduces the new and changed features of Oracle Traffic Director and other significant changes that are described in this guide, and provides pointers to additional information:

- **Weblogic Management Framework**

Oracle Traffic Director now supports database-based installation, or Restricted JRF-based installation. See Prerequisites for installing Oracle Traffic Director. Oracle Traffic Director introduces the WebLogic Management Framework, a set of tools that leverage Oracle WebLogic interfaces to provide a simple, consistent and distributed framework for managing Oracle. See What is the Weblogic Management Framework.

- **WebLogic Scripting Tool Commands**

Oracle Traffic Director now supports a command line interface, Weblogic Scripting Tool (WLST). WLST commands are used to configure and administer Oracle Traffic Director.

There are two types of WLST commands:

- Online Command
- Offline Command

The online commands requires connection to the WebLogic Server. The offline commands does not require connection to the WebLogic Server. Some commands are both online and offline, and can be invoked in both ways. All the commands configured through WLST should be activated.

See *WebLogic Scripting Tool Command Reference for Oracle Traffic Director*.

- **Multi-tenancy Support**

Oracle WebLogic Server provides a sharable infrastructure for multiple organizations. It allows one domain to support multiple tenants at a time, where a dedicated domain is not required.

Multi-tenancy provides resource isolation within a domain partition, an administrative and runtime partition of a WebLogic domain dedicated to running application instances and related resources for a tenant. In a typical deployment scenario, WebLogic Server is front-ended by Oracle Traffic Director. When

partition tasks are performed from WebLogic Server, Oracle Traffic Director must be configured appropriately to successfully front- end the Web Logic Partition.

With the Multi-tenancy support, Oracle Traffic Director is configured automatically without user action. See

See *Configuring Oracle Traffic Director and Life Cycle Management*.

See *Oracle Fusion Middleware Using WebLogic Server Multitenant*

- **Queueing with Request Limiting**

The requests that overflow are queued and are de-queued as per request priority. The request and response bandwidth can be controlled by this feature.

See [Tuning the Thread Pool and Connection Queue](#).

- **Origin Server Traffic Control**

The user can set limit on reusing same origin server connection for multiple requests by the keep-alive option. Bandwidth can be limited and controlled for each origin server. This feature can be used to control the HTTP origin server pools, but, not the TCP origin server pools.

See [Tuning Keep-Alive Settings](#).

- **Prioritized Backend Connection Management**

Oracle Traffic Director supports the prioritized requests to the back end server, for critical application requests. With this feature, requests with higher priority are de-queued before the ones with lower priority.

- **ModSecurity Upgrade**

Web Application Firewall support based on ModSecurity version 2.6.7 has been upgraded to 2.8.0 .

- **Event Notifications**

Oracle Traffic Director supports sending notifications to one or more user configured HTTP endpoints for the following two events:

- Origin Server Status Change event
- Request Limit Exceeded event

See [Event Notifications](#).

- **High Availability Using Active-Active Failover Mode**

Oracle Traffic Director provides support for failover between the instances by deploying two or more OTD instances on the nodes which are in the same subnet. One of the nodes is chosen as the active router node and the remaining node(s) are the backup router node(s).The traffic will be managed among all the OTD instances.

See [Configuring Oracle Traffic Director for High Availability](#).

- **Status Listeners to check Oracle Traffic Director instances**

Oracle Traffic Director supports configuring a dedicated status listener to check the health of an Oracle Traffic Director instance.

See [Configuring Status Listener](#).

- **Support for Front-ending FTP Traffic**

Oracle Traffic Director provides options to enable FTP support on a TCP proxy. You can enable client FTP and server FTP settings on a TCP proxy.

See [Managing TCP Proxies](#).

1

Introduction

Learn about the features of Oracle Traffic Director and also understand the basic terminology and the administration tasks.

Note:

As of 12.2.1.4.0, Oracle Traffic Director is deprecated. In the future, use Oracle HTTP Server, Microsoft IIS Web Server, or Apache HTTP Server plug-ins, or a native Kubernetes load balancer, such as Traefik for equivalent functionality.

This chapter includes the following sections:

- [Overview](#)
- [Features of Oracle Traffic Director](#)
- [Administration Interfaces](#)
- [Oracle Traffic Director Terminology](#)
- [Overview of Administration Tasks](#)

Overview

Oracle Traffic Director distributes the requests that it receives from clients to servers in the back end based on the specified load-balancing method, routes the requests based on specified rules, caches frequently accessed data, prioritizes traffic, and controls the quality of service.

Oracle Traffic Director is a fast, reliable, and scalable layer-7 software load balancer. You can set up Oracle Traffic Director to serve as the reliable entry point for all HTTP, HTTPS, FTP and TCP traffic to application servers and web servers in the back end. Depending on the needs of your IT environment, you can configure Oracle Traffic Director to apply multiple, complex rules when distributing requests to the back-end servers and when forwarding responses to clients.

The architecture of Oracle Traffic Director enables it to handle large volumes of application traffic with low latency. The product is optimized for use in Oracle Exalogic Elastic Cloud and Oracle SuperCluster. It can communicate with servers in the back end over Exalogic's InfiniBand fabric using protocols such as Socket Direct Protocol (SDP) for better throughput. For more information about Exalogic, see the Oracle Exalogic Elastic Cloud documentation, http://docs.oracle.com/cd/E18476_01/index.htm. Oracle Traffic Director is also certified with various Fusion Middleware products.

Features of Oracle Traffic Director

Oracle Traffic Director 12c (12.2.1) includes new features in high availability, websocket connections support, integration with Oracle Fusion Middleware, enhanced security and performance, and more. This document describes the new features made in the initial release of 12c (12.2.1), and also describes the changes made in the subsequent patch set releases: 12.2.1.1.0, 12.2.1.2.0, and 12.2.1.3.0.

On engineered system platforms, you can set up pairs of Oracle Traffic Director instances and leverage its built-in High Availability capability to setup either Active-Passive or Active-Active failover with additional back-end servers to which it can route the requests. These back-end servers can also be added dynamically by the origin-server. For example, in the WebLogic cluster, the Oracle Traffic Director auto detects such changes using Dynamic Node Discovery method. As the volume of traffic to your network grows, you can easily scale the environment by reconfiguring Oracle Traffic Director with additional back-end servers to which it can route requests.

Oracle Traffic Director provides the following features:

- **Advanced methods for load distribution**

Configure Oracle Traffic Director to distribute client requests to servers in the back-end using one of these methods:

 - Round robin
 - Least connection count
 - Least response time
 - Weighted round robin
 - Weighted least connection count
 - IP Hash
- **Flexible routing and load control on back-end servers**
 - **Request-based routing**

Oracle Traffic Director can be configured to route HTTP/S requests to specific servers in the back end based on information in the request URI: pattern, query string, domain, source and destination IP addresses, and so on.
 - **Content-based routing**

Oracle Traffic Director can be configured to route HTTP/S requests to specific servers in the back end based on contents within a request. This way, web service requests such as XML or JSON can be easily routed to specific origin servers based on specific elements within the body content. Content-based routing is enabled by default.
 - **Request rate acceleration**

Administrators can configure the rate at which Oracle Traffic Director increases the load (number of requests) for specific servers in the back end. By using this feature, administrators can allow a server that has just been added to the pool, or has restarted, to perform startup tasks such as loading data and allocating system resources.
 - **Connection limiting**

Oracle Traffic Director can be configured to limit the number of concurrent connections to a back end origin-server. When the configured connection limit for a server is reached, further requests that require new connections are not sent to that server.

- **Controlling the request load and quality of service**

- **Request rate limiting**

Oracle Traffic Director can be set up to limit the rate of incoming requests from specific clients and for specific types of requests. This feature enables administrators to optimize the utilization of the available bandwidth, guarantee a certain level of quality of service, and prevent denial-of-service (DoS) attacks.

- **Quality of service tuning**

To ensure equitable utilization of the available network resources for incoming requests, you can configure Oracle Traffic Director virtual servers to limit the maximum number of concurrent connections to clients and the maximum speed at which data can be transferred to clients.

- **Support for WebSocket connections**

Oracle Traffic Director handles WebSocket connections by default. WebSocket connections are long-lived and allow support for live content, games in real-time, video chatting, and so on. In addition, Oracle Traffic Director can be configured to ensure that only those clients that strictly adhere to RFC 6455 are allowed. See [Configuring Routes for a Virtual Server](#) and the *Oracle Traffic Director Command-Line Reference*.

- **Integration with Oracle Fusion Middleware**

- Oracle Traffic Director is designed to recognize and handle headers that are part of requests to, and responses from, Oracle WebLogic Server managed servers in the back end.
 - When an Oracle Traffic Director instance is configured to distribute client requests to clustered Oracle WebLogic Server managed servers, Oracle Traffic Director automatically detects changes in the cluster—such as the removal or addition of managed servers, and considers such changes while routing requests.
 - Patches that Oracle delivers for the Oracle Traffic Director software can be applied by using OPatch, a Java-based utility, which is the standard method for applying patches to Oracle Fusion Middleware products.

- **Easy-to-use administration interfaces**

Administrators can use either a graphical user interface or a command-line interface to administer Oracle Traffic Director instances.

Administrators can also use Fusion Middleware Control, a browser-based graphical user interface, to monitor statistics and perform lifecycle tasks for Oracle Traffic Director instances.

- **Security**

Oracle Traffic Director enables and enhances security for your IT infrastructure in the following ways:

- **Reverse proxy**

By serving as an intermediary between clients outside the network and servers in the back end, Oracle Traffic Director masks the names of servers in the back end and provides a single point at which you can track access to critical data and applications hosted by multiple servers in the back end.

- **Support for TLS 1.1, and 1.2**

To secure data during transmission and to ensure that only authorized users access the servers in the back end, you can configure TLS-enabled HTTP and TCP listeners for Oracle Traffic Director instances.

You can either use digital certificates issued by commercial CAs such as VeriSign or generate RSA- and Elliptic Curve Cryptography (ECC)-type self-signed certificates with key sizes of up to 4096 bits by using the administration console or the WLST.

- **Web Application Firewall**

A Web application firewall enables you to apply a set of rules to an HTTP request, which are useful for preventing common attacks such as Cross-site Scripting (XSS) and SQL Injection. The Web Application Firewall module for Oracle Traffic Director supports open source ModSecurity 2.8.0.

- **HTTP Forward Proxy Support in Origin Server Pools**

In an environment where access to intended origin servers is restricted through corporate proxy servers, you can optionally associate an HTTP forward proxy server with an origin server pool so that all of its member origin servers (of said pool) are communicated with via the configured HTTP forward proxy server.

- **High availability**

On engineered systems platforms, you can set up pairs of Oracle Traffic Director instances and leverage its built-in High Availability capability to setup either Active-Passive or Active-Active failover. As the volume of traffic to your network grows, you can easily scale the environment by reconfiguring Oracle Traffic Director with additional back-end servers to which it can route requests.

Oracle Traffic Director provides high availability for your enterprise applications and services through the following mechanisms:

- **Health checks for the back end**

If a server in the back end is no longer available or is fully loaded, Oracle Traffic Director detects this situation automatically through periodic health checks and stops sending client requests to that server. When the failed server becomes available again, Oracle Traffic Director detects this automatically and resumes sending requests to the server.

- **Backup servers in the back end**

When setting up server pools for an Oracle Traffic Director instance, you can designate a few servers in the back end as backup servers. Oracle Traffic Director sends requests to the backup servers only when none of the primary servers are available. This feature ensures continued availability even when some servers in the back end fail.

- **Failover for load balancing**

Oracle Traffic Director instances can be deployed in an active-passive or active-active configuration. If the primary Oracle Traffic Director instance fails, the backup instance takes over.

- **Dynamic reconfiguration**

Most configuration changes to Oracle Traffic Director instances can be deployed dynamically, without restarting the instances and without affecting requests that are being processed.
- **Monitoring statistics**

Administrators can monitor a wide range of statistics pertaining to the performance of Oracle Traffic Director instances through several methods: the administration console, the command-line interface, and a report in XML format.
- **High performance**
 - **SSL/TLS offloading**

Oracle Traffic Director can be configured as the SSL/TLS termination point for HTTP/S and TCP requests. This reduces the processing of overhead on the servers in the back end.
 - **Content caching**

Oracle Traffic Director can be configured to cache (in its process memory) content that it receives from origin servers. By caching content, Oracle Traffic Director helps reduce the load on servers in the back end and helps improve performance for clients.
 - **HTTP compression**

Administrators can configure Oracle Traffic Director instances to compress the data received from servers in the back end and forward the compressed content to the requesting clients. This feature improves the response time for clients connected on slow connections.

Administration Interfaces

Oracle Traffic Director includes a robust command-line interface-WebLogic Scripting Tool as well as a simple, wizard-driven graphical interface-Fusion Middleware Control to help you administer Oracle Traffic Director instances. You can create, modify, and manage Oracle Traffic Director instances.

Command Line Interface-WebLogic Scripting Tool

The command line interface in Oracle Traffic Director is the WebLogic Scripting Tool (WLST).

The WLST scripting environment is based on Jython which is an implementation of the Python language for the Java platform. The tool can be used both online and offline. Oracle Traffic Director ships custom WLST commands that you can run using WLST.

Note:

Oracle Traffic Director ships a `wlst.sh` wrapper `<oracle_home>/otd/common/bin/wlst.sh` which initializes the required environment and libraries for Oracle Traffic Director commands. All Oracle Traffic Director custom commands can only be executed from this `wlst.sh`.

See *WebLogic Scripting Tool Command Reference for Oracle Traffic Director*.

Usage Modes

You can use the following techniques to invoke Oracle Traffic Director custom commands.

- Interactive Mode
- Script Mode
- Embedded Mode

Graphical User Interface-Fusion Middleware Control

To display Fusion Middleware Control, you enter the Fusion Middleware Control URL, which includes the name of the host and the administration port number assigned during the installation. The following shows the format of the URL:

```
http://hostname.domain:port/em
```

The port number is the port number of the Fusion Middleware Control. By default, the port number is 7001. The port number is listed in the following file:

```
DOMAIN_HOME/config/config.xml
```

For some installation types, such as Web Tier, if you saved the installation information by clicking Save on the last installation screen, the URL for Fusion Middleware Control is included in the file that is written to disk (by default to your home directory). For other installation types, the information is displayed on the Create Domain screen of the Configuration Wizard when the configuration completes.

To display Fusion Middleware Control:

1. Enter the URL in your Web browser. For example:

```
http://host1.example.com:7001/em
```

2. Enter the Fusion Middleware Control administrator user name and password and click **Login**.

You can now create Oracle Traffic Director configurations and deploy them as instances on Node Managers.

Oracle Traffic Director Terminology

Learn about the terms used in this document when describing administrative tasks for Oracle Traffic Director.

The following table describes the Oracle Traffic Director terms used throughout this document.

Term	Description
Configuration	<p>A collection of configurable elements (metadata) that determine the run-time behavior of an Oracle Traffic Director instance.</p> <p>A typical configuration contains definitions for the listeners (IP address and port combinations) on which Oracle Traffic Director should listen for requests and information about the servers in the back end to which the requests should be sent. Oracle Traffic Director reads the configuration when an Oracle Traffic Director instance starts and while processing client requests.</p>
Instance	<p>An Oracle Traffic Director server that is instantiated from a configuration and deployed on a Node Manager.</p>
Failover group	<p>Two Oracle Traffic Director instances grouped by a virtual IP address (VIP), to provide high availability in active-active or active-passive mode. Requests are received at the VIP and routed to the Oracle Traffic Director instance that is designated as the primary instance. If the primary instance is not reachable, requests are routed to the backup instance.</p> <p>For active-active failover, two failover groups are required, each with a unique VIP, but both consisting of the same nodes with the primary and backup roles reversed. Each instance in the failover group is designated as the primary instance for one VIP and the backup for the other VIP.</p>
ORACLE_HOME	<p>A directory of your choice in which you install the Oracle Traffic Director binaries.</p>
DOMAIN_HOME	<p>A path to the directory which contains Oracle Traffic Director domain,</p>
Fusion Middleware Control	<p>A web-based graphical interface on the server that you can use to create, deploy, and manage Oracle Traffic Director instances.</p>
Client	<p>Any agent—a browser or an application, for example—that sends HTTP, HTTPS and TCP requests to Oracle Traffic Director instances.</p>
Origin server	<p>A server in the back end, to which Oracle Traffic Director forwards the HTTP, HTTPS and TCP requests that it receives from clients, and from which it receives responses to client requests.</p> <p>Origin servers can be application servers like Oracle WebLogic Server managed servers, web servers, and so on.</p>
Origin-server pool	<p>A collection of origin servers that host the same application or service that you can load-balance by using Oracle Traffic Director.</p> <p>Oracle Traffic Director distributes client requests to servers in the origin-server pool based on the load-distribution method that is specified for the pool.</p>
Virtual server	<p>A virtual entity within an Oracle Traffic Director server instance that provides a unique IP address (or host name) and port combination through which Oracle Traffic Director can serve requests for one or more domains.</p> <p>An Oracle Traffic Director instance on a node can contain multiple virtual servers. Administrators can configure settings such as the maximum number of incoming connections specifically for each virtual server. They can also customize how each virtual server handles requests.</p>

Overview of Administration Tasks

An Oracle Traffic Director administrator can install the product, configure domain, manage instances, and so on. The tools to manage are WebLogic Scripting Tool(WLST) and Fusion Middleware Control.

Prerequisites

- Install the product.
You can install Oracle Traffic Director on Oracle Linux 6.5+ on an x86_64 system, by using an interactive graphical wizard or in silent mode. Note that in 12c, Oracle Traffic Director does not have its own separate Administration Server, but uses the Administration Server in Oracle WebLogic Server.
See [Installing Oracle Traffic Director](#).
- Create a WebLogic domain for Oracle Traffic Director. See [Creating Domains Using the Pack and Unpack Commands](#) or [Creating WebLogic Domains Using the Configuration Wizard](#).
- Access Fusion Middleware Control and WLST
You can use Fusion Middleware Control and command-line interface of Oracle Traffic Director to create, modify, and monitor Oracle Traffic Director configurations.
For information about accessing Fusion Middleware Control and command-line interface, see [Administration Interfaces](#).

The standard tasks include:

- Create and manage configurations
Create configurations that define your Oracle Traffic Director instances. A configuration is a collection of metadata that you can use to instantiate Oracle Traffic Director. Oracle Traffic Director reads the configuration when a server instance starts and while processing client requests.
See [Managing Configurations](#).
- Create and manage instances
After creating a configuration, you can create Oracle Traffic Director server instances by deploying the configuration on one or more hosts. You can view the current state of each instance, start or stop it, reconfigure it to reflect configuration changes, and so on.
See [Managing Instances](#).
- Define and manage origin-server pools
For an Oracle Traffic Director instance to distribute client requests, you should define one or more origin-server pools or in the back end. For each origin-server pool, you can define the load-distribution method that Oracle Traffic Director should use to distribute requests. In addition, for each origin server in a pool, you can define how Oracle Traffic Director should control the request load.
See [Managing Origin Servers](#) and [Managing Origin-Server Pools](#).
- Create and manage virtual servers and listeners

An Oracle Traffic Director instance running on a node contains one or more virtual servers. Each virtual server provides one or more listeners for receiving requests from clients. For each virtual server, you can configure parameters such as the origin-server pool to which the virtual server should route requests, the quality of service settings, request limits, caching rules, and log preferences.

See [Managing Virtual Servers](#) and [Managing Listeners](#).

- Manage security

Oracle Traffic Director, by virtue of its external-facing position in a typical network, plays a critical role in protecting data and applications in the back end against attacks and unauthorized access from outside the network. In addition, the security and integrity of data traversing through Oracle Traffic Director to the rest of the network needs to be guaranteed.

See [Managing Security](#).

- Manage logs

Oracle Traffic Director records data about server events such as configuration changes, instances being started and stopped, errors while processing requests, and so on in log files. You can use the logs to troubleshoot errors and to tune the system for improved performance.

See [Managing Logs](#).

- Monitor statistics

The state and performance of Oracle Traffic Director instances are influenced by several factors: configuration settings, volume of incoming requests, health of origin servers, nature of data passing through the instances, and so on. As the administrator, you can view metrics for all of these factors through the command-line interface and Fusion Middleware Control, and extract the statistics in the form of XML files for detailed analysis. You can also adjust the granularity at which Oracle Traffic Director collects statistics.

See [Monitoring Oracle Traffic Director Instances](#).

- Set up Oracle Traffic Director instances for high availability

In the event that an Oracle Traffic Director instance or the node on which it runs fails, you need to ensure that the load-balancing service that the instance provides continues to be available uninterrupted. You can achieve this goal by configuring a backup Oracle Traffic Director instance that can take over processing of requests when the primary instance fails.

See [Setting up Failover](#).

- Tune for performance

Based on your analysis of performance statistics and to respond to changes in the request load profile, you might want to adjust the request processing parameters of Oracle Traffic Director to maintain or improve the performance. Oracle Traffic Director provides a range of performance-tuning controls and knobs that you can use to limit the size and volume of individual requests, control timeout settings, configure thread pool settings, SSL/TLS caching behavior, and so on.

See [Tuning Oracle Traffic Director for Performance](#).

- Diagnose and troubleshoot problems

Despite the best possible precautions, you might occasionally run into problems when installing, configuring, and monitoring Oracle Traffic Director instances. You

can diagnose and solve some of these problems based on the information available in error messages and logs. For complex problems, you would need to gather certain data that Oracle support personnel can use to understand, reproduce, and diagnose the problem.

See [Diagnosing and Troubleshooting Problems](#).

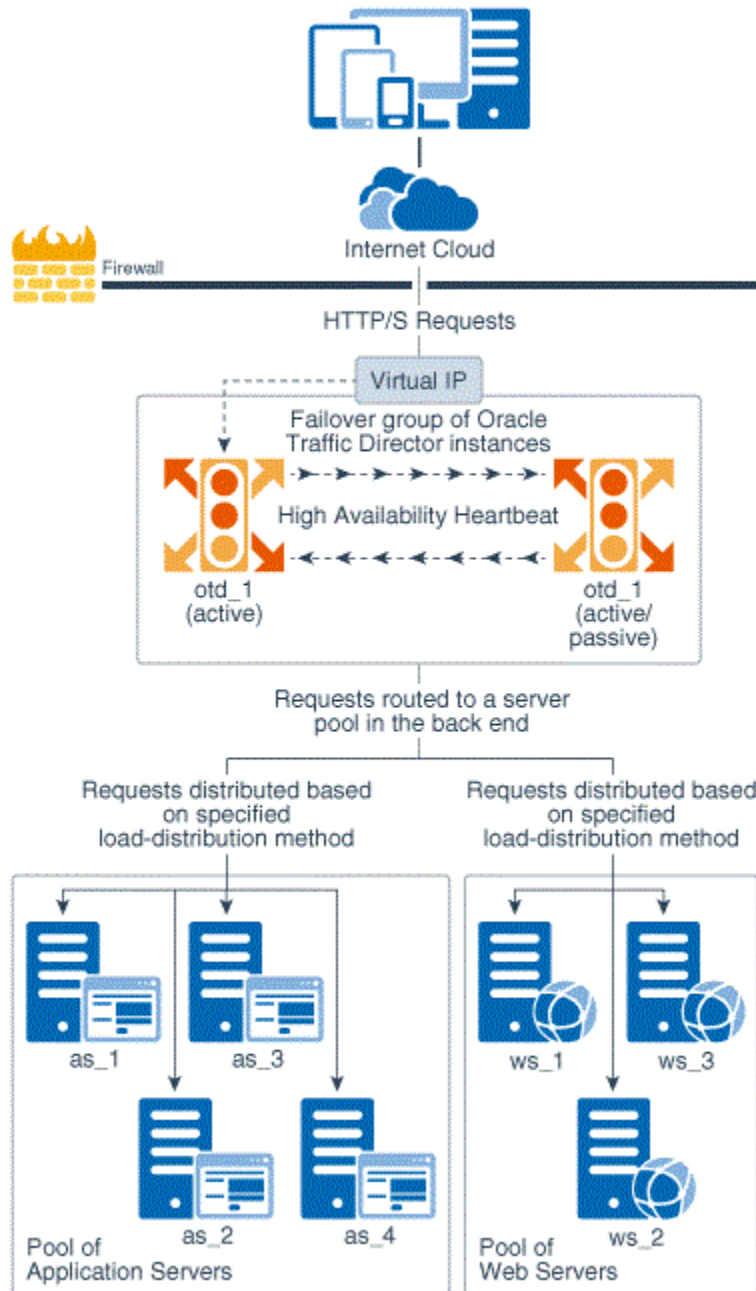
2

Typical Deployment Topology

In the simplest implementation, you can have a single Oracle Traffic Director instance running on a dedicated compute node distributing client requests to a pool of servers in the back end.

The topology that you create for Oracle Traffic Director varies depending on your business requirements such as the number of back-end applications for which you want to use Oracle Traffic Director to balance requests, IT requirements such as security, and the features of Oracle Traffic Director that you want to use.

To ensure that the node on which an Oracle Traffic Director instance runs does not become the single point of failure in the topology, you can have Oracle Traffic Director instances running on different nodes forming a *failover group* for high availability. The failover group is represented by virtual IP (VIP) addresses. Both the hosts in a failover group must run the same operating system version, use identical patches and service packs, and run Oracle Traffic Director instances of the same configuration.



The topology consists of two Oracle Traffic Director instances—`otd_1` and `otd_2`—forming a failover pair and providing a single virtual IP address for client requests. Based on the mode of failover configured, the primary node will determine how and where to forward the request. For information on failover modes, see [Failover configuration modes](#).

Note that the topology shows only two server pools in the back end, but you can configure Oracle Traffic Director to route requests to servers in multiple server pools.

3

Setting up an Administration Domain

Setting up an Oracle Traffic Director domain starts with creating an administration domain, either managed or standalone. It also includes creating a repository and creating a load balancer.

This chapter includes the following sections:

- [Types of Administration Domain](#)
- [Creating a Managed Domain](#)
- [Creating a Standalone Domain](#)
- [Verifying the Load-Balancing Behavior of the Oracle Traffic Director Instance](#)

Types of Administration Domain

You need to create an administration domain for managing Oracle Traffic Director instances. The administration domain can either be managed or standalone.

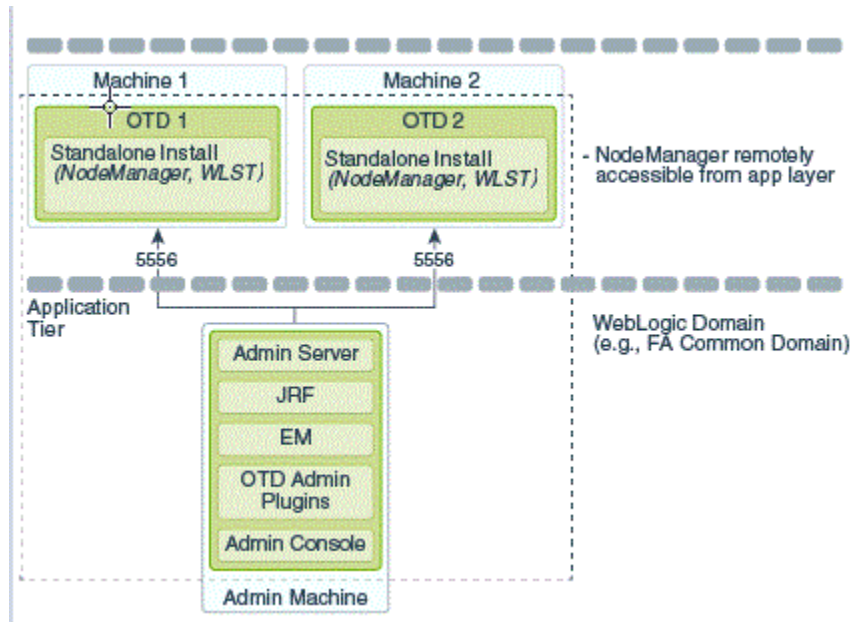
This section describes the various domain types supported in Oracle Traffic Director.

Managed Domain

A managed domain includes a special WebLogic Server instance, Administration Server, the central point from where all Oracle Traffic Director instances in the domain are configured and managed. This consists of:

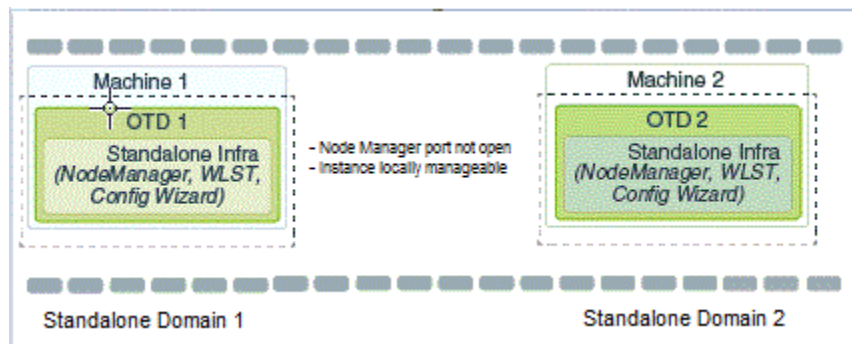
- A node where Oracle Traffic Director is installed collocated with WebLogic Administration Server within an Oracle Fusion Middleware Infrastructure Oracle home.
- Zero or more remote nodes managed by the administration server.

A managed domain allows you to use Fusion Middleware Control or custom WLST commands for administering Oracle Traffic Director.



Standalone Domain

This is a single node installation where Oracle Traffic Director is installed in standalone mode. There is no administration support for standalone domains and the custom WLST commands for administering Oracle Traffic Director cannot be used in a standalone domain.



Choosing Between Managed Domain and Standalone Domain

Table 3-1 Managed Domain vs Standalone Domain

Advantages/ Disadvantages	Managed Domain	Standalone Domain
Advantages	<ul style="list-style-type: none"> • Oracle Traffic Director Administration support including support for SSL and HA. • Farm wide deployment with minimum overhead. • Monitoring / Server Lifecycle on multiple machines through Node manager. 	<ul style="list-style-type: none"> • Sandbox Environment with complete Isolation. • No JRE or WebLogic Server dependency.
Disadvantages	<ul style="list-style-type: none"> • Requires Node Manager to communicate with Administration Server. • JRE and Node manager are needed for Oracle Traffic Director administration. These can be shutdown post provisioning. 	<ul style="list-style-type: none"> • No Oracle Traffic Director administration support other than Oracle Traffic Director instance creation/deletion. • Monitoring/Server Lifecycle management is limited to the local Oracle Traffic Director instance.

Note:

An Oracle Traffic Director instance can be deployed in a DMZ environment by making it a part of a managed Oracle Traffic Director domain. The administration server on a remote node can be used to manage the same through the Node manager. If you do not want to run the Node manager in DMZ environment, Oracle Traffic Director can be deployed in a standalone domain.

Creating a Managed Domain

After installing Oracle Traffic Director collocated with WebLogic Server, you can create an Oracle Traffic Director managed domain. The Oracle Traffic Director managed domain is created from an Oracle Traffic Director domain template. An Oracle Traffic Director domain can either be created using restricted JRF template or a full JRF template.

Restricted JRF is the recommended template for creating an Oracle Traffic Director domain. The restricted JRF mode creates a WebLogic Server runtime without a data source connection. Hence, there is no need for a running Oracle database to create the domain. You should use full JRF template if Oracle Traffic Director and SOA must be in the same domain.

Choosing Between Restricted JRF and Full JRF

Table 3-2 Restricted JRF vs Full JRF

Advantages/ Disadvantages	Restricted JRF Template	Full JRF Template
Advantages	<ul style="list-style-type: none"> Recommended mode of creating an Oracle Traffic Director domain. Does not need a data-source connection. 	<ul style="list-style-type: none"> Domain supports various components/scenarios such as CCWS, OPSS support for partitions, OPSS support for up-stack components of WebLogic and so on. ADF applications and Oracle Fusion Middleware products such as SOA and WebCenter depend on Full JRF Template. Supports cross-component wiring.
Disadvantages	<ul style="list-style-type: none"> You cannot migrate the domain to a full JRF domain later. ADF applications and Oracle Fusion Middleware products such as SOA and WebCenter does not support this template. Does not support cross-component wiring. 	<ul style="list-style-type: none"> You require an Oracle database to create a domain using full JRF template.

Topics

- [Creating a Domain Using Restricted JRF Template](#)
- [Creating a Domain using Full JRF Template](#)
- [Logging to the Administration Console](#)
- [Creating the Load Balancer for a Managed Domain](#)

Creating a Managed Domain Using Restricted JRF Template

The restricted JRF mode creates a WebLogic Server runtime without a data source connection. Hence, one does not need a running Oracle database to create the domain.

Follow these steps to create the Oracle Traffic Director domain using Restricted JRF Template:

- Run the configuration wizard using `config.sh` located in the `$FMW_HOME/oracle_common/common/bin` directory.
The Create Domain screen appears.
- Select the default values and click **Next**.
The Templates screen appears.
- Select **Create Domain Using Product Templates**. In the **Available Templates** section, choose **Oracle Traffic Director - Restricted JRF [otd]**.

Oracle Enterprise Manager - Restricted JRF, Oracle Restricted JRF and Weblogic Coherence Cluster Extension are automatically selected. Retain the selection.

4. Click **Next**.

The Application Location screen appears.

5. Retain the default value for Application location and click **Next**.

The Administrator Account screen appears. Enter the WebLogic Domain administration user name and password. This information is needed to access WebLogic Server Control and Fusion Middleware Control.

6. Click **Next**.

The Domain Mode and JDK screen appears.

7. Choose the default Domain Mode, **Development** and **Default** for JDK

. Click **Next**

The Advanced Configuration screen appears.

8. Select '*Admin Server*', '*Node Manager*', '*Topology*'.

 **Note:**

Select 'Managed Server' to configure a 'Machine'. The screen for Machine is visible only if 'Managed Servers ' option is selected.

9. Click **Next**.

The Administration Server screen appears. Enter the following:

- Server Name: Use default (*AdminServer*)
- Listen Address: Use default (*All local Addresses*)
- Listen Port: *\$WLS_ADMIN_PORT*
- Enable SSL: Use default (*unchecked*)
- SSL Port: Use default (*disabled*)
- Server Groups: Use default (*unspecified*)

10. Click **Next**.

The Node Manager screen appears. Use the default values (default: *Per Domain*). For Node Manager Credentials, enter the following:

- User Name: *\$NM_USER*
- Password: *\$NM_PASSWORD*
- Confirm password: *\$NM_PASSWORD*

11. Click **Next**.

The Managed Servers screen appears. Do not add any managed servers.

12. Click **Next**.

The Clusters screen appears. Do not add any clusters.

13. Click **Next**.

The Server Templates screen appears.

14. Click Next.

The Coherence Clusters screen appears. Do not add any clusters.

15. Click Next.

The Machines screen appears. Click **Add** and enter the following information:

- Name: `$MACHINE_NAME`
- Node Manager Listen Address: Use default (`localhost`)
- Node Manager Listen Port: `$NM_PORT`

16. Click Next.

The Assign Servers to Machines screen appears.

17. Select one or more servers in the left pane and one machine in the right pane. Then use the right arrow button (>) to assign the server or servers to the machine. Select 'Admin Server', 'new_ManagedServer_1','new_Machine_1'. Click Next.

The Virtual Targets screen appears. Do not add any virtual targets.

18. Click Next.

The Partitions screen appears.

19. Click Next.

The Configuration Summary screen appears. If the message *The Security configuration in your domain is invalid* appears, you may ignore it.

20. Click Create.

The Configuration Progress screen appears, which indicates the progress of the Configuration. After successful completion of the configuration process, the *Domain Created Successfully* message appears.

21. Click Next.

Wait for this part of the configuration to complete. Depending on the location and performance of the Repository database, this process may take a few minutes. Click **Finish**. The *End of Configuration* screen appears.

Creating a Domain using Full JRF Template

Follow these steps to create the Oracle Traffic Director domain using the Full JRF Template:

**Note:**

You require an Oracle database to create a domain using full JRF template.

1. Run the configuration wizard using `config.sh` located in the `$FMW_HOME/oracle_common/common/bin` directory.

The first screen in the Config Wizard, **Create Domain** appears.

2. Choose **Create a new domain, and enter the required domain home path.****3. Click Next.**

The dependent templates are automatically selected. **The dependent templates are Oracle Enterprise Manager Plug-in for OTD - 12.2.1.x.0 [em] and Oracle JRF** screen appears.

4. In the Available Templates section, choose **Oracle Traffic Director - 12.2.1.2.0 [otd]** and click **Next**.

The **Application Location** screen appears. Keep the default value for Application location.

5. Click **Next**.

The **Administrator Account** screen appears. Enter the WebLogic Domain administration username and password. This information is needed to access WebLogic Server Control and Fusion Middleware Control.

6. Click **Next**.

The **Domain Mode and JDK** screen appears. Choose the Domain Mode *Development*. Leave the default JDK selection as it appears.

7. Click **Next**.

The **Database Configuration Type** screen appears. Enter the RCU DB connection information. Enter the following:

- Auto Configuration Option: *RCU Data*
- Vendor: *Oracle*
- Driver: *Oracle's Driver (Thin) for Service connection*; Version: 9.0.1 and later
- DBMS/Service - *\$\$SERVICE_ID* (xe for Oracle XE)
- Port: *\$\$DB_PORT* (default is 1521)
- Schema owner - *\$\$SCHEMA_PREFIX}_STB*
- Passwd - *\$\$DB_PASSWORD*

8. Click **Get RCU Configuration**. Wait for the verification process to complete.

9. Click **Next**.

The **Component DataSources** screen appears. Ensure that the hostname and port information is correct.

10. Click **Next**.

The **Component JDBC Schema Test** screen appears. Wait for the test to complete.

11. Click **Next**.

The **Advanced Configuration** screen appears. Select *Admin Server, Managed Servers, Clusters and Coherence*.

12. Click **Next**.

The *Administration Server* screen appears. Enter the following:

- Server Name: Use default (*AdminServer*)
- Listen Address: Use default (*All local Addresses*)
- Listen Port: *\$\$WLS_ADMIN_PORT*
- Enable SSL: Use default (*unchecked*)

- SSL Port: Use default (*disabled*)
- Server Groups: Use default (*unspecified*)

13. Click Next.

The **Node Manager** screen appears. Do not add any nodes. Use defaults (default: *Per Domain*). For Node Manager Credentials, enter the following and click **Next**.

- User Name: *\$NM_USER*
- Password: *\$NM_PASSWORD*
- Confirm password: *\$NM_PASSWORD*

14. Click Next.

The **Managed Servers** screen appears. Do not add any managed servers.

15. Click Next.

The **Clusters** screen appears. Do not add any clusters.

16. Click Next.

The **Coherence Clusters** screen appears. Do not add any clusters.

17. Click Next.

The **Machines** screen appears. Click **Add** and enter the following information:

- Name: *\$MACHINE_NAME*
- Node Manager Listen Address: Use default (*localhost*)
- Node Manager Listen Port: *\$NM_PORT*

18. Click Next.

The **Configuration Summary** screen appears. If the message **The Security configuration in your domain is invalid** appears, you may ignore it.

19. Click Create.

The Configuration Progress screen appears.

20. Click Next.

Wait for this part of the configuration to complete. Depending on the location and performance of the Repository database, this process may take a few minutes. Click **Finish**. The *End of Configuration* screen appears.

 **Note:**

The creation of Oracle Traffic Director configuration/instance is not supported using configuration wizard, the system component screen in the configuration wizard should be ignored while configuring Oracle Traffic Director. For creating Oracle Traffic Director configurations/instances, use either Oracle Traffic Director custom WLST commands or Fusion Middleware Control.

Creating a Repository using Repository Creation Utility in Graphical Mode

Before proceeding to the next tasks, use the Repository Creation Utility (RCU). RCU is available with the Oracle Fusion Middleware Infrastructure distribution. Follow these steps.

1. Run `$FMW_HOME/oracle_common/bin/rcu.sh`
2. The *Welcome* page appears. Click **Next**.
3. The *Create Repository* page appears. Select *CreateRepository*, and *System Load and Product Load* (default). Click **Next**.
4. The Database Connection Details page appears. Enter the RCU DB connection information as shown in the screen below. Click **Next**.
5. The *Checking Prerequisites* box pops up. It shows the progress of prerequisites checking. When it is complete, click **OK**.
6. The *Select Components* page appears. Select the *Create newprefix* radio button and provide a schema prefix (such as DEMO). Select the following components: *Oracle Platform Security Services*, *Audit Services*, *Audit Services Append* and *Audit Services Viewer*. Click **Next**.
7. The *Checking Prerequisites* box pops up. It shows the progress of prerequisites checking. When it is complete, click **OK**.
8. The *Schema Passwords* page appears. Leave the default *Use same passwords for all schemas* radio button selected, and enter the password in the *Password* field. Click **Next**.
9. The Map Tablespaces page appears. No action is required. Click **Next**.
10. A *Repository Creation Utility* box pops up, requiring your confirmation. Click **OK**.
11. A *Creating Tablespaces* pop up appears, showing the progress of tablespace creation. Click **OK**, then **Next**.
12. The Summary page appears, showing your actions and choices. Click **Create**.
13. A System Load progress box appears, showing progress. The box will disappear when complete.
14. Click **Close**.

Creating a Repository in Silent Mode

When you install Oracle Traffic Director in a managed domain, the recommended configuration is to use the Restricted JRF domain template. In that configuration a database is not required, and you do not have to create a repository. However, in the case where you did not use the Restricted JRF domain template in the managed domain, you will require a database and a repository with schema space for the domain. To create a repository, run the repository creation.

```
$ORACLE_HOME/oracle_common/bin/rcu -silent -createRepository -  
connectString $DB_HOST:$DB_PORT:$SERVICE_ID  
-dbUser $DB_USER -dbRole $DB_ROLE -schemaPrefix $SCHEMA_PREFIX -  
useSamePasswordForAllSchemaUsers true  
-selectDependentsForComponents true -component OPSS -component IAU -f <  
<path_to_password_file>
```

The contents of the password file should appear as follows:

```
Password1
Password1
Password1
Password1
Password1
Password1
```

Example:

```
$ORACLE_HOME/oracle_common/bin/rcu -silent -createRepository -connectString $
{DB_HOST}:1521:xe
-dbUser sys -dbRole SYSDBA -schemaPrefix $SCHEMA_PREFIX -
useSamePasswordForAllSchemaUsers true -selectDependentsForComponents true
-component OPSS -component IAU -f < /tmp/pass.txt
```

```
Processing command line ...
Repository Creation Utility - Checking Prerequisites
Checking Global Prerequisites
The database you are connecting is not a supported version. Refer to the
certification matrix for supported DB versions.
...
Repository Creation Utility - Create : Operation Completed
```

Logging to the Administration Console

After installing Oracle Traffic Director, you can verify the installation by trying to log in to the administration console of the Oracle Traffic Director administration server, by performing the following steps:

1. Start the administration server instance, by running the following command:

```
$DOMAIN_HOME/bin/startWebLogic.sh
```

2. In your web browser, enter the URL that you noted in the previous step.

```
https://bin.example.com:1895/em
```

An error message about a problem with the server's security certificate is displayed. The text of the message varies depending on the browser you use. The error message is displayed because the Oracle Traffic Director administration server uses a self-signed certificate, rather than a certificate issued by a trusted certificate authority.

3. Proceed to the log-in page of the administration console by choosing to trust the certificate.

The steps to be performed to trust a certificate vary depending on the browser you use. As an example, in Mozilla Firefox 4.0, click on the **I Understand the Risks** link on the error page, then click the **Add Exception** button, and finally, on the result page, click the **Confirm Security Exception** button.

4. Log in using the administrator user name and password that you specified while creating the administration server instance.

Creating the Load Balancer for a Managed Domain

You can create a load balancer after deploying Oracle Traffic Director in a managed domain by using either Fusion Middleware Control or the WLST as described in the following topics.

Using Fusion Middleware Control

To create a load balancer by using Fusion Middleware Control, do the following:

1. Log in to Fusion Middleware Control for Traffic Director, as described in [Graphical User Interface-Fusion Middleware Control](#).
2. Click the WebLogic Domain button at the upper left corner of the page.
3. Select Administration > OTD Configurations.
4. In the Common Tasks pane select the Create button.

The Create Configuration wizard opens.

5. Follow the on-screen prompts to complete creation of the configuration by using the details as specified below.
 - Name: hr-config
 - (Listener) Port number: 1905
 - IP Address: default
 - Server Name: default
 - Origin Server Name: hr-apps.example.com
 - Origin Server Port Number: 80

6. In the Deployment page, check the listed machine under the Machine list and click Next.

The Create Configuration: Review page appears.

7. Click Create Configuration.

After the configuration is created, the Results screen of the New Configuration wizard displays a message confirming successful creation of the configuration. If you chose to create instances of the configuration, then a message confirming successful creation of the instances is also displayed.

8. Perform lifecycle tasks.
 - a. In the left panel, Target Navigation, click and expand Traffic Director.

The instances are listed under the configurations.

- b. Click on the instance you want to start.

- c. To start an instance.

- i. Navigate to the resulting page on the right panel.
 - ii. Click Start Up. The following message appears.

Start Operation on target /Domain_em_domain/em_domain/otd_hr-config_example.com. Completed Successfully

- d. To stop an instance.

- i. Click Shut Down.

- ii. Provide your confirmation in the resulting confirmation dialogue box. The following message appears.

Shutdown Operation on target /Domain_em_domain/em_domain/otd_hr-config_example.com completed successfully.

Using WLST Commands

To create a load balancer using offline WLST commands, do the following tasks:

1. Launch WLST command and connect to the Administration server.

```
$ORACLE_HOME$/oracle_common/common/bin/wlst.sh
> connect("$WLS_ADMIN_USER", "$WLS_ADMIN_PASSWORD", "t3://localhost:$
{WLS_ADMIN_PORT}")
```

2. Start an edit session.

```
> editCustom()
> startEdit()
```

3. Create a configuration `hr-config` using the `otd_createConfiguration` WLST command.

```
props = {}
props['configuration'] = 'hr-config'
props['listener-port'] = '1905'
props['server-name'] = 'hr-apps.example.com'
props['origin-server'] = 'hr-1.example.com:80,hr-2.example.com:80'
otd_createConfiguration(props)
```

4. Create an instance of the configuration `hr-config` by running the `otd_createInstance` WLST command. Specify the machine as the name you specified when creating the machine in Fusion Middleware Control, corresponding to the host name of the machine on which the Oracle Traffic Director instance is running.

```
props = {}
props['configuration'] = 'hr-config'
props['machine'] = 'machine1'
otd_createInstance(props)
```

5. Activate the changes. This propagates the changes to each instance of the configuration and creates it only if it currently does not exist.

```
activate()
```

6. Once the command runs successfully, check if the instance directory is created from the system command line.

```
ls $DOMAIN_HOME/config/fmwconfig/components/OTD/instances/hr-config_machine1
```

7. Perform lifecycle tasks.

```
> start("hr-config_machine1", 'SystemComponent')
Starting system component "hr-config_machine1" ....
System component with name "hr-config_machine1" started successfully
[Handle the machine-name as we do in schema-prefix]

> state("hr-config_machine1")
Current state of "hr-config_machine1" : RUNNING

> softRestart("hr-config_machine1")
Restarting the system component with name "hr-config_machine1" ....
System component with name "hr-config_machine1" restarted successfully

> shutdown("hr-config_machine1")
Shutting down the system component with name "hr-config_machine1" ....
System component with name "hr-config_machine1" shutdown successfully
```

```
> state("hr-config_machine1")
Current state of "hr-config_machine1" : SHUTDOWN
```

We have now successfully created an Oracle Traffic Director configuration, and started the instance.

Creating a Standalone Domain

Learn how to create Oracle Traffic Director on a standalone domain using Oracle Traffic Director standalone template.

You can create a standalone domain by using either Coconfiguration Wizard or WLST as described in the following topics:

- [Creating a Standalone Domain using the Configuration Wizard](#)
- [Creating a Standalone Domain Using Offline WLST Commands](#)
- [Creating the Load Balancer for a Standalone Domain](#)

Creating a Standalone Domain using the Configuration Wizard

To create a standalone domain using Configuration Wizard, do the following tasks:

1. Run the installer.

```
$ORACLE_HOME/oracle_common/common/bin/config.sh -log=config.log
```

The **Create Domain** page appears.

2. Use the default values and click **Next**.

The **Templates** page appears.

Note:

- The **Basic Standalone System Component Domain - 12.2.1.x.0 [wlserver]** is selected by default.
- The other Oracle Traffic Director templates, **Oracle Traffic Director - 12.2.1.x.0 [otd]** and **Oracle Traffic Director - Restricted JRF - 12.2.1.x.0 [otd]** are not applicable for standalone domain.

3. In the Available Templates section, choose **Oracle Traffic Director - Standalone 12.2.1.x.0 [otd]** and click **Next**.

The **JDK Selection** page appears

4. Retain the default JDK and click **Next**.

The **System Components** page appears.

5. Click **Next**.

The **Node Manager** page appears.

 **Note:**

You need not create system component, as Oracle Traffic Director has commands for configuring system components.

6. Select the following options.
 - Node Manager Type: default/Per Domain Default Location
 - User Name: *USERNAME*
 - Password: *PASSWORD*
 - Confirm Password: *PASSWORD*
7. Click **Next**
The **Configuration Summary** page appears.
8. Click **Create**.
The **Configuration Progress** page appears. This shows the progress of the configuration.
9. After the standalone domain is created successfully, click **Next**.
The **Configuration Success** page appears.
10. Click **Finish**.

Creating a Standalone Domain Using Offline WLST Commands

To create a standalone domain using offline WLST commands, do the following tasks:

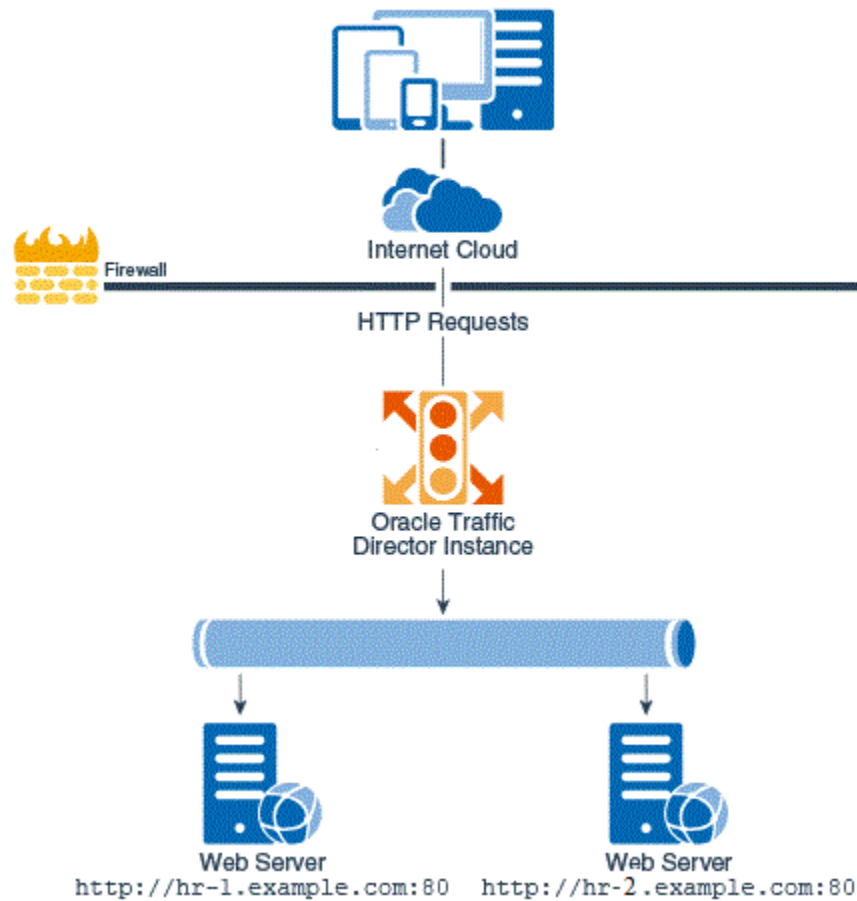
1. Launch the WLST command shell.

```
$ORACLE_HOME/oracle_common/common/bin/wlst.sh
```
2. Run the `otd_createStandaloneDomain` command to create an Oracle Traffic Director standalone domain. See `otd_createStandaloneDomain` command in *WebLogic Scripting Tool Command Reference for Oracle Traffic Director*.

```
> props = {'domain-home': '$DOMAIN_HOME'}  
> otd_createStandaloneDomain(props)
```

Creating the Load Balancer for a Standalone Domain

The following section describes creating a load balancer after deploying Oracle Traffic Director in standalone mode and creating a standalone domain as shown above. There is no Fusion Middleware Control in standalone domain. You should use Oracle Traffic Director offline WLST commands for creating the load balancer.



The topology is based on the following configuration:

- Oracle Traffic Director to receive requests from clients: `hr-apps.example.com:1905`
- Host and port of origin servers (web servers in this example):
 - `hr-1.example.com:80`
 - `hr-2.example.com:80`

In the real world, both origin servers would serve identical content. But for this example, to be able to see load balancing in action, we will set up the `index.html` page to show slightly different content, as follows:

- For `hr-1.example.com:80`: **"Page served from origin-server 1"**
- For `hr-2.example.com:80`: **"Page served from origin-server 2"**
- Load-balancing method is set as `Round robin`.

 **Note:**

The custom WLST commands for administering Oracle Traffic Director cannot be used in a standalone domain.

Using Offline WLST Commands

1. Launch the WLST command shell.

```
$ORACLE_HOME/oracle_common/common/bin/wlst.sh
```

2. Create an Oracle Traffic Director instance.

```
> props = {'domain-home': '$DOMAIN_HOME', 'origin-server': 'hr-1.example.com:80,hr-2.example.com:80', 'listener-port': '1905', 'instance': 'hr', 'server-name': 'hr-apps.example.com'}
```

```
> otd_createStandaloneInstance(props)
```

3. Once the command runs successfully, check if the instance directory is created from the system command line.

```
ls $DOMAIN_HOME/config/fmwconfig/components/OTD/instances/hr
```

4. Perform instance management tasks.

- Start

```
$DOMAIN_HOME/config/fmwconfig/components/OTD/instances/hr/bin/startserv
```

- Stop

```
$DOMAIN_HOME/config/fmwconfig/components/OTD/instances/hr/bin/stop
```

- Restart

```
$DOMAIN_HOME/config/fmwconfig/components/OTD/instances/hr/bin/restart
```

- Reconfigure

```
$DOMAIN_HOME/config/fmwconfig/components/OTD/instances/hr/bin/reconfig
```

- Rotate log files

```
$DOMAIN_HOME/config/fmwconfig/components/OTD/instances/hr/bin/rotate
```

Verifying the Load-Balancing Behavior of the Oracle Traffic Director Instance

You can verify the load-balancing behavior of the Oracle Traffic Director instance by using your browser. The Oracle Traffic Director instance that we created and started earlier is now listening for HTTP requests at the URL `http://hr-apps.example.com:1905`.

Ensure that the web servers `hr-1.example.com:80` and `hr-2.example.com:80` are running. If necessary, update the `/etc/hosts` file on the host from which you are going to access the Oracle Traffic Director virtual server, to make sure that the browser can resolve `hr-apps.example.com` to the correct IP address.

1. Enter the URL `http://hr-apps.example.com:1905` in your browser.

A page with the following text is displayed:

```
"Page served from origin-server 1"
```

This indicates that the Oracle Traffic Director instance running on the `apps.example.com` Node Manager received the HTTP request that you sent from the browser, and forwarded it to the origin server `hr-1.example.com:80`.

2. Send another HTTP request to `http://hr-apps.example.com:1905` by refreshing the browser window.

A page with the following text is displayed:

```
"Page served from origin-server 2"
```

This indicates that Oracle Traffic Director sent the second request to the origin server `hr-2.example.com:80`

3. Send a third HTTP request to `http://hr-apps.example.com:1905` by refreshing the browser window again.

A page with the following text is displayed:

```
"Page served from origin-server 1"
```

This indicates that Oracle Traffic Director used the simple round-robin load-distribution method to send the third HTTP request to the origin server `hr-1.example.com:80`.

4

Configuring Oracle Traffic Director for High Availability

High availability is the ability of a system or device to be available when it is needed. A high availability architecture ensures that users can access a system without loss of service. Deploying a high availability system minimizes the time when the system is down, or unavailable, and maximizes the time when it is running, or available.

This section describes the instructions for configuring Oracle Traffic Director for failover across Oracle Traffic Director instances. It includes the following sections:

- [Overview](#)
- [Preparing your System for High Availability](#)
- [Configuring High Availability](#)

Overview

The high availability solution for Oracle Traffic Director provides the ability to create multiple instances and then configure the IP failover for a given VIP between the instances.

In Oracle Traffic Director, high availability solution creates a redundancy for a given virtual IP address (VIP) by configuring IP failover between two or more instances. The IP failover is configured as a **Failover Group** which is a grouping of a VIP, an instance designated as the primary and one or more instances designated as the backup instances. The failover is transparent to both the client which is sending the traffic and the Oracle Traffic Director instance that is receiving the traffic.

Failover configuration modes

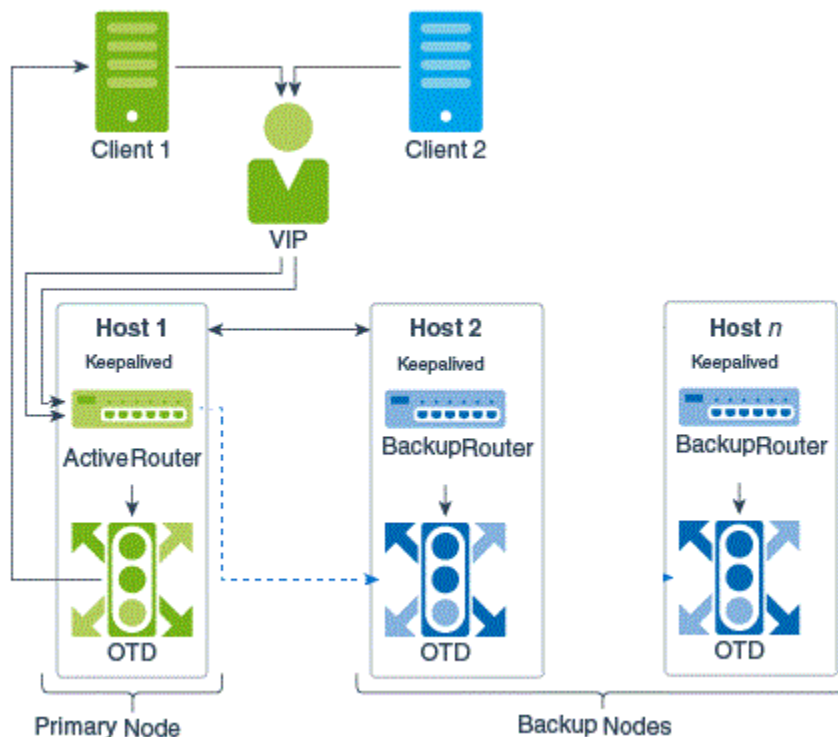
You can configure the Oracle Traffic Director instances in a failover group to work in the following modes:

- **Active-passive:** A single VIP address is used. One instance in the failover group is designated as the primary node. If the primary node fails, the requests are routed through the same VIP to the other instance.
- **Active-active:** A single VIP address is used. One of the nodes is the master node, and the other nodes are backup nodes. The incoming requests to VIP is distributed among the OTD instances. If the master node fails, then the backup node having the highest priority will be chosen as the next master node.

Failover in Active-Passive Mode

In the active-passive setup described here, one node in the failover group is redundant at any point in time.

Oracle Traffic Director provides support for failover between the instances in a failover group by using an implementation of the Virtual Routing Redundancy Protocol (VRRP), such as `keepalived` for Linux and `vrpd` (native) for Solaris. This mode of failover is supported only on Solaris and Linux platforms.



Keepalived provides other features such as load balancing and health check for origin servers, but Oracle Traffic Director uses only the VRRP subsystem. For more information about Keepalived, go to <http://www.keepalived.org>.

VRRP specifies how routers can failover a VIP address from one node to another if the first node becomes unavailable for any reason. The IP failover is implemented by a router process running on each of the nodes. In a two-node failover group, the router process on the node to which the VIP is currently assigned is called the master. The master continuously advertises its presence to the router process on the second node.

Note:

On a Linux host that has an Oracle Traffic Director instance configured as a member of a failover group, Oracle Traffic Director should be the only consumer of Keepalived. Otherwise, when Oracle Traffic Director starts and stops the `keepalived` daemon for effecting failovers during instance downtime, other services using `keepalived` on the same host can be disrupted.

If the node on which the master router process is running fails, the router process on the second node waits for about three seconds before deciding that the master is down, and then assumes the role of the master by assigning the VIP to its node. When

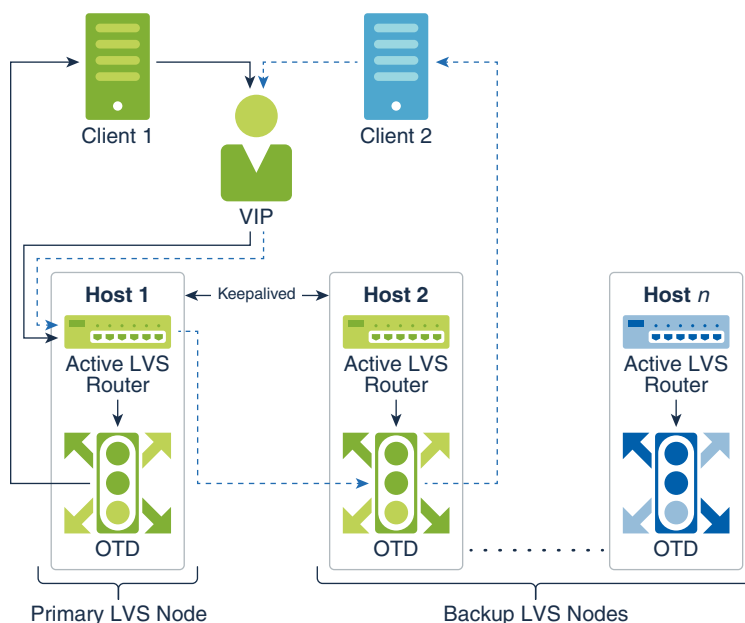
the first node is online again, the router process on that node takes over the master role. For more information about VRRP, see RFC 5798 at <http://datatracker.ietf.org/doc/rfc5798>.

Failover in Active-Active Mode

Oracle Traffic Director provides support for failover between the instances by deploying instances on the nodes which are in the same subnet. One of the nodes is chosen as the active router node and the remaining node(s) are the backup router node(s). The traffic will be managed among all the Oracle Traffic Director instances.

This mode of failover is supported only on the Linux platform. The solution also uses Keepalived v 1.2.13 and Linux Virtual Server (LVS) to perform load balancing and failover tasks. In addition, the following packages are required.

- ipvsadm (1.26 or later)
- iptables (1.4.7 or later)



In the beginning, all the nodes are configured as the backup nodes and the nodes are assigned different priorities. The highest priority node is chosen as the master and the other nodes are the backup nodes. If the master node fails, then the backup node having the highest priority is chosen as the next master node. The keepalived master node will also be the master node for LVS.

Keepalived does following:

- Plumbs the virtual IP on the master
- Sends out gratuitous ARP messages for the VIP
- Configure the LVS (ipvsadm)
- Health-check for Keepalived on other nodes

LVS does following:

- Balance the load across the Oracle Traffic Director instances
- Share the existing connection information to the backup nodes via multicasting.
- To check the integrity of the services on each Oracle Traffic Director instance. In case, an Oracle Traffic Director instance fails then that instance will be removed from the LVS configuration and when it comes back online then it will be added again.

Preparing your System for High Availability

The typical deployment topology for Oracle Traffic Director is a three-node installation which are part of a single domain.

- There is the WebLogic Server Administration machine on which Oracle Traffic Director is collocated with a WebLogic Server and JRF installation. This hosts the WebLogic Server Administration server.
- There are two other machines which have the Oracle Traffic Director standalone installations which have only a subset of WebLogic Server binaries and which hosts managed Oracle Traffic Director domains.
- The two Oracle Traffic Director instances typically provide high availability for a VIP (virtual IP) by forming a failover group.

Prerequisites

Operating System	High Availability Requirements	System Requirements
Linux	Active-Active <ul style="list-style-type: none"> • keepalived package (1.2.13 or later) 	<ul style="list-style-type: none"> • 3 machines with OEL 6.6 or above. • 2 machines must be in the same subnet.
Linux	Active-Passive <ul style="list-style-type: none"> • keepalived package (1.2.12 or later) 	<ul style="list-style-type: none"> • A virtual IP on the same subnet to create a failover group. This IP address should be free and should not already be assigned to any machine on the network.
Solaris	Active-Active	Not Supported
Solaris	Active-Passive <ul style="list-style-type: none"> • vrrpadm package • ipadm package 	<ul style="list-style-type: none"> • 3 machines with S11u2 or above. • 2 machines must be in the same subnet. • A virtual IP on the same subnet to create a failover group. This IP address should be free and should not already be assigned to any machine on the network.
Windows	Active-Active	Not Supported
Windows	Active-Passive	Not Supported
AIX	Active-Active	Not Supported
AIX	Active-Passive	Not Supported

Notations

The following tokens are used to represent the same data throughout this chapter.

Token	Description
OTD_HOST_1	Host name of the machine on which the primary instance of the failover group is running.
OTD_HOST_2	Host name of the machine on which the back up instance of the failover group is running.
OTD_MACHINE_1	Name specified while creating machine corresponding to OTD_HOST_1 in Weblogic Server Console.
OTD_MACHINE_2	Name specified while creating machine corresponding to OTD_HOST_2 in Weblogic Server Console.
VIP	The virtual IP (on the same subnet as OTD_HOST_1 and OTD_HOST_2) used to create a failover group.
WLS_ADMIN_HOST	Host name of the server on which WebLogic Server admin runs.

Configuring High Availability

Configuring high availability for Oracle Traffic Director starts with installing WebLogic Server, Oracle Traffic Director. It also includes creating a WebLogic domain, a standalone Oracle Traffic Director installation and then creating failover instances.

1. Install WebLogic Server. See [Installing WebLogic Server](#).
2. Install Oracle Traffic Director on a collocated mode with WebLogic Server and restricted JRF template. See [Installing Oracle Traffic Director](#).
3. Create a WebLogic domain using JRF template to manage Oracle Traffic Director instances on remote machines. See [Creating a Domain using JRF Template](#).
4. Create a standalone Oracle Traffic Director installation on 2 other hosts OTD_HOST_1 and OTD_HOST_2. See [Installing Oracle Traffic Director in a Standalone Domain](#).
5. Create Managed Domains with Standalone Oracle Traffic Director Installations.
 - a. Open a new terminal on the WLS_ADMIN_HOST and start the WebLogic Administration Server.
Syntax: `$DOMAIN_HOME/startWeblogic.sh`
 - b. Log on to the WebLogic Server console at `http://$WLS_ADMIN_HOST:$WLS_ADMIN_PORT/console` with credentials WLS_ADMIN_USER and WLS_ADMIN_PASSWORD.
 - c. Create machine entries corresponding to OTD_HOST_1 and OTD_HOST_2 with the properties shown in the table below. For information on creating and configuring machines, see [Create and configure machines](#).

Table 4-1 Create machine entries using these values

Parameter	Sub-Parameter	Value
Machine Properties	Name	It is recommended that this be the same as the host name of the machine (OTD_HOST_1 or OTD_HOST_2). This name will be referred to hereafter as OTD_MACHINE_1 and OTD_MACHINE_2.
	Machine Properties-Machine OS	Use the default value.
Node Manager Properties	Type	SSL
	Listen Address	Host name of the remote host (OTD_HOST_1 or OTD_HOST_2).
	Listen Port	The port the Node Manager is configured to listen on the remote host (default is 5556).

For example:

Table 4-2 Example for creating machine entries

Parameter	Sub-Parameter	Value
Machine Properties	Name	example.com.
	Machine Properties-Machine OS	Use the default value
Node Manager Properties	Type	SSL
	Listen Address	example.com.
	Listen Port	5556

- d. Open a terminal on WLS_ADMIN_HOST and execute the pack.sh command.

 **Note:**

Ensure that both the remote hosts OTD_HOST_1 and OTD_HOST_2 are added as machines on the console before packing the domain.

Syntax: \$ORACLE_HOME/oracle_common/common/bin/pack.sh -domain=<full path to the domain that needs to be packed> -template=<full path to a template jar file to be created> -template_name=<description> -managed=true

Example: \$ORACLE_HOME/oracle_common/common/bin/pack.sh -domain=\$DOMAIN_HOME -template=/share/files/otd_ha.jar -template_name=ha -managed=true

This command creates a template archive .jar file that contains a subset of the domain that can be used to create an Oracle Traffic Director managed domain on the remote machine.

- e. Copy the template jar created by pack to OTD_HOST_1 and OTD_HOST_2 or keep the jar in a file system location which can be accessed from OTD_HOST_1 and OTD_HOST_2.
- f. Execute the `unpack.sh` command on both OTD_HOST_1 and OTD_HOST_2 to create an Oracle Traffic Director managed domain.

Syntax: `$ORACLE_HOME/oracle_common/common/bin/unpack.sh - domain=<full path to the domain that needs to be created> - template=<full path to the template jar file created using pack>`

Example: `$ORACLE_HOME/oracle_common/common/bin/unpack.sh - domain=$DOMAIN_HOME -template=/share/files/otd_ha.jar`

- g. Start the Node Manager in a new terminal on both OTD_HOST_1 and OTD_HOST_2.
Syntax: `$DOMAIN_HOME/bin/startNodeManager.sh`
- h. Log on to the Weblogic Server Console on WLS_ADMIN_HOST and ensure that the status of the Node Manager on OTD_HOST_1 and OTD_HOST_2 is shown as active. For information about monitoring the Node Monitor status, see [Monitor Node Manager status](#).

6. Create Remote Instances and Enable Failover.

- a. Install Oracle Traffic Director on two remote hosts OTD_HOST_1 and OTD_HOST_2.

Note:

To enable failover, the two remote hosts must be on the same subnet.

- b. Ensure that WebLogic Server is running on WLS_ADMIN_HOST and Node Managers are running on OTD_HOST_1 and OTD_HOST_2.
- c. Connect to the Administration Server and start an edit session using WLST command interface.

Syntax:

```
> $ORACLE_HOME/oracle_common/common/bin/wlst.sh
>
connect('$WLS_ADMIN_USER', '$WLS_ADMIN_PASSWORD', 't3://$WLS_ADMIN_HOST:$WLS_ADMIN_PORT')
> editCustom()
> startEdit()
```

- d. Create a new Oracle Traffic Director configuration.

Syntax:

```
> props = {'origin-server': '<origin servers>', 'listener-port': '<listener port>', 'name': '<config name>', 'server-name': '<server
```

```
name>'}  
> otd_createConfiguration(props)
```

Example:

```
> props = {'origin-server': 'localhost:20004', 'listener-port':  
'20009', 'name': 'ha', 'server-name': 'myservername'}  
> otd_createConfiguration(props)
```

e. Create Oracle Traffic Director instance on the 2 remote machines.**Syntax:**

```
> props={'configuration': '<config name>', 'machine':  
'$OTD_MACHINE_1'}  
> otd_createInstance(props)  
> props={'configuration': '<config name>', 'machine':  
'$OTD_MACHINE_2'}  
> otd_createInstance(props)
```

This creates 2 instances, `otd_<config name>_OTD_MACHINE_1` and `otd_<config name>_OTD_MACHINE_2`.

```
> props={'configuration': 'ha', 'machine': 'example.com'}  
> otd_createInstance(props)  
> props={'configuration': 'ha', 'machine': 'example.com'}  
> otd_createInstance(props)
```

f. Create a failover group using the two instances.

- To create an active-active failover:

- **Syntax:**

```
> props = {'configuration': '<config name>', 'virtual-ip':  
'$VIP', 'failover-type': 'active-active'}  
> otd_createFailoverGroup(props)
```

Example:

```
> startEdit()  
> props = {'configuration': 'ha', 'virtual-ip':  
'10.20.30.40', 'failover-type': 'active-active'}  
> otd_createFailoverGroup(props)
```

- Add 2 failover instance details from different nodes to the active-active Failover group.

Syntax:

```
> props = {'configuration': '<config name>', 'virtual-ip':  
'$VIP', 'instance': 'otd_ha_OTD_MACHINE_1', 'nic':  
'<network interface name>'}  
> otd_addFailoverInstance(props)
```

Example:

```
> startEdit()
> props = {'configuration': 'ha', 'virtual-ip':
'10.20.30.40', 'instance': 'otd_ha_example1.com', 'nic':
'eth0'}
> otd_addFailoverInstance(props)
> props = {'configuration': 'ha', 'virtual-ip':
'10.20.30.40', 'instance': 'otd_ha_example2.com', 'nic':
'eth0'}
> otd_addFailoverInstance(props)
```

- Verify the failover group creation.

Syntax:

```
> props = {'configuration': '<config name>', 'virtual-ip':
'$VIP'}
> otd_getFailoverGroupProperties(props)
```

Example:

```
> props = {'configuration': 'ha', 'virtual-ip':
'10.20.30.40'}
> otd_getFailoverGroupProperties(props)
```

The details of the created group are displayed as follows:

```
> failover-type=active-active
> router-id=72
> virtual-ip=10.20.30.40
> instances=otd_ha_example1.com,otd_ha_example2.com
```

- To create an active-passive failover:

Syntax:

```
> props = {'configuration': '<config name>', 'virtual-ip':
'$VIP', 'primary-instance': 'otd_ha_${OTD_MACHINE}_1', 'backup-
instance': 'otd_ha_${OTD_MACHINE}_2',
          'primary-nic': 'network interface on the primary
instance', 'backup-nic': 'network interface on the backup
instance'}
> otd_createFailoverGroup(props)
```

 **Note:**

- VIP should be an IP address on the same subnet as OTD_HOST_1 and OTD_HOST_2.
- Ensure that VIP has not been assigned to any machine on the subnet.

Example:

```
> startEdit()
> props = {'configuration': 'ha', 'virtual-ip': '10.20.30.40',
'primary-instance': 'otd_ha_example.com', 'backup-instance':
'otd_ha_example.com',
'primary-nic': 'eth0', 'backup-nic': 'eth0'}
> otd_createFailoverGroup(props)
```

- g.** Activate the changes. This propagates the failover configuration to the 2 instances on OTD_HOST_1 and OTD_HOST_2.

Syntax: `activate()`

The instance is created in the path, `<DOMAIN_HOME>/config/fmwconfig/components/OTD/instances/otd_<config name>_<machine name>` on both the machines.

- 7.** Start Failover. After creating remote Oracle Traffic Director instances on the remote hosts OTD_HOST_1 and OTD_HOST_2, you must start failover using *Superuser* privileges.

Ensure that Weblogic Server is running on WLS_ADMIN_HOST and Node Manager is running on both OTD_HOST_1 and OTD_HOST_2.

- a.** Using the WLST Command interface on the WLS_ADMIN_HOST connect to the WebLogic Administration Server.

Syntax:

```
> $ORACLE_HOME/oracle_common/common/bin/wlst.sh
> connect('$WLS_ADMIN_USER', '$WLS_ADMIN_PASSWORD', 't3://
localhost:$WLS_ADMIN_PORT')
```

- b.** Start instance `otd_<config name>_<OTD_MACHINE_N>` using WLST command.

Syntax:

```
> start('otd_<config name>_<OTD_MACHINE_1>', 'SystemComponent')
> state('otd_<config name>_<OTD_MACHINE_1>')
Current state of "otd_<config name>_<OTD_MACHINE_1>" : RUNNING
> start('otd_<config name>_<OTD_MACHINE_2>', 'SystemComponent')
> state('otd_<config name>_<OTD_MACHINE_2>')
Current state of "otd_<config name>_<OTD_MACHINE_2>" : RUNNING
```


Example:

```
> start('otd_ha_example.com', 'SystemComponent')
> state('otd_ha_example.com')
Current state of "otd_ha_example.com" : RUNNING
> start('otd_ha_example.com', 'SystemComponent')
> state('otd_ha_example.com')
Current state of "otd_ha_example.com" : RUNNING
```

- c. Start failover on both OTD_HOST_1 and OTD_HOST_2. Run `otd_startFailover` with *Superuser* privileges since the failover daemon needs to be started as root.

On OTD_HOST_1, run the following:

```
sudo $ORACLE_HOME/oracle_common/common/bin/wlst.sh
> props={'domain-home': $DOMAIN_HOME, 'instance':
otd_ha_$OTD_MACHINE_1, 'log-verbose': 'true'}
> otd_startFailover(props)
```

On OTD_HOST_2, run the following:

```
sudo $ORACLE_HOME/oracle_common/common/bin/wlst.sh
> props={'domain-home': $DOMAIN_HOME, 'instance':
otd_ha_$OTD_MACHINE_2, 'log-verbose': 'true'}
> otd_startFailover(props)
```

For example:

```
sudo $ORACLE_HOME/oracle_common/common/bin/wlst.sh

Initializing WebLogic Scripting Tool (WLST) ...

Welcome to WebLogic Server Administration Scripting Shell

Type help() for help on available commands

wls:/offline> props = {'domain-home': '$ORACLE_HOME/user_projects/
domains/base_domain', 'instance': 'otd_ha_example.com', 'log-
verbose': 'true'}
wls:/offline> otd_startFailover(props)

OTD-67856 Failover has been started successfully.
```

5

Managing Configurations

The first step toward creating a load-balanced service with Oracle Traffic Director is to create a configuration. A configuration is a collection of metadata defining the run-time characteristics of an Oracle Traffic Director server. After creating a configuration, you can use it to create instances of Oracle Traffic Director servers on one or more administration nodes.

This chapter contains the following topics:

- [Creating a Configuration](#)
- [Viewing a List of Configurations](#)
- [Activating Configuration Changes](#)
- [Modifying an Oracle Traffic Director Configuration](#)
- [Copying an Oracle Traffic Director Configuration](#)
- [Deleting an Oracle Traffic Director Configuration](#)

See [Oracle Traffic Director Terminology](#) for the definitions of the Oracle Traffic Director terminology such as *configuration*, *administration node*, and *instance*. To understand the relationship between configurations, node managers, and instances, see [Overview of Administration Tasks](#).

Creating an Oracle Traffic Director Configuration

You can create configurations that define your Oracle Traffic Director instances. A configuration is a collection of metadata that you can use to instantiate Oracle Traffic Director. Oracle Traffic Director reads the configuration when a server instance starts and while processing client requests.

Before You Begin

Before you begin creating a configuration, decide the following:

- A unique name for the configuration. Choose the name carefully. After creating a configuration, you cannot change the name.
- A unique listener `host:port` combination for the default virtual server that you create as part of the configuration.
- `host:port` addresses of the servers in the origin-server pool that you create as part of the configuration.
- (optional) Host names of the Node Managers on which you want to create instances of the configuration.

While creating a configuration using the New Configuration wizard, you can also choose to instantiate the configuration on one or more Node Managers. The wizard displays the host names of the Node Managers that are registered with the server.

Topics

- [Creating a Configuration Using Fusion Middleware Control](#)
- [Creating a Configuration Using WLST](#)

Creating a Configuration Using Fusion Middleware Control

To create a configuration by using Fusion Middleware Control, do the following:

1. Log in to Fusion Middleware Control for Traffic Director, as described in [Graphical User Interface-Fusion Middleware Control](#).
2. Click the WebLogic Domain button at the upper left corner of the page.
3. Select Administration > OTD Configurations.
4. In the Common Tasks pane select the Create button.

The New Configuration wizard opens.

5. Follow the on-screen prompts to complete creation of the configuration by using the details—origin server type, and so on—that you decided earlier.

After the configuration is created, the Results screen of the New Configuration wizard displays a message confirming successful creation of the configuration. If you chose to create instances of the configuration, then a message confirming successful creation of the instances is also displayed.

6. Click **Close** on the Results screen.

In the New Configuration wizard, if you chose not to create an instance of the configuration, the message **Undeployed Configuration** is displayed, indicating that the configuration that you just created is yet to be deployed.

Creating a Configuration Using WLST

To create a configuration, run the `otd_createConfiguration` command. This command can be run in online and offline mode.

The example creates a configuration named `soa.example.com` with an origin server, `vault.example.com:80`.

```
# Online
props = {}
props['name'] = 'soa.example.com'
props['listener-port'] = '12345'
props['server-name'] = 'foo'
props['origin-server'] = 'vault.example.com:80'
otd_createConfiguration(props)
```

The example creates a configuration named `foo` with an origin server, `vault.mycompany.com:80`.

```
# Offline
readDomain('/export/domains/otd_domain')
props = {}
props['name'] = 'foo'
props['listener-port'] = '12345'
props['server-name'] = 'foo'
props['origin-server'] = 'vault.mycompany.com:80'
```

```
otd_createConfiguration(props)
updateDomain()
closeDomain()
```

Viewing a List of Configurations

After creating Oracle Traffic Director configurations, you can view a list of the available configurations at any time. To view a list of configurations, run the `otd_listConfigurations` command.

You can view the list of configurations by using either Fusion Middleware Control or the WLST as described in the following topics:

Topics

- [Viewing a List of Configurations Using Fusion Middleware Control](#)
- [Viewing a List of Configurations Using WLST](#)

Viewing a List of Configurations Using Fusion Middleware Control

1. Log in to Fusion Middleware Control, as described in [Displaying Fusion Middleware Control](#).
2. Click the WebLogic Domain button at the upper left corner of the page.
3. Select Administration > OTD Configurations.

A list of the available configurations is displayed.

Configuration Name	Target Full Name	Deployed on Machines
mt	/Domain_otd_domain/otd_domain/mt	slc01nbd.us.oracle.com
config3	/Domain_otd_domain/otd_domain/config3	slc01nbd.us.oracle.com

You can view the properties of a configuration by clicking on its name.

Viewing a List of Configurations Using WLST

To view a list of the available configurations, run the `otd_listConfigurations` command, as shown in the following example. You can run this command in online and offline mode.

```
# Online
otd_listConfigurations()

# Offline
readDomain(' /export/2110_12c/iplanet/ias/server/work/TD_Linux2.6_DBG.OBJ/domains/
otd_domain')
otd_listConfigurations()
closeDomain()
```

See *Offline Commands* in the *WebLogic Scripting Tool Command Reference for Oracle Traffic Director* for information about using WLST commands in offline mode.

Activating Configuration Changes

You can activate configuration changes to instances using the `activate` command. The `activate` command activates changes done after starting an edit session by executing the command `startEdit`.


Apart from activating the changes to Oracle Traffic Director, the `activate` command also activates changes to other components and managed servers done after starting an edit session. Certain configuration changes cannot be applied dynamically without restarting the instances.

You can activate the configuration changes by using either Fusion Middleware Control or WLST as described in the following topics:

- [Activate Configuration Changes Using Fusion Middleware Control](#)
- [Activate Configuration Changes Using WLST](#)

Activate Configuration Changes Using Fusion Middleware Control

To activate configuration changes using the Fusion Middleware Control, do the following:

1. Log in to Fusion Middleware Control, as described in [Graphical User Interface-Fusion Middleware Control](#).
2. Click the  button just below Weblogic Domain.
3. The **Auto-Commit Mode** is enabled by default.
 - **Enable:** Enable Auto-Commit Mode and create a configuration. It displays a information that all changes have been activated.
 - **Disable:** Disable Auto-Commit Mode and create a configuration. It displays a information that changes are pending for activation. Use change center to activate pending changes.

Activate Configuration Changes Using WLST

All commands executed in WLST must be activated using the `activate` command.

For example, the following command updates all instances of the configuration with the latest configuration settings.

```
wls:/mydomain/edit !> activate(200000, block='true')
Activating all your changes, this may take a while ...
The edit lock associated with this edit session is released once the activation
iscompleted.
Action completed.
wls:/mydomain/edit>
```

 **Note:**

See [Updating Oracle Traffic Director Instances Without Restarting](#) for information about parameters that can be re-configured without restarting Oracle Traffic Director instances.

Modifying an Oracle Traffic Director Configuration

After you create a configuration and create instances from it, you might need to change some of the settings such as log preferences, performance parameters, virtual server listener, origin-server pools, and so on.

You can modify a configuration by using either Fusion Middleware Control or the WLST as described in the following topics:

Topics

- [Modifying a Configuration Using Fusion Middleware Control](#)
- [Modifying a Configuration Using WLST](#)

Modifying a Configuration Using Fusion Middleware Control

To modify a configuration by using the Fusion Middleware Control, do the following:

1. Log in to Fusion Middleware Control for Traffic Director, as described in [Displaying Fusion Middleware Control](#).
2. Click the WebLogic Domain button at the upper left corner of the page.
3. Select Administration > OTD Configurations.
A list of the available configurations is displayed.
4. Select the configuration that you want to modify.
5. In the navigation pane, you can select the following additional categories of settings for the configuration. The parameters relevant to the selected category are displayed on the main pane.
 - **SSL**
 - Schedule and manage CRL-update events. See [Update CRLs Automatically](#).
 - SSL/TLS caching preferences. See [SSL/TLS Session Caching](#).
 - **Logging**
 - Set and change parameters for the server log file-name and location of the log file, log level, date format, and so on.
 - Enable and disable the access log.
 - Set and change parameters for the access log file-name and location of the log file and log format
 - Schedule and manage events to rotate the server and access log files.

- Configure access-log buffer settings to tune performance. See [Managing Logs](#).
 - **Advanced Settings**
 - Specify general settings: the server user ID, the temporary directory in which the process ID and socket information for the instances of the configuration are stored, and the localization preferences.
 - Configure DNS lookup and cache settings. See [Tuning DNS Caching Settings](#).
 - Create, enable, disable, view, delete events for the configuration. See [Controlling Oracle Traffic Director Instances Through Scheduled Events](#).
 - **HTTP**: Set and change parameters to tune the performance of the virtual servers defined for the configuration such as, request buffer size, response buffer size, timeout thresholds for the request body and header, thread-pool settings, and keep-alive settings. See [Tuning HTTP Request and Response Limits](#).
 - **Monitoring**: Enable and disable statistics collection, profiling, and the SNMP subagent. Specify the statistics-collection interval. See [Monitoring Oracle Traffic Director Instances](#).
6. Specify the parameters that you want to change.

On-screen help and prompts are provided for all of the parameters.

When you change the value in a field or tab out of a text field that you changed, the **Save** button near the upper right corner of the page is enabled.

At any time, you can discard the changes by clicking the **Reset** button.

7. After making the required changes, click **Save**.
- A message, confirming that the updated configuration was saved, is displayed in the Console Messages pane.
 - In addition, the **Deployment Pending** message is displayed at the top of the main pane. You can either deploy the updated configuration immediately by clicking **Deploy Changes**, or you can do so later after making further changes as described in [Activate Configuration Changes](#).

 **Note:**

In the Advanced Settings page, if you change the **Temporary Directory** value, you should first stop all the instances of the configuration, deploy the changes, and then start the instances.

Modifying a Configuration Using WLST

WLST provides several commands that you can use to change specific parameters of a configuration.

Table 5-1 WLST Commands for Modifying a Configuration

Task	WLST Commands
Change the configuration properties	<code>otd_setConfigurationProperties</code>
Change access-log buffer properties	<code>otd_setAccessLogBufferProperties</code> <code>otd_getAccessLogBufferProperties</code>
Change caching properties	<code>otd_setCacheProperties</code> <code>otd_getCacheProperties</code>
Change DNS properties	<code>otd_setDnsProperties</code> <code>otd_getDnsProperties</code>
Change DNS caching properties	<code>otd_setDnsCacheProperties</code> <code>otd_getDnsCacheProperties</code>
Change HTTP request properties	<code>otd_setHttpProperties</code> <code>otd_getHttpProperties</code>
Change keep-alive settings for client connections	<code>otd_setKeepAliveProperties</code> <code>otd_getKeepAliveProperties</code>
Change error log settings	<code>otd_setLogProperties</code> <code>otd_getLogProperties</code>
Enable SNMP	<code>otd_setSnmpProperties</code> <code>otd_getSnmpProperties</code>
Change SSL/TLS session caching properties	<code>otd_setSslSessionCacheProperties</code> <code>otd_getSslSessionCacheProperties</code>
Change statistics collection properties	<code>otd_setStatsProperties</code> <code>otd_getStatsProperties</code>
Change TCP thread pool properties	<code>set-tcp-thread-pool-prop</code>
Change HTTP thread pool properties	<code>otd_setHttpThreadPoolProperties</code> <code>otd_getHttpThreadPoolProperties</code>
Change TCP thread pool properties	<code>otd_setTcpThreadPoolProperties</code> <code>otd_getTcpThreadPoolProperties</code>

For example, the following command changes the log level for the configuration `foo` to the most verbose (finest) setting, `TRACE:32`.

```
props = {}
props['configuration'] = 'foo'
otd_getConfigurationProperties(props)
```

See *WebLogic Scripting Tool Command Reference for Oracle Traffic Director* or run the commands with the `--help` option.

Copying an Oracle Traffic Director Configuration

When you want to create a configuration that is similar to an existing configuration, you can copy the existing configuration and make the required changes later.

You can copy a configuration by using either Fusion Middleware Control or the WLST as described in the following topics:

Topics

- [Copying a Configuration Using Fusion Middleware Control](#)
- [Copying a Configuration Using WLST](#)

Copying a Configuration Using Fusion Middleware Control

To copy a configuration by using the Fusion Middleware Control, do the following:

1. Log in to Fusion Middleware Control for Traffic Director, as described in [Displaying Fusion Middleware Control](#).
2. Click the WebLogic Domain button at the upper left corner of the page.
3. Select Administration > OTD Configurations.
A list of the available configurations is displayed.
4. Select the configuration that you want to copy.
5. In the Common Tasks pane, click **Duplicate Configuration**.
6. In the resulting dialog box, enter a name for the new configuration, and then click **OK**.
A message is displayed confirming that the configuration was copied.
7. Click **OK**.

Copying a Configuration Using WLST

To copy a configuration, run the `otd_copyConfiguration` command.

For example, the following command copies the configuration `foo` to a new configuration named `foo1`.

```
props = {}
props['source-configuration'] = 'foo'
props['dest-configuration'] = 'bar'
otd_copyConfiguration(props)
```

Deleting an Oracle Traffic Director Configuration

You can delete configurations that are not required. To delete Oracle Traffic Director configurations, run the `otd_deleteConfiguration` command.

You can delete Oracle Traffic Director configurations by using either Fusion Middleware Control or the WLST.

 **Note:**

To delete a configuration that has one or more failover groups, you should first delete the failover groups. See [Managing Failover Groups](#).

Topics

- [Deleting a Configuration Using Fusion Middleware Control](#)
- [Deleting a Configuration Using WLST](#)

Deleting a Configuration Using Fusion Middleware Control

To delete a configuration by using the Fusion Middleware Control, do the following:

1. Log in to Fusion Middleware Control, as described in [Displaying Fusion Middleware Control](#).
2. Click the WebLogic Domain button at the upper left corner of the page.
3. Select Administration > OTD Configurations.
A list of the available configurations is displayed.
4. Select the configuration that you want to delete.
5. In the Common Tasks pane, click **Delete Configuration**.
 - If there are no instances of the configuration that you want to delete, a prompt to confirm deletion of the configuration is displayed.
 - a. Click **OK**.
A message is displayed confirming that the configuration was deleted.
 - b. Click **OK**.
 - If there are instances of the configuration that you want to delete, a dialog box is displayed listing the administration nodes on which the configuration is deployed. The list also indicates whether the instances are running.
 - a. If you want to proceed with the deletion, you can choose to save the log files of the instances by selecting the **Save Instance Logs** check box.
To confirm deletion, click **OK**.
A message is displayed confirming that the configuration and its instances were deleted.
 - b. Click **OK**.

 **Note:**

If you selected the **Save Instance Logs** check box, the server access and error logs for the instances that were deleted are retained in the `INSTANCE_HOME/net-config_name/logs` directory.

6. Click the **Delete** button corresponding to the configuration that you want to delete.

Deleting a Configuration Using WLST

To delete a configuration, run the `otd_deleteConfiguration` command, as shown in the following example. You can run the command in online and offline modes.

```
# Online
props = {}
props['configuration'] = 'foo'
otd_deleteConfiguration(props)

# Offline
readDomain('/export/2110_12c/iplanet/ias/server/work/TD_Linux2.6_DBG.OBJ/domains/
otd_domain')
props = {}
props['configuration'] = 'foo'
otd_deleteConfiguration(props)
updateDomain()
closeDomain()
```

6

Managing Instances

An instance is an Oracle Traffic Director server running on a Node Manager, or on the server, and listening on one or more ports for requests from clients. This chapter contains the following sections:

- [Creating Oracle Traffic Director Instances](#)
- [Viewing a List of Oracle Traffic Director Instances](#)
- [Starting, Stopping, and Restarting Oracle Traffic Director Instances](#)
- [Updating Oracle Traffic Director Instances Without Restarting](#)
- [Deleting Oracle Traffic Director Instances](#)
- [Controlling Oracle Traffic Director Instances Through Scheduled Events](#)

Creating Oracle Traffic Director Instances

After creating a configuration, you can create Oracle Traffic Director server instances by deploying the configuration on one or more hosts.

You can create Oracle Traffic Director instances of a configuration by using either Fusion Middleware Control or the WLST as described in the following topics:

 **Note:**

Ensure that you have defined a configuration before you create an instance. See [Creating an Oracle Traffic Director Configuration](#)

Topics

- [Creating Oracle Traffic Director Instances Using Fusion Middleware Control](#)
- [Creating Oracle Traffic Director Instance Using WLST](#)

Creating Oracle Traffic Director Instances Using Fusion Middleware Control

To create Oracle Traffic Director instances of a configuration by using the Fusion Middleware Control, do the following:

1. Log in to Fusion Middleware Control for Traffic Director, as described in [Displaying Fusion Middleware Control](#).
2. Click the WebLogic Domain button at the upper left corner of the page.
3. Select Administration > OTD Configurations.
A list of the available configurations is displayed.

4. Select the configuration for which you want to create an instance.
5. Click the Traffic Director Configuration In the Common Tasks pane.
6. Select Administration > Instances.
The Instances page is displayed
7. In the Common Tasks pane, click **Create**.
The New Instance wizard is displayed.
8. Select the check boxes corresponding to the administration nodes on which you want to create instances of the configuration. Then, click **OK**.
A message is displayed confirming the successful creation of the instance. The Instances page is displayed, showing the instance that you just created.

Creating Oracle Traffic Director Instance Using WLST

To create one or more Oracle Traffic Director instances, run the `otd_createInstance` command. You can run the command in online and offline modes.



Note:

On Microsoft Windows, only a single domain with Oracle Traffic Director instance is allowed. However, there can be multiple domains without Oracle Traffic Director instances.

In the examples, the `otd_createInstance` creates an instance of the configuration named `foo` on machine `machine1`.

```
# Online
props = {}
props['configuration'] = 'foo'
props['machine'] = 'machine1'
otd_createInstance(props)

# Offline
readDomain('/export/2110_12c/iplanet/ias/server/work/TD_Linux2.6_DBG.OBJ/domains/
otd_domain')
props = {}
props['configuration'] = 'foo'
props['machine'] = 'machine1'
otd_createInstance(props)
updateDomain()
closeDomain()
```

Viewing a List of Oracle Traffic Director Instances

After creating Oracle Traffic Director server instances, you can view the current state of each instance. To view a list of the Oracle Traffic Director instances of a configuration, run the `otd_listInstances` command.

You can view a list of Oracle Traffic Director instances by using either Fusion Middleware Control or the WLST as described in the following topics:

Topics

- [Viewing a List of Oracle Traffic Director Instances Using Fusion Middleware Control](#)
- [Viewing a List of Oracle Traffic Director Instances Using WLST](#)

Viewing a List of Oracle Traffic Director Instances Using Fusion Middleware Control

To view a list of the Oracle Traffic Director instances of a configuration by using the Fusion Middleware Control, do the following:

1. Log in to Fusion Middleware Control for Traffic Director, as described in [Graphical User Interface-Fusion Middleware Control](#).
2. Click the WebLogic Domain button at the upper left corner of the page.
3. Select Administration > OTD Configurations.
A list of the available configurations is displayed.
4. Select the configuration for which you want to view instance.
5. Click the Traffic Director Configuration In the Common Tasks pane.
6. Select Administration > Instances.

The Instances page is displayed, showing the instances of the configuration.

You can view the properties of an instance by clicking on its name.

Viewing a List of Oracle Traffic Director Instances Using WLST

To view a list of the Oracle Traffic Director instances of a configuration, run the `otd_listInstances` command, as shown in the following example. You can run this command in online and offline modes.

```
# Online
props = {}
props['configuration'] = 'foo'
otd_listInstances(props)

# Offline
readDomain('/export/2110_12c/iplanet/ias/server/work/TD_Linux2.6_DBG.OBJ/domains/otd_domain')
props = {}
props['configuration'] = 'foo'
otd_listInstances(props)
closeDomain()
```

Starting, Stopping, and Restarting Oracle Traffic Director Instances

After creating a configuration, you can create start or stop them. To start, stop, or restart one or more Oracle Traffic Director instances of a configuration, run the `start`, `shutdown`, or `softRestart` command.

You can start, stop or restart configurations by using either Fusion Middleware Control or WLST as described in the following topics:

Topics

- [Starting, Stopping, and Restarting Oracle Traffic Director Instances Using Fusion Middleware Control](#)
- [Starting, Stopping, and Restarting Oracle Traffic Director Instances Using WLST](#)

Starting, Stopping, and Restarting Oracle Traffic Director Instances Using Fusion Middleware Control

To start, stop, or restart Oracle Traffic Director instances by using the Fusion Middleware Control, do the following:

1. Log in to Fusion Middleware Control for Traffic Director, as described in [Displaying Fusion Middleware Control](#).
2. Click the WebLogic Domain button at the upper left corner of the page.
3. Select Administration > OTD Configurations.

A list of the available configurations is displayed.

4. Select the configuration for which you want to start, stop, or restart instances.
5. Click the Traffic Director Configuration In the Common Tasks pane.
6. Select Administration > Instances.

The Instances page is displayed

7. Select on the list of instances available.

Click the Start Instances, Stop Instances, or Restart Instances button, as required, for the instance that you want to start, stop, or restart.

Starting, Stopping, and Restarting Oracle Traffic Director Instances Using WLST

To start, stop, or restart one or more Oracle Traffic Director instances of a configuration, run the `start`, `shutdown`, or `softRestart` command.

For example, the following three commands start, restart, and stop the instance the instance on the machine `otd_foo_machine1`.

```
start('otd_foo_machine1')
```

```
shutdown('otd_foo_machine1')
```

```
softRestart('otd_foo_machine1')
```

Updating Oracle Traffic Director Instances Without Restarting

When you make changes to some configuration parameters, the running Oracle Traffic Director instances of the configuration need not be restarted for the changes in the configuration to take effect. You can dynamically reconfigure the Oracle Traffic Director instances to reflect the new configuration.

Only dynamically reconfigurable changes in the configuration take effect. Changes in the `user`, `temp-path`, `log`, `thread-pool`, `pkcs11`, `stats`, `dns`, `dns-cache`, `ssl-session-cache`, and `access-log-buffer` settings remain the same after a reconfiguration procedure is completed. A `restart-required` exception is thrown if there are any such changes that require restart when a reconfiguration is done.

For a list of the parameters that support dynamic reconfiguration, see [Dynamic Reconfiguration](#) in the Configuration File Reference for Oracle Traffic Director .

You can dynamically reconfigure the running instances of a configuration by using either Fusion Middleware Control or the WLST as described in the following topics:

Topics

- [Reconfiguring an Oracle Traffic Director Instance Using Fusion Middleware Control](#)
- [Reconfiguring Oracle Traffic Director Instances Using WLST](#)

Reconfiguring an Oracle Traffic Director Instance Using Fusion Middleware Control

To reconfigure an Oracle Traffic Director instance by using the Fusion Middleware Control, do the following:

1. Log in to Fusion Middleware Control for Traffic Director, as described in [Displaying Fusion Middleware Control](#).
2. Click the WebLogic Domain button at the upper left corner of the page.
3. Select Administration > OTD Configurations. A list of the available configurations is displayed.
4. Select the configuration for which you want to reconfigure instances.
5. Click the Traffic Director Configuration In the Common Tasks pane.
6. Select Administration > Instances. The Instances page is displayed
7. Select the **Instance** from the list available.
8. Click the **Reconfigure** button for the instance that you want to update dynamically.

A message is displayed in the Console Messages pane confirming that the instance was reconfigured.

Reconfiguring Oracle Traffic Director Instances Using WLST

To reconfigure instances of a configuration using WLST, run the `softrestart` command.

For example, the `softrestart` command reconfigures the instance on the machine `otd_foo_machine1`.

```
props = java.util.Properties()
props.setProperty("MODE", "RECONFIG")
softRestart('otd_foo_machine1', props=props)
```

Deleting Oracle Traffic Director Instances

You can delete Oracle Traffic Director instances that are no longer required. To delete Oracle Traffic Director instances of a configuration, run the `otd_deleteInstance` command

You can delete instances of a configuration by using either Fusion Middleware Control or the WLST as described in the following topics:

Topics

- [Deleting Oracle Traffic Director Instances Using Fusion Middleware Control](#)
- [Deleting Oracle Traffic Director Instances Using WLST](#)

Deleting Oracle Traffic Director Instances Using Fusion Middleware Control

Note:

To delete an instance that is part of a failover group, you should first remove the instance from the failover group. See [Managing Failover Groups](#).

To delete an Oracle Traffic Director instance by using the Fusion Middleware Control, do the following:

1. Log in to Fusion Middleware Control for Traffic Director, as described in [Displaying Fusion Middleware Control](#).
2. Click the WebLogic Domain button at the upper left corner of the page.
3. Select Administration > OTD Configurations.
A list of the available configurations is displayed.
4. Select the configuration for which you want to delete instances.
5. Click the Traffic Director Configuration In the Common Tasks pane.
6. Select Administration > Instances.

The Instances page is displayed.

7. Select the **Instance** from the list available.
8. Click the **Delete** button for the instance that you want to delete.

A message is displayed in the Console Messages pane confirming that the instance was deleted.

Deleting Oracle Traffic Director Instances Using WLST

To delete Oracle Traffic Director instances of a configuration, run the `otd_deleteInstance` command. You can run this command in online and offline modes.

For example, the following command deletes the instance of the configuration:

```
# Online
props = {}
props['configuration'] = 'foo'
props['instance'] = 'otd_foo_machine1'
otd_deleteInstance(props)

# Offline
readDomain('/export/2110_12c/iplanet/ias/server/work/TD_Linux2.6_DBG.OBJ/domains/otd_domain')
props = {}
props['configuration'] = 'foo'
props['instance'] = 'otd_foo_machine1'
otd_deleteInstance(props)
updateDomain()
closeDomain()
```

Controlling Oracle Traffic Director Instances Through Scheduled Events

If you have to manage a large number of configurations and their instances you can schedule events for tasks to be performed automatically at defined intervals; or on specific days of the week, times of the day, or dates of the month.

You can create and manage events by using either Fusion Middleware Control or the WLST as described in the following topics:

Topics

- [Managing Events Using Fusion Middleware Control](#)
- [Managing Events Using WLST](#)

Managing Events Using Fusion Middleware Control

To create and manage events by using the Fusion Middleware Control, do the following:

1. Log in to Fusion Middleware Control, as described in [Graphical User Interface-Fusion Middleware Control](#).
2. Click the WebLogic Domain button at the upper left corner of the page.

3. Select Administration > OTD Configurations. A list of the available configurations is displayed.
4. Select the configuration for which you want to do schedule events.
5. Click the Traffic Director Configuration In the Common Tasks pane.
6. Select Advanced Configurations > Scheduled Events.

The Scheduled Events page is displayed.

7. Scroll down to the Scheduled Events section of the page.

The events that are currently scheduled for the configuration are listed.

- To enable or disable an event, select the **Enable/Disable** check box.
- To delete an event, click the **Delete** icon.
- To create an event, click **New Event**.

The New Configuration Event dialog box is displayed.

8. Select the event that you want to schedule, and specify the interval or time at which the event should be performed, and then click **OK**.

A message, confirming the change, is displayed in the Console Messages pane. In addition, the **Deployment Pending** message is displayed at the top of the main pane. You can either deploy the updated configuration immediately by clicking **Deploy Changes**, or you can do so later after making further changes as described in [Activating Configuration Changes](#).

Managing Events Using WLST

- **Creating an event**

To create an event, run the `otd_createEvent` command, as shown in the following examples.

```
props = {}
props['configuration'] = 'foo'
props['event'] = 'event-1'
props['command'] = 'bar'
props['time'] = '12:00'
otd_createEvent(props)
```

The first command schedules an event to perform the command 'bar' at 12:00pm.

Note:

For the scheduled events to take effect, you should redeploy the configuration.

- **Viewing a list of events**

To view a list of scheduled events, run the `otd_listEvents` command.

For example, to display the events scheduled for instances of the configuration:

```
props = {}  
props['configuration'] = 'foo'  
otd_listEvents(props)
```

- **Disabling an event**

When you create an event, it is enabled automatically:

The command 'otd_setEventProperties' with 'enabled' as 'false' can be used to disable the event

To disable an event, set the enabled property to false:

```
props = {}  
props['configuration'] = 'foo'  
props['event'] = 'bar'  
props['enabled'] = 'false'  
otd_setEventProperties(props)
```

- **Enabling an event**

The command 'otd_setEventProperties' with 'enabled' as 'true' must be used to enable the event

To enable an event, set the enabled property to true:

```
props = {}  
props['configuration'] = 'foo'  
props['event'] = 'event-1'  
props['enabled'] = 'true'  
otd_setEventProperties(props)
```

- **Deleting an event**

To delete an event, run the otd_deleteEvent command:

```
props = {}  
props['configuration'] = 'foo'  
props['event'] = 'event-1'  
otd_deleteEvent(props)
```

7

Managing Origin-Server Pools

You can define origin-server pools in a configuration, and then configure each virtual server in an Oracle Traffic Director instance to route client requests to a specific pool.

An *origin server* is a back-end server to which Oracle Traffic Director forwards requests that it receives from clients, and from which it receives responses to client requests. The origin servers could, for example, be Oracle WebLogic Server instances or Oracle iPlanet Web Server instances. A group of origin servers providing the same service or serving the same content is called an *origin-server pool*.

This chapter describes how create and manage several such origin-server pools. It contains the following sections:

- [Creating an Origin-Server Pool](#)
- [Viewing a List of Origin-Server Pools](#)
- [Modifying an Origin-Server Pool](#)
- [Deleting an Origin-Server Pool](#)
- [Configuring an Oracle WebLogic Server Cluster as an Origin-Server Pool](#)
- [Configuring a Custom Maintenance Page](#)
- [Configuring Health-Check Settings for Origin-Server Pools](#)

Creating an Origin-Server Pool

You can define origin-server pools in a configuration, and then configure each virtual server in an Oracle Traffic Director instance to route client requests to a specific pool.

Before You Begin

Before you begin creating an origin-server pool, decide the following:

- A unique name for the origin-server pool. Choose the name carefully; after creating an origin-server pool, you cannot change its name.
- `host:port` combinations for the servers in the origin-server pool.

Note:

If the origin servers for which you want to create a pool are Oracle WebLogic Server managed servers in a cluster, it is sufficient to create the pool with any one of the managed servers as the origin server. You can then configure Oracle Traffic Director to discover the other managed servers in the pool dynamically. See [Configuring an Oracle WebLogic Server Cluster as an Origin-Server Pool](#).

- The communication protocol—HTTP, HTTPS or TCP—of the servers in the pool.

- The address family that the servers in the origin-server pool use to listen for requests.

The supported address families are:

- `inet` (IPv4)
- `inet6` (IPv6)
- `inet-sdp` (Sockets Direct Protocol): Select this family if the servers in the origin-server pool are on the InfiniBand fabric and listen on an SDP interface, such as Oracle WebLogic Servers deployed on Oracle Exalogic machines.



Note:

When you create an origin-server pool, you are, in effect, modifying a configuration. So for the settings of the new origin-server pool to take effect in the Oracle Traffic Director instances, you should redeploy the configuration as described in [Activate Configuration Changes](#).

You can create an origin-server pool by using either Fusion Middleware Control or WLST as described in the following topics:

Topics

- [Creating an Origin-Server Pool Using Fusion Middleware Control](#)
- [Creating an Origin-Server Pool Using WLST](#)

Creating an Origin-Server Pool Using Fusion Middleware Control

To create an origin-server pool by using the Fusion Middleware Control, do the following:

1. Log in to Fusion Middleware Control, as described in [Graphical User Interface-Fusion Middleware Control](#).
2. Click the WebLogic Domain button at the upper left corner of the page.
3. Select Administration > OTD Configurations.
A list of the available configurations is displayed.
4. Select the configuration for which you want to Origin-Server Pool.
5. Click the Traffic Director Configuration In the Common Tasks pane.
6. Select Administration > Server Pools.
The Server Pools page is displayed. It shows a list of the server pools (HTTP/S and TCP server pools) defined for the configuration.
7. Select the Server Pool for which you want to configure.
8. In the Common Tasks pane, click **Create** button.
The Create Origin-Server Pool page is displayed.
9. Follow the on-screen prompts to complete creation of the origin-server pool by using the details—name, type, and so on—that you decided earlier.

After the origin-server pool is defined, Click OK on the right top of the screen. The results screen of the New Origin-Server Pool displays a message confirming successful creation of the origin-server pool.

10. The details of the origin-server pool that you just created are displayed on the Origin-Server Pools page.

In addition, the **Deployment Pending** message is displayed at the top of the main pane. You can either deploy the updated configuration immediately by clicking **Deploy Changes**, or you can do so later after making further changes as described in [Activating Configuration Changes](#).

Creating an Origin-Server Pool Using WLST

To create an origin-server pool, run the `otd_createOriginServerPool` command.

For example, the following command creates an origin-server pool `origin-server-pool-1` containing origin server `www.example.com:12345` in the configuration `foo`.

```
props = {}
props['configuration'] = 'foo'
props['origin-server-pool'] = 'origin-server-pool-1'
props['origin-server'] = 'www.example.com:12345'
otd_createOriginServerPool(props)
```

Specifying an HTTP Forward Proxy Server

The `otd_createOriginServerPool` command takes `proxy-server` as an optional option which you can use to specify a HTTP forward proxy server to be associated with an origin server pool so that all member origin servers of the pool are communicated with via the configured HTTP forward proxy server. The type must be `http` or `https`.

For example:

```
props = {}
props['configuration'] = 'foo'
props['origin-server-pool'] = 'origin-server-pool-1'
props['origin-server'] = 'www.example.com:12345'
props['type'] = 'http'
props['proxy-server'] = 'proxy.example.com:12345'
otd_createOriginServerPool(props)
```

Viewing a List of Origin-Server Pools

After creating origin-server pool in a configuration, you can view a list of Oracle Traffic Director instances associated to origin-server pools. To view a list of origin-server pools, run the `otd_listOriginServerPool` command.

You can view the list of origin-server pools by using either Fusion Middleware Control or WLST as described in the following topics:

Topics

- [Viewing a List of Origin-Server Pools Using Fusion Middleware Control](#)
- [Viewing a List of Origin-Server Pools Using WLST](#)

Viewing a List of Origin-Server Pools Using Fusion Middleware Control

To view a list of origin-server pools by using the Fusion Middleware Control, do the following:

1. Log in to Fusion Middleware Control, as described in [Graphical User Interface-Fusion Middleware Control](#).
2. Click the WebLogic Domain button at the upper left corner of the page.
3. Select Administration > OTD Configurations.
A list of the available configurations is displayed.
4. Select the configuration for which you want to view Origin-Server Pools.
5. Click the Traffic Director Configuration In the Common Tasks pane.
6. Select Administration > Server Pools.
The Server Pools page is displayed.
7. It shows a list of the origin-server pools defined for that configuration.

Viewing a List of Origin-Server Pools Using WLST

To view a list of origin-server pools, run the `otd_listOriginServerPools` command, as shown in the following example:

```
props = {}  
props['configuration'] = 'foo'  
otd_listOriginServerPools(props)
```

You can view the general properties and health-check settings of an origin-server pool by running the `otd_getOriginServerPoolProperties` and `otd_getHealthCheckProperties` commands respectively.

Modifying an Origin-Server Pool

After you create an origin-server pool, you might need to change some of the settings such as network protocol, proxy server, load balancing method and so on.

You can modify a configuration by using either Fusion Middleware Control or the WLST as described in the following topics:

Note:

When you modify an origin-server pool, you are, in effect, modifying a configuration. So for the updated origin-server pool settings to take effect in the Oracle Traffic Director instances, you should redeploy the configuration as described in [Activate Configuration Changes](#).

Topics

- [Changing the Properties of an Origin-Server Pool Using Fusion Middleware Control](#)
- [Changing the Properties of an Origin-Server Pool Using WLST](#)

Changing the Properties of an Origin-Server Pool Using Fusion Middleware Control

To change the properties of an origin-server pool by using the Fusion Middleware Control, do the following:

1. Log in to Fusion Middleware Control, as described in [Displaying Fusion Middleware Control](#).
2. Click the WebLogic Domain button at the upper left corner of the page.
3. Select Administration > OTD Configurations.
A list of the available configurations is displayed.
4. Select the configuration for which you want to modify Origin-Server Pools.
5. Click the Traffic Director Configuration In the Common Tasks pane.
6. Select Administration > Server Pools.
The Server Pools page is displayed.
7. It shows a list of the origin-server pools that are defined for that configuration.
8. Click the name of the origin-server pool that you want to modify. Click the Edit button in the common task pane

The Origin Server Pool Settings page is displayed. On this page, you can do the following:

- Change the network protocol—IPv4, IPv6, or SDP—for the servers in the pool.
- Set a proxy server via the **Connect to Origin Servers via Proxy Server** section. This setting specifies a HTTP forward proxy server to be associated with an origin server pool so that all member origin servers of the pool are communicated with via the configured HTTP forward proxy server.
- Change the load-balancing method that Oracle Traffic Director should use to distribute client requests to the pool.
 - **Least connection count** (default): When processing a request, Oracle Traffic Director assesses the number of connections that are currently active for each origin server, and forwards the request to the origin server with the least number of active connections.

The least connection count method works on the premise that origin servers that are faster have fewer active connections, and so can take on more load. To further adjust the load distribution based on the capacities of the origin servers, you can assign relative weights to the origin servers.

 **Note:**

WebSocket connections affect the least connection count load balancing algorithm because WebSocket connections are potentially long lasting and will be counted as active connections until they are closed.

- **Least response time:** Though least connection count works well on most workloads, there could be situations when the response time of origin servers in a given pool for the same amount of load could differ. For example:

- When origin servers of a given pool are deployed on machines that differ in hardware specification.

- When some origin server nodes are used for other services.

- When network connectivity for different nodes is not uniform or some network interfaces are more loaded than others.

Least response time is useful in such scenarios because it is a dynamic weighted least connection algorithm and it calculates weights based on the response time. These weights are continuously adjusted based on how the origin servers respond. Least response time helps you avoid manual tuning of weights in the least connection algorithm.

- **Round robin:** Oracle Traffic Director forwards requests sequentially to the available origin servers—the first request to the first origin server in the pool, the second request to the next origin server, and so on. After it sends a request to the last origin server in the pool, it starts again with the first origin server.

Though the round-robin method is simple, predictable, and low on processing overhead, it ignores differences in the origin servers' capabilities. So, over time, requests can accumulate at origin servers that are significantly slow. To overcome this problem, you can use a *weighted* round-robin method, by assigning relative weights to the origin servers.

- **IP Hash:** All the incoming requests from the same client IP address should go to the same content origination server. This load balancing policy is especially useful in the context of TCP Load Balancing, Oracle Traffic Director suggests customers to make use of this load balancing policy.

See [Modifying an Origin Server](#) information about assigning weights to origin servers.

- Configure health-check settings. See [Configuring Health-Check Settings for Origin-Server](#).
- Specify whether Oracle Traffic Director should dynamically discover Oracle WebLogic Server managed servers in a cluster. See [Configuring an Oracle WebLogic Server Cluster as an Origin-Server Pool](#).

 **Note:**

You can add, modify, and remove origin servers in the pool, by selecting **Origin Servers** in the navigation pane. See [Managing Origin Servers](#).

9. Specify the parameters that you want to change.

On-screen help and prompts are provided for all of the parameters.

When you change the value in a field or tab out of a text field that you changed, the **Save** button near the upper right corner of the page is enabled.

At any time, you can discard the changes by clicking the **Cancel** button.

10. After making the required changes, click **OK**.

- A message, confirming that the updated configuration was saved, is displayed in the Console Messages pane.
- In addition, the **Deployment Pending** message is displayed at the top of the main pane. You can either deploy the updated configuration immediately by clicking **Deploy Changes**, or you can do so later after making further changes as described in [Activate Configuration Changes](#).

Changing the Properties of an Origin-Server Pool Using WLST

- To change the network protocol and load-balancing method for an origin-server pool, run the `otd_setOriginServerPoolProperties` command.

For example, the following command changes the load-balancing method for the origin-server pool `origin-server-pool-1` in the configuration `foo` to the least connection count method.

```
props = {}
props['configuration'] = 'foo'
props['origin-server-pool'] = 'origin-server-pool-1'
props['load-distribution'] = 'least-connection-count'
otd_setOriginServerPoolProperties(props)
```

- To change the health-check parameters for an origin-server pool, run the `otd_setHealthCheckProperties` command.

For example, the following command changes the size of the response body for servers in the origin-server pool `origin-server-pool-1` of the configuration `foo` to 4096 bytes.

```
props = {}
props['configuration'] = 'foo'
props['origin-server-pool'] = 'origin-server-pool-1'
props['response-body-match-size'] = '4096'
otd_setHealthCheckProperties(props)
```

Deleting an Origin-Server Pool

You can delete origin-server pools that are no longer required. To delete origin-server pool instances of a configuration, run the `otd_deleteOriginServerPool` command.

 **Note:**

- You cannot delete an origin-server pool that is associated with one or more routes in virtual servers.
To delete an origin-server pool that is associated with routes, you must first delete the referring routes, as described in [Configuring Routes](#).
- When you delete an origin-server pool, you are, in effect, modifying a configuration. So for the updated configuration to take effect in the Oracle Traffic Director instances, you should redeploy the configuration as described in [Activating Configuration Changes](#).

You can delete an origin-server pool by using either Fusion Middleware Control or the WLST as described in the following topics:

Topics

- [Deleting an Origin-Server Pool Using Fusion Middleware Control](#)
- [Deleting an Origin-Server Pool Using WLST](#)

Deleting an Origin-Server Pool Using Fusion Middleware Control

To delete an origin-server pool by using the Fusion Middleware Control, do the following:

1. Log in to Fusion Middleware Control, as described in [Displaying Fusion Middleware Control](#).
2. Click the WebLogic Domain button at the upper left corner of the page.
3. Select Administration > OTD Configurations.
A list of the available configurations is displayed.
4. Select the configuration for which you want to delete Origin-Server Pools.
5. Click the Traffic Director Configuration In the Common Tasks pane.
6. Select Administration > Server Pools.
The Server Pools page is displayed.
7. It shows a list of the origin-server pools that are defined for that configuration.
8. Select the pool which you want to delete from the list available.
9. Click the **Delete** button in the common task pane.
 - If the origin-server pool is associated with one or more routes in virtual servers, a message is displayed indicating that you cannot delete the pool.
 - If the origin-server pool is not associated with any virtual server, a prompt to confirm the deletion is displayed.
10. Click **Yes**.
The origin-server pool is deleted.

Deleting an Origin-Server Pool Using WLST

To delete an origin-server pool, run the `otd_deleteOriginServerPool` command, as shown in the following example:

```
props = {}
props['configuration'] = 'foo'
props['origin-server-pool'] = 'origin-server-pool-1'
otd_deleteOriginServerPool(props)
```

Configuring an Oracle WebLogic Server Cluster as an Origin-Server Pool

You can configure Oracle Traffic Director to *discover* the presence of other Oracle WebLogic Server instances in the cluster dynamically, and distribute client requests to the managed server that is configured as an origin server *and* to the dynamically discovered managed servers in the same cluster.

If you want to create an origin-server pool that represents a cluster of Oracle WebLogic Server managed servers, you need not specify each managed server in the cluster as an origin server. It is sufficient to specify *any one* of the managed servers as the sole origin server in the pool.

So when dynamic discovery is enabled, if any of the managed servers in the cluster is stopped, added, or removed, you need not update the definition of the origin-server pool. However, for detecting changes in the Oracle WebLogic Server cluster, Oracle Traffic Director sends health-check requests at a specified interval, which causes some overhead.

Note:

Oracle Traffic Director has built-in support for some common functionality offered by the WebLogic Server plug-in. Hence Oracle Traffic Director does not require any other plug-in to inter-operate with WebLogic Server.

- [How Dynamic Discovery Works](#)
- [Enabling Dynamic Discovery](#)

How Dynamic Discovery Works

When dynamic discovery is enabled for an origin-server pool, Oracle Traffic Director discovers the remaining Oracle WebLogic Server managed servers in the cluster, by doing the following:

1. **When an Oracle Traffic Director instance starts**, it checks whether the origin servers specified in the pool are Oracle WebLogic Server managed servers and whether the servers belong to a cluster, by sending an HTTP health-check request to each configured origin server.

The origin server's response indicates whether the server is an Oracle WebLogic Server managed server. If the origin server is an Oracle WebLogic Server managed server that belongs to a cluster, the response also includes a list of the managed servers in the cluster.

2. Oracle Traffic Director uses the information in the response from the origin server to update the configuration with the discovered managed servers.

The dynamically discovered origin servers inherit all of the properties—weight, maximum connections, and so on—that are specified for the configured origin server.

3. **Subsequently, at each health-check interval (default: 30 seconds) configured for the origin-server pool**, Oracle Traffic Director attempts to detect changes in the cluster, by sending dynamic-discovery health-check requests to the Oracle WebLogic Server instances that are configured as origin servers in the pool.

If the response indicates a change—removal or addition of a managed server—in the cluster since the previous health check, Oracle Traffic Director updates the configuration with the new set of dynamically discovered origin servers.

 **Note:**

- Dynamically discovered origin servers are not stored permanently in the origin-server pool definition of the instance's configuration. So when you restart an Oracle Traffic Director instance, the process of dynamic discovery starts afresh.
- The HTTP request type that Oracle Traffic Director sends for dynamic discovery is the health-check request type that is currently configured for the origin-server pool—`OPTIONS` (default) or `GET`. See [Configuring Health-Check Settings for Origin-Server Pools](#).

Enabling Dynamic Discovery

When you create an origin-server pool, dynamic discovery of Oracle WebLogic Server managed servers in a cluster is *not* enabled by default. You can enable dynamic discovery by using either Fusion Middleware Control or the WLST.

 **Note:**

When you modify an origin-server pool, you are, in effect, modifying a configuration. So for the updated origin-server pool settings to take effect in the Oracle Traffic Director instances, you must redeploy the configuration as described in [Activating Configuration Changes](#).

Enabling Dynamic Discovery Using Fusion Middleware Control

To enable dynamic discovery of WebLogic Server managed servers in a cluster by using the Fusion Middleware Control, do the following:

1. Log in to Fusion Middleware Control, as described in [Displaying Fusion Middleware Control](#).
2. Click the WebLogic Domain button at the upper left corner of the page.
3. Select Administration > OTD Configurations.
A list of the available configurations is displayed.
4. Select the configuration for which you want to enable dynamic discovery.
5. Click the Traffic Director Configuration In the Common Tasks pane.
6. Select Administration > Server Pools.
The Server Pools page is displayed.
7. It shows a list of the origin-server pools that are defined for that configuration.
8. Select the pool which you want to enable dynamic discovery from the list available.
9. Go to the **Advanced Settings** section of the page.
10. Under the Health Check subsection, make sure that the **Protocol** is HTTP, select the **Dynamic Discovery** check box.
11. Click **OK** button on the top right corner of the window.

**Note:**

If the current health-check protocol is TCP, an error message is displayed indicating that the protocol must be changed to HTTP in order to enable dynamic discovery.

A message is displayed in the Console Message pane confirming that the updated health-check settings were saved.

Enabling Dynamic Discovery Using WLST

To enable dynamic discovery of Oracle WebLogic Server managed servers in a cluster, run the `otd_setHealthCheckProperties` command.

For example, the following command enables dynamic discovery of managed servers in the WebLogic Server cluster that the `origin-server-pool-1` origin-server pool represents.

```
props = {}
props['configuration'] = 'foo'
props['origin-server-pool'] = 'origin-server-pool-1'
props['dynamic-server-discovery'] = '4096'
otd_setHealthCheckProperties(props)
```

**Note:**

If TCP is the current health-check protocol, an error message is displayed indicating that the protocol must be changed to HTTP in order to enable dynamic discovery.

Configuring a Custom Maintenance Page

You can configure custom maintenance pages in Oracle Traffic Director to serve a custom response code, and HTML page, when back-end servers maintenance is required.

When maintenance is enabled for an origin server pool, then:

- All the requests to Oracle Traffic Director, are aborted with a 503 response code, if both response-code and response-file are not configured.
- All the requests to Oracle Traffic Director, are aborted with response-code value as the response code, if only response-code is specified.
- All the requests to Oracle Traffic Director, are not aborted, but are responded to with a response-file content and response-code value as the response code, if both are specified.
- Health-check is disabled on its origin servers.

When maintenance is not enabled for an origin server pool but no origin servers are configured or enabled, then:

- All the requests to Oracle Traffic Director, are aborted with a 503 response code.
- Health-check is disabled on its origin servers.

Monitoring of Statistics for Origin Server Pool in Maintenance

If the origin-server pool is in a maintenance state, there will be no statistics for the origin server pool and the origin servers. Statistics will be available only for active origin server pools and active origin servers.

Enabling or Disabling Maintenance for an Origin-Server Pool Using WLST

To enable maintenance for an origin-server pool, run the `otd_enableOriginServerPoolMaintenance` command.

For example, the following command enables maintenance for the `origin-server-pool-1` origin-server pool, and specifies a response-code of 503. This command takes `response-code` and `response-file` as optional properties. A response-code of 200 is not allowed without a response-file.

```
props = {}
props['configuration'] = 'foo'
props['origin-server-pool'] = 'origin-server-pool-1'
props['response-code'] = '503'
otd_enableOriginServerPoolMaintenance(props)
```

To disable maintenance, use the `otd_disableOriginServerPoolMaintenance` command:

```
props = {}
props['configuration'] = 'foo'
props['origin-server-pool'] = 'origin-server-pool-1'
otd_disableOriginServerPoolMaintenance(props)
```

To return the enabled, response-file and response-code properties for the origin-server pool, use the `otd_getOriginServerPoolMaintenanceProperties` command:


```
props = {}  
props['configuration'] = 'foo'  
props['origin-server-pool'] = 'origin-server-pool-1'  
otd_getOriginServerPoolMaintenanceProperties(props)
```

Configuring Health-Check Settings for Origin-Server Pools

To ensure that requests are distributed to only those origin servers that are available and can receive requests, Oracle Traffic Director monitors the availability and health of origin servers by sending health-check requests to all of the origin servers in a pool.

You can configure health-check parameters for an origin-server pool by using either Fusion Middleware Control or the WLST.

Note:

When you configure health-check settings for an origin-server pool, you are, in effect, modifying a configuration. So for the updated configuration to take effect in the Oracle Traffic Director instances, you should redeploy the configuration as described in [Activating Configuration Changes](#).

When Does Oracle Traffic Director Send Health-Check Requests?

When an Oracle Traffic Director instance starts, it performs an initial health check for all the origin servers in all of the configured origin-server pools.

If the initial health check indicates that an origin server is healthy, Oracle Traffic Director sends further health-check requests to an origin server only in the following situations:

- The server has not served any request successfully for the entire duration of the previous health-check interval.
- Dynamic discovery is enabled for this origin server pool. See [Configuring an Oracle WebLogic Server Cluster as an Origin-Server Pool](#).

If a health check—either initial or subsequent—indicates that an origin server is not available, Oracle Traffic Director repeats the health check at the specified health-check interval.

When Is an Origin Server Considered Available and Healthy?

If the configured health-check connection type is TCP, an origin server is considered available if the connection is successfully established, indicating that the server is actively listening on its service port.

If the configured health-check connection type is HTTP, an origin server is considered available and health when all of the following conditions are fulfilled:

- There is no error while sending the HTTP request.
- The response is received before timeout period is reached.
- The status code in the response matches any of the acceptable response codes, if specified.

By default, Oracle Traffic Director accepts response codes from 1xx to 4xx as indicators of a healthy origin server.

- The response body matches the acceptable response body, if specified.

Configuring Health-Check Settings for Origin Servers Using the Fusion Middleware Control

To view and change health-check settings origin servers in a pool by using the Fusion Middleware Control, do the following:

1. Log in to the Fusion Middleware Control, as described in [Displaying Fusion Middleware Control](#).
2. Click the **Configurations** button that is situated at the upper left corner of the page.
A list of the available configurations is displayed.
3. Select the configuration for which you want to view or change origin-server health-check settings.
4. In the navigation pane, expand **Origin-Server Pools**, and select the origin-server pool for which you want to view or change health-check settings.
The Origin-Server Pools page is displayed. It shows a list of the origin-server pools that are defined for the configuration.
5. Click the name of the origin-server pool that you want to modify.
The Server Pool Settings page is displayed.
6. Go to the **Advanced Settings** section of the page.
7. Specify the parameters that you want to change.
On-screen help and prompts are provided for all of the parameters.
When you change the value in a field or tab out of a text field that you changed, the **Save** button near the upper right corner of the page is enabled.
At any time, you can discard the changes by clicking the **Reset** button.
8. After making the required changes, click **Save**.

Configuring Health-Check Settings for Origin Servers Using WLST

- To view the current health-check settings for an origin-server pool in a configuration, run the `otd_getHealthCheckProperties` command, as shown in the following example:

```
props = {}  
props['configuration'] = 'foo'  
props['origin-server-pool'] = 'origin-server-pool-1'  
otd_getHealthCheckProperties(props)
```

```
protocol=HTTP  
interval=30  
timeout=5  
failover-threshold=3  
request-method=OPTIONS  
request-uri=/  

```

```
response-body-match-size=2048  
dynamic-server-discovery=false
```

- To change the health-check settings for an origin-server pool in a configuration, run the `otd_setHealthCheckProperties` command.

For example, the following command changes the health-check interval to 60 seconds and the health-check timeout period to 10 seconds for the origin-server pool `origin-server-pool-1` in the configuration `foo`.

```
props = {  
  props['configuration'] = 'foo'  
  props['origin-server-pool'] = 'origin-server-pool-1'  
  props['interval'] = '60'  
  props['timeout'] = '10'  
  otd_setHealthCheckProperties(props)
```

Using an External Health-Check Executable to Check the Health of a Server

Oracle Traffic Director supports a generic health check hook-up mechanism, so that you can write your own health check programs/scripts to monitor the health of specific origin servers. An external executable is especially useful for a protocol-level health check monitor for the origin servers.

If you configure Oracle Traffic Director to use an external executable to check the health of a server, Oracle Traffic Director periodically invokes the executable and passes certain parameters to it as arguments and environment variables. If the executable successfully returns a status code 0 before a timeout, Oracle Traffic Director sets the server's status to online. If the executable returns a value other than zero or a timeout occurs before the execution ends, Oracle Traffic Director immediately sets the server status to offline without retrying, and terminates the execution in the timeout case. There are different reasons why the executable could return a non-zero status code, including a core dump, signal termination, or the logic of external executable itself. Oracle Traffic Director marks the server offline whenever the return status is non-zero.

Also, Oracle Traffic Director captures the standard output and standard error from the executable and logs the messages into the event log (server log).

The external executable handles the actual health check jobs, including establishing connection to the origin server, sending/receiving request/response, dealing with SSL (if applicable), retry logic (if required), and so on. The executable is expected to exit with a status 0 after it finishes the health check operation and wants to set the server status to online. If the executable wants to have some messages logged in the event log, it should print those messages to standard output.

Configuring Health-Check Settings to Use an External Executable

To configure the health-check settings to use an external executable for an origin-server pool in a configuration, run the `otd_setHealthCheckProperties` command.

For example, the following command sets the health-check method to `command`, and specifies a path of `/path/myhcsript` for the external health-check executable. The interval, and timeout properties are also specified.

```

props = {}
props['configuration'] = 'foo'
props['origin-server-pool'] = 'origin-server-pool-1'
props['protocol'] = 'command'
props['interval'] = '60'
props['timeout'] = '10'
props['command'] = '/path/myhcscrip'
otd_setHealthCheckProperties(props)

```



Note:

In case of an HTTP type of origin server pool, the `COMMAND` health check protocol is not considered if:

- the origin server type is `UNDETECTED` or,
- the origin server type is `WLS` and dynamic discovery is set.

For the updated configuration to take effect, you should deploy it to the Oracle Traffic Director instances by using the `activate` command.

Parameters to the External Health Check Executable

Oracle Traffic Director passes parameters to the external health check executable in two ways. In particular, Oracle Traffic Director passes the origin server host, origin server port, and timeout value via arguments, and passes all the existing environment variables as well as `ORACLE_HOME`, `INSTANCE_HOME`, `INSTANCE_NAME`, `DOMAIN_HOME`, and `OTD_LOG_LEVEL` as environment variables. The argument parameters are passed in the format of command line options, as shown in the following example command:

```
/path/myhcscrip -h server1.myserver.com -p 389 -t 10
```

Where, `-h`, `-p`, and `-t` stand for host, port, and timeout respectively.

Table 7-1 Argument Parameters

Option	Meaning
<code>-h</code>	Origin server host.
<code>-p</code>	Origin server port.
<code>-t</code>	Health-check timeout.

You can pass other parameters to the external executable by specifying additional option arguments in the parameter `command`:

```
/path/myhcscrip --secure -d /dbpath
```

Correspondingly, Oracle Traffic Director passes those additional arguments to the external executable:

```
/path/myhcscrip --secure -d /dbpath -h server1.myserver.com -p 389 -t 10
```

Oracle Traffic Director does not automatically pass the origin server port type (for example, LDAP over SSL) to the executable. If the type information is needed in the executable, you can specify the type information in the command string as an additional argument (as shown in the example above) or have the type hard-coded or obtained from other resource (for example, its own configuration file or environment variable) in their health check program/script.

Furthermore, it is recommended that the external executable takes the timeout value into account and tries to complete execution and return status before timeout. If timeout occurs but execution is not complete, Oracle Traffic Director terminates the process and set the server status to offline.

Logging

Oracle Traffic Director passes the configured logging level to the external program via the environment variable `OTD_LOG_LEVEL`, and the value of the environment variable is an integer. In the external executable, you can customize the amount of logging messages based on the logging level. The following table defines the mapping between the Oracle Traffic Director logging levels and the argument values.

Table 7-2 Mapping Oracle Traffic Director Logging Levels and Argument Values

Value	Oracle Traffic Director Logging Level
0	NOTIFICATION:1 or higher
1	TRACE:1
2	TRACE:16
3	TRACE:32

Oracle Traffic Director logs contents in both standard output and the standard error of the external executable in a single log entry in the server log. If the exit status of the command health check script is 0, the messages are logged at TRACE:1 level. Otherwise, standard output is logged at NOTIFICATION:1 level and the standard error is logged at WARNING:1 level.

8

Managing Origin Servers

An *origin server* is a back-end server to which Oracle Traffic Director forwards requests that it receives from clients, and from which it receives responses to client requests. The origin servers could, for example, be Oracle WebLogic Server instances or Oracle iPlanet Web Server instances.

This chapter describes how to create and manage origin servers. It contains the following sections:

- [Adding an Origin Server to a Pool](#)
- [Viewing a List of Origin Servers](#)
- [Modifying an Origin Server](#)
- [Removing an Origin Server from a Pool](#)

Adding an Origin Server to a Pool

You can add an origin server to an origin-server pool by specifying the host name, IP address and the proportion of the total request load to be distributed to the origin server.

When you add an origin server to a pool, you are, in effect, modifying a configuration. So for the updated configuration to take effect in the Oracle Traffic Director instances, you should redeploy the configuration as described in [Activating Configuration Changes](#).

Before You Begin

Before you begin adding an origin server to a pool, decide the following:

- The origin-server pool to which you want to add the origin server.
- The **host** name or IP address of the origin server. It is recommended that the IP address that you provide is the InfiniBand interface IP address (IPoIB) or Socket Director Protocol (SDP) address.

Note:

SDP is a native Infiniband protocol. With SDP, performance is very specific to work load. Hence, it is important to evaluate and compare the performance with SDP and IPoIB, and then select the one that meets your requirement.

- The **port** number at which the origin server listens for requests.
- Whether the server is a **backup** origin server.

Oracle Traffic Director forwards requests to a backup origin server only when the health check indicates that none of the primary origin servers is available.

- The proportion of the total request load that Oracle Traffic Director should distribute to the origin server. You define this proportion as a **weight** number that is relative to the weights assigned to the other origin servers in the pool.

You can use weights to get Oracle Traffic Director to distribute the request load based on the relative capacities of the origin servers in a pool.

Consider a pool consisting of three origin servers—`os1`, `os2`, and `os3`, with the weights 1, 2, and 2 respectively. The total of the weights assigned to all the servers in the pool is $1+2+2=5$. Oracle Traffic Director distributes a fifth ($1/5$) of the total load to `os1`, and two-fifths ($2/5$) of the load to each of `os2` and `os3`.

You can add an origin server to an origin-server by using either Fusion Middleware Control or WLST as described in the following topics:

Topics

- [Adding an Origin Server to a Pool Using Fusion Middleware Control](#)
- [Adding an Origin Server to a Pool Using WLST](#)

Adding an Origin Server to a Pool Using Fusion Middleware Control

To add an origin server to a pool by using the Fusion Middleware Control, do the following:

1. Log in to Fusion Middleware Control, as described in [Graphical User Interface-Fusion Middleware Control](#).
2. Click the WebLogic Domain button at the upper left corner of the page.
3. Select Administration > OTD Configurations.
A list of the available configurations is displayed.
4. Select the configuration for which you want to add origin server.
5. Click the Traffic Director Configuration In the Common Tasks pane.
6. Select Administration > Server Pools.
The Server Pools page is displayed. It shows a list of the server pools (HTTP/S and TCP server pools) defined for the configuration.
7. Select the Server Pool for which you want to configure origin server.
8. In the Common Tasks pane, click **Configure Origin Server**.
9. Click **Create** button in the common task pan
The new Create Origin Server page appears.
10. Follow the on-screen prompts to complete creation of the origin-server pool by using the details—origin-server pool, host, port, and so on—that you decided earlier. Click **OK** button on right top corner of the page.
After the origin server is created, the Results screen of the New Origin Server wizard displays a message confirming successful creation of the origin server.
11. The details of the origin server that you just defined are displayed on the Origin Servers page.

Adding an Origin Server to a Pool Using WLST

To add an origin server to a pool, run the `otd_createOriginServer` command.

For example, the following command adds host `www.example.com` and port `12345` as the origin server in the pool `origin-server-pool-1` of the configuration `foo`.

```
props = {}
props['configuration'] = 'foo'
props['origin-server-pool'] = 'origin-server-pool-1'
props['host'] = 'www.example.com'
props['port'] = '12345'
otd_createOriginServer(props)
```

Viewing a List of Origin Servers

After adding an origin server to an origin-server pool, you can view the current list of each origin server. To view a list of origin servers, run the `otd_listOriginServers` command.

You can view a list of origin servers by using either Fusion Middleware Control or the WLST as described in the following topics:

Topics

- [Viewing a List of Origin Servers Using Fusion Middleware Control](#)
- [Viewing a List of Origin Servers Using WLST](#)

Viewing a List of Origin Servers Using Fusion Middleware Control

To view a list of origin servers by using the Fusion Middleware Control, do the following:

1. Log in to Fusion Middleware Control, as described in [Displaying Fusion Middleware Control](#).
2. Click the WebLogic Domain button at the upper left corner of the page.
3. Select Administration > OTD Configurations.

A list of the available configurations is displayed.

4. Select the configuration for which you want to view origin server.
5. Click the Traffic Director Configuration In the Common Tasks pane.
6. Select Administration > Server Pools.

The Server Pools page is displayed. It shows a list of the server pools (HTTP/S and TCP server pools) defined for the configuration.

7. Select the Server Pool for which you want to view origin server.
8. In the Common Tasks pane, click **Configure Origin Server**.
9. Select the Server Pool for which you want to view origin server.

Viewing a List of Origin Servers Using WLST

To view a list of origin servers defined in a pool, run the `otd_listOriginServers` command as shown in the following example:

```
props = {}
props['configuration'] = 'foo'
props['origin-server-pool'] = 'origin-server-pool-1'
otd_listOriginServers(props)
```

You can view the properties of an origin server in detail by running the `otd_getOriginServerProperties` command.

Modifying an Origin Server

After you add an origin server to a pool, you might need to change some of the settings such as host, port, or enable/disable the origin server. To modify the properties of an origin server, run the `otd_setOriginServerProperties` command.

This section describes how you can do the following:

- Change the properties—host, port, weight, and so on—that you defined while creating the origin server.
- Enable or disable the origin server.
- Specify the maximum number of connections that the origin server can handle concurrently.
- Specify the duration (ramp-up time) over which Oracle Traffic Director should increase the request-sending rate to the origin server. You can use this parameter to ensure that the request load, on origin servers that have just come up after being offline, is increased *gradually* up to the capacity of the server.



Note:

When you change the properties of an origin server in a pool, you are modifying a configuration. To see the updated configuration in the Oracle Traffic Director instances, you should redeploy the configuration as described in [Activating Configuration Changes](#).

You can modify the properties of an origin server by using either Fusion Middleware Control or the WLST as described in the following topics:

Topics

- [Modifying an Origin Server Using Fusion Middleware Control](#)
- [Changing the Properties of an Origin Server Using WLST](#)

Modifying an Origin Server Using Fusion Middleware Control

To change the properties of an origin server by using the Fusion Middleware Control, do the following:

1. Log in to Fusion Middleware Control, as described in [Graphical User Interface-Fusion Middleware Control](#).

2. Click the WebLogic Domain button at the upper left corner of the page.

3. Select Administration > OTD Configurations.

A list of the available configurations is displayed.

4. Select the configuration for which you want to modify origin server.

5. Click the Traffic Director Configuration In the Common Tasks pane.

6. Select Administration > Server Pools.

The Server Pools page is displayed. It shows a list of the server pools (HTTP/S and TCP server pools) defined for the configuration.

7. Select the Server Pool for which you want to modify origin server.

8. In the Common Tasks pane, click **Configure Origin Server**.

9. Select the Server Pool for which you want to modify origin server.

10. Click the name of the origin server that you want to modify.

The Editing Origin Server dialog box is displayed. In this dialog box, you can do the following:

- General Settings:

- Enable and disable the origin server
- Change the host and port
- Mark the origin server as a backup server

- Advanced Settings:

- Change the relative weight
- Set the maximum number of connections that the origin server can handle concurrently
- Set the time that Oracle Traffic Director should take to ramp up the request-forwarding rate to the full capacity of the origin server

11. Specify the parameters that you want to change.

On-screen help and prompts are provided for all of the parameters.

When you change the value in a field or tab out of a text field that you changed, the **OK** button near the upper right corner of the page is enabled.

12. After making the required changes, click **OK**.

A message, confirming that the updated configuration was saved, is displayed in the Console Messages pane.

Changing the Properties of an Origin Server Using WLST

To change the properties of an origin server, run the `otd_setOriginServerProperties` command.

For example, the following command changes the ramp up time to 1200 for the origin server `www.example.com` in the pool `origin-server-pool-1` of the configuration `foo`.

```
props = {}
props['configuration'] = 'foo'
props['origin-server-pool'] = 'origin-server-pool-1'
props['host'] = 'www.example.com'
props['port'] = '12345'
props['ramp-up-time'] = '1200'
otd_setOriginServerProperties(props)
```

Managing Ephemeral Ports

In a topology that includes a client, OTD and Oracle WebLogic Server (WLS), OTD receives external requests at the configured HTTP listener port. OTD then opens up another connection while communicating and proxying the request to the WLS/origin server.

As part of this connection, OTD leverages *ephemeral ports* so that WLS/origin server can send data back to OTD. An ephemeral port is a short-lived transport protocol port for Internet Protocol (IP) communications allocated automatically from a predefined range by the IP software. In Linux, you can limit or restrict these ephemeral ports.

Note:

OTD relies on having sufficient ephemeral ports available so that it can have sufficient pool of connections established with WLS/origin server. Not having enough ephemeral ports will cause delays processing the requests.

Removing an Origin Server from a Pool

You can remove an origin server from a pool that is no longer required. To delete origin server, run the `otd_deleteOriginServer` command.

Note:

When dynamic discovery is enabled (see [Configuring an Oracle WebLogic Server Cluster as an Origin-Server Pool](#)), if you delete an origin server that is an Oracle WebLogic Server instance in a cluster, and then re-configure the Oracle Traffic Director instance. The instance might not start if no valid origin servers remain in the pool.

You can remove an origin server by using either Fusion Middleware Control or the WLST as described in the following topics:

Topics

- [Removing an Origin Server from a Pool Using Fusion Middleware Control](#)
- [Removing an Origin Server from a Pool Using WLST](#)

Removing an Origin Server from a Pool Using Fusion Middleware Control

To remove an origin server from a pool by using the Fusion Middleware Control, do the following:

1. Log in to Fusion Middleware Control, as described in [Displaying Fusion Middleware Control](#).
2. Click the WebLogic Domain button at the upper left corner of the page.
3. Select Administration > OTD Configurations.

A list of the available configurations is displayed.

4. Select the configuration for which you want to delete origin server.
5. Click the Traffic Director Configuration In the Common Tasks pane.
6. Select Administration > Server Pools.

The Server Pools page is displayed. It shows a list of the server pools (HTTP/S and TCP server pools) defined for the configuration.

7. Select the Server Pool for which you want to delete origin server.
8. In the Common Tasks pane, click **Configure Origin Server**.
9. Click the name of the origin server that you want to delete.
10. Click the **Delete** icon for the origin server that you want to delete. After that a window prompts for confirmation, click **OK**.

A message, confirming that the origin server is deleted.

Removing an Origin Server from a Pool Using WLST

To remove the origin server with the specified host and port from a pool, run the `otd_deleteOriginServer` command, as shown in the following example:

```
props = {}
props['configuration'] = 'foo'
props['origin-server-pool'] = 'origin-server-pool-1'
props['host'] = 'www.example.com'
props['port'] = '12345'
otd_deleteOriginServer(props)
```

9

Managing Virtual Servers

You can use multiple virtual servers within a single Oracle Traffic Director instance to provide several entry points—domain names and IP addresses—for client requests, and to offer differentiated services for caching, quality of service, and so on. You can bind virtual servers to one or more listeners—HTTP or HTTPS—and configure them to forward requests to different origin-server pools.

You can configure caching, compression, routing, quality of service, log-file and web application firewall settings individually for each virtual server.

This chapter describes how to create, view, modify, and delete virtual servers, and configure caching. It contains the following sections:

- [Creating Virtual Servers](#)
- [Viewing a List of Virtual Servers](#)
- [Modifying Virtual Server Settings](#)
- [Configuring Routes for a Virtual Server](#)
- [Copying a Virtual Server](#)
- [Deleting a Virtual Server](#)
- [Content Serving](#)

Creating Virtual Servers

When you create a configuration, a virtual server is created automatically with the same name as that of the configuration and is associated with the HTTP listener that was specified while creating the configuration. A default routing rule is also created for the virtual server, to distribute all requests received at the associated HTTP listener to the origin servers that were specified while creating the configuration.

You can create additional virtual servers in a configuration by using either Fusion Middleware Control or WLST as described in the following topics:

- [Creating a Virtual Server Using Fusion Middleware Control](#)
- [Creating a Virtual Server Using WLST](#)

Note:

When you create a virtual server, you are, in effect, modifying a configuration. So for the new virtual-server to take effect in the Oracle Traffic Director instances, you should redeploy the configuration as described in [Activate Configuration Changes](#).

Before You Begin

Before you begin creating a virtual server, decide the following:

- A unique name for the virtual server. Choose the name carefully; after creating a virtual server, you cannot change its name.
- One or more unique listen ports. For information about creating listeners, see [Managing Listeners](#).
- The names of the hosts, or the host patterns, for which the virtual server will handle requests.

When a request is received, Oracle Traffic Director determines the virtual server that should process it, by comparing the `Host` header in the request with the host patterns defined for each virtual server in the configuration.

- The request is routed to the first virtual server that has a host pattern matching the `Host` header in the request.
- If the `Host` header in the request does not match the host patterns defined for any of the virtual servers, or if the request does not contain the `Host` header, the request is routed to the default virtual server that is associated with the HTTP listener through which the request was received.

Note:

When Strict SNI Host Matching is enabled for an HTTP listener, and if for that listener at least one of the virtual servers has certificates, then Oracle Traffic Director returns a `403-Forbidden` error to the client, if any of the following conditions is true:

- The client did not send the SNI host extension during the SSL/TLS handshake.
 - The request does not have the `Host:` header.
 - The host name sent by the client in the SNI host extension during the SSL/TLS handshake does not match the `Host:` header in the request.
- The name of the origin-server pool to which the virtual server should forward requests. For information about creating origin-server pools, see [Managing Origin-Server Pools](#).

Creating a Virtual Server Using Fusion Middleware Control

To create a virtual server by using the Fusion Middleware Control, do the following:

1. Log in to Fusion Middleware Control, as described in [Graphical User Interface-Fusion Middleware Control](#).
2. Click the WebLogic Domain button at the upper left corner of the page.
3. Select Administration > OTD Configurations.
A list of the available configurations is displayed.
4. Select the configuration for which you want to create a virtual server.

5. Click the Traffic Director Configuration In the Common Tasks pane.
6. Select Administration > Virtual Server.
7. In the Common Tasks pane, click **Create**.
The New Virtual Server wizard appears.
8. Follow the on-screen prompts to complete creation of the virtual server by using the details-listener, origin-server pool, and so on-that you decided earlier.
After the virtual server is created, the Results screen of the New Virtual Server wizard displays a message confirming successful creation of the virtual server.
9. Click **Create Virtual Server** on the Results screen.
The details of the virtual server that you just created are displayed on the Virtual Servers page.

Creating a Virtual Server Using WLST

To create a virtual server, run the `otd_createVirtualServer` command.

For example, the following command creates a virtual server named `bar` for the configuration `foo`, and configures the virtual server to forward client requests to the origin-server pool `origin-server-pool-1`.

```
props = {}
props['configuration'] = 'foo'
props['virtual-server'] = 'bar'
props['origin-server-pool'] = 'origin-server-pool-1'
otd_createVirtualServer(props)
```

Viewing a List of Virtual Servers

After a virtual server is created, you can view the current list of virtual servers. To view a list of virtual servers, run the `otd_listVirtualServers` command.

You can view a list of virtual servers by using either Fusion Middleware Control or the WLST as described in the following topics:

Topics

- [Viewing a List of Virtual Servers Using Fusion Middleware Control](#)
- [Viewing a List of Virtual Servers Using WLST](#)

Viewing a List of Virtual Servers Using Fusion Middleware Control

To view a list of virtual servers by using the Fusion Middleware Control, do the following:

1. Log in to Fusion Middleware Control, as described in [Displaying Fusion Middleware Control](#).
2. Click the WebLogic Domain button at the upper left corner of the page.
3. Select Administration > OTD Configurations.

A list of the available configurations is displayed.

4. Select the configuration for which you want to view virtual server.
5. Click the Traffic Director Configuration In the Common Tasks pane.
6. Select Administration > virtual server.

You can view the properties of a virtual server by clicking on its name.

You can view the properties of a virtual server by clicking on its name.

Viewing a List of Virtual Servers Using WLST

To view a list of virtual servers, run the `otd_listVirtualServers` command, as shown in the following example:

```
props = {}  
props['configuration'] = 'foo'  
otd_listVirtualServers(props)
```

You can view the properties of a virtual server in detail by running the `otd_getVirtualServerProperties` command.

You can set the properties of a virtual server by running the `otd_setVirtualServerProperties` command.

Modifying Virtual Server Settings

After you add a virtual server, you might want to change some of the settings such as host patterns, HTTP listeners, language options, and so on.

You can modify virtual servers by using either Fusion Middleware Control or the WLST as described in the following topics:

Note:

When you modify a virtual server, you are, in effect, modifying a configuration. So for the new virtual-server settings to take effect in the Oracle Traffic Director instances, you should redeploy the configuration as described in [Activating Configuration Changes](#).

Topics

- [Modifying a Virtual Server Using Fusion Middleware Control](#)
- [Modifying a Virtual Server Using WLST](#)

Modifying a Virtual Server Using Fusion Middleware Control

To modify a virtual server by using the Fusion Middleware Control, do the following:

1. Log in to Fusion Middleware Control, as described in [Graphical User Interface-Fusion Middleware Control](#).
2. Click the WebLogic Domain button at the upper left corner of the page.
3. Select Administration > OTD Configurations.

A list of the available configurations is displayed.

4. Select the configuration for which you want to modify virtual server.
5. Click the Traffic Director Configuration In the Common Tasks pane.
6. Select Administration > Virtual Server.

The Virtual Servers page is displayed. It shows a list of the virtual servers defined for the configuration.

7. Select the virtual server that you want to modify and click **Edit** button in common tasks pan.

The Virtual Server Settings page is displayed. On this page, you can do the following:

- Enable and disable the virtual server.
- Add, remove, and change host patterns served by the virtual server.
- Add and remove HTTP listeners. For information about creating HTTP listeners, see [Creating a Listener](#).
- Enable SSL/TLS, by associating an RSA or an ECC certificate (or both) with the virtual server. See [RSA and ECC Certificates](#).
- Configure the virtual server to serve instance-level statistics in the form of XML and plain-text reports that users can access through a browser. Note that the statistics displayed in the XML and plain-text reports are for the Oracle Traffic Director instance as a whole and not specific to each virtual server. See [Configuring URL Access to Statistics](#).
- The default language for messages is English. If required, this can be set to other languages that Oracle Traffic Director supports.
- Specify error pages that the virtual server should return to clients for different error codes. This is necessary only if you do not wish to use the default error pages and would like to customize them.

To specify error codes and error pages of your choice, first create html pages that you would like displayed for specific error codes and save them to any directory that can be accessed by the administration server. Next, on the Virtual Server Settings page, in the Error Pages section, click **New Error Page**.

In the **New Error Page** dialog box that appears, select an error code and enter the full path to the error page for that particular error code. In addition to the error codes that are provided, you can create your own custom error code by clicking **Custom Error Code** and entering a value for the same. When done, click **Create Error Page**.

- Enable and quality of service limits—the maximum speed at which the virtual server should transfer data to clients and the maximum number of concurrent connections that the virtual server can support.

In the navigation pane, under the **Virtual Servers** node, you can select the following additional categories of settings for the virtual server. The parameters relevant to the selected category are displayed in the main pane.

- **Settings:** Create, change, and delete rules for routing requests to origin servers. See [Configuring Routes for a Virtual Server](#).
- **Routes:** Create, change, and delete rules for routing requests to origin servers.

- **Caching:** Create, change, and delete rules for caching responses received from origin servers. See [Configuring Caching Parameters](#).
 - **Compression:** Create, change, and delete rules for compressing responses from origin servers before forwarding them to the clients. See [Enabling and Configuring Content Compression](#).
 - **Request Limits:** Create, change, and delete rules for limiting the number and rate of requests received by the virtual server.
 - **Bandwidth Limits:** Enable, change, and delete rules for the number of requests received by the virtual server.
 - **Content Serving:** Create, change, and delete rules for static content serving to origin servers. See [Content Serving Using Fusion Middleware Control](#).
 - **Web Application Firewall:** Enable or disable web application firewall rule set, specify rule set patterns and install rule set files.
 - **Logging:** Define a server log file and location that is specific to the virtual server. See [Configuring Log Preferences](#).
8. Specify the parameters that you want to change.
- On-screen help and prompts are provided for all of the parameters.
- When you change the value in a field or tab out of a text field that you changed, the **Apply** button near the upper right corner of the page is enabled.
- At any time, you can discard the changes by clicking the **Revert** button.
9. After making the required changes, click **Apply**.
- A message, confirming that the updated configuration was saved, is displayed in the Console Messages pane.

Modifying a Virtual Server Using WLST

WLST provides several commands that you can use to change specific parameters of a virtual server.

Table 9-1 WLST Commands for Modifying a Virtual Server

Task/s	WLST Command/s
Enable or disable a virtual server; change the host, the HTTP listener, name and location of the log file; enable SSL/TLS by associating an RSA, or an ECC certificate, or both (see also: RSA and ECC Certificates and Configuring Log Preferences)	<code>otd_setVirtualServerProperties</code>
Create and manage routes (see Configuring Routes)	<code>otd_createRoute</code> <code>otd_listRoutes</code> <code>otd_deleteRoute</code> <code>otd_setRouteProperties</code> <code>otd_getRouteProperties</code>

Table 9-1 (Cont.) WLST Commands for Modifying a Virtual Server

Task/s	WLST Command/s
Create and manage caching rules (see Caching in Oracle Traffic Director)	otd_createCacheRule otd_listCacheRules otd_deleteCacheRule otd_getCacheRuleProperties otd_setCacheRuleProperties
Create and manage compression rules (see Enabling and Configuring Content Compression)	otd_createCompressionRule otd_setCompressionRuleProperties otd_deleteCompressionRule otd_listCompressionRules otd_getCompressionRuleProperties
Change request limiting settings	otd_createRequestLimit otd_deleteRequestLimit otd_getRequestLimitProperties otd_listRequestLimits otd_setRequestLimitProperties
Create and manage content rules (see Content Serving Using Fusion Middleware Control)	otd_createContentRule otd_deleteContentRule otd_listContentRules otd_setContentRuleProperties otd_getContentRuleProperties
Create and manage error pages	otd_createErrorPage otd_deleteErrorPage otd_listErrorPages

For example, the following command changes the name of the HTTP listener associated with the virtual server `bar` to `http-listener-1`.

```
props = {}
props['configuration'] = 'foo'
props['virtual-server'] = 'bar'
props['http-listener-name'] = 'http-listener-1'
otd_setVirtualServerProperties(props)
```

Configuring Routes for a Virtual Server

When you create a configuration, a virtual server is automatically created with the listener that you specified while creating the configuration. For the automatically created virtual server, as well as for any virtual server that you add subsequently in the configuration, a default route is created. The default route rule specifies that all requests to the virtual server should be routed to the origin-server pool that you

specified while creating the virtual server. The default route of a virtual server cannot be deleted, but you can change its properties.

You can create additional routes for the virtual server, to route requests that satisfy specified conditions to specific origin-server pools. For example, in a banking software solution, if customer transactions for loans and deposits are processed by separate applications, you can host each of those applications in a separate origin-server pool behind an Oracle Traffic Director instance. To route customer requests to the appropriate origin-server pool depending on whether the request pertains to the loans or deposits applications, you can set up two routes as follows:

- Route 1: If the request URI starts with `/loan`, send the request to the origin-server pool that hosts the loans application.
- Route 2: If the request URI starts with `/deposit`, send the request to the origin-server pool that hosts the deposits application.

When a virtual server that is configured with multiple routes receives a request, it checks the request URI against each of the available routes. The routes are checked in the order in which they were created.

- If the request satisfies the condition in a route, Oracle Traffic Director sends the request to the origin-server pool specified for that route.
- If the request does not match the condition in any of the defined routes, Oracle Traffic Director sends the request to the origin-server pool specified in the default route.

WebSocket upgrade is enabled by default. In Fusion Middleware Control, use the **WebSocket Upgrade** check box to enable or disable WebSocket protocol for a route. Similarly, WebSocket protocol can also be enabled or disabled using the `websocket-upgrade-enabled` property, which can be set using the `otd_setRouteProperties` WLST command.

You can configure routes in a virtual server by using either Fusion Middleware Control or WLST as described in the following topics:

- [Configuring Routes Using Fusion Middleware Control](#)
- [Configuring Routes Using WLST](#)

Configuring Routes Using Fusion Middleware Control

To configure routes by using the Fusion Middleware Control, do the following:

1. Log in to Fusion Middleware Control, as described in [Graphical User Interface-Fusion Middleware Control](#).
2. Click the WebLogic Domain button at the upper left corner of the page.
3. Select Administration > OTD Configurations.
A list of the available configurations is displayed.
4. Select the configuration for which you want to configure routes.
5. Click the Traffic Director Configuration In the Common Tasks pane.
6. Select Administration > Virtual Servers.

The Virtual Servers page is displayed.

7. In the navigation pane, expand **Virtual Servers**, expand the name of the virtual server for which you want to configure routes, and select **Routes**.

The Routes page is displayed. It lists the routes that are currently defined for the virtual server.

8. **Creating a Route**

- a. Click **Create**.

The New Route dialog box is displayed.

In the **Name** field, enter a name for the new route.

In the **Origin Server Pool** field, select the origin-server pool to which requests that satisfy the specified condition should be routed.

- b. In the Condition Information pane, select a Variable/Function and an Operator from the respective drop-down lists, and provide a value in the **Value** field.

Select the *and/or* operator from the drop-down list when configuring multiple expressions. Similarly, use the *Not* operator when you want the route to be applied only when the given expression is not true.

Click **OK**.

To enter a condition manually, click **Cancel** and then click **Edit Expressions**, the new window opens, Click **Edit Manually**. In the **Condition** field, specify the condition under which the routing rule should be applied. For information about building condition expressions, click the help button near the Condition field or see *Using Variables, Expressions, Wildcards, and String Interpolation in Configuration File Reference for Oracle Traffic Director* .

- c. Click **OK**.

The route that you just created is displayed on the Routes page.

Editing a Route

To change the settings of a route, do the following:

- a. Click the **Name** and select Edit button for the route.

The Route Settings page is displayed.

- b. Specify the parameters that you want to change.

On-screen help and prompts are provided for all of the parameters.

When you change the value in a field or tab out of a text field that you changed, the **OK** button near the upper right corner of the page is enabled.

At any time, you can discard the changes by clicking the **Cancel** button.

- c. After making the required changes, click **OK**.

The updated configuration is saved.

Deleting a Route Rule

To delete a route rule, click the **Delete** button. At the confirmation prompt, click **OK**.

Configuring Routes Using WLST

To create a route, run the `otd_createRoute` command.

Examples:

- The following command creates a route named `loan-route` in the virtual server `bar` of the configuration `foo`, to send requests for which the URI matches the pattern `/loan` to the origin-server pool `loan-app`.

```
props = {}
props['configuration'] = 'foo'
props['virtual-server'] = 'bar'
props['route'] = 'loan-route'
props['origin-server-pool'] = 'loan-app'
props['condition'] = "headers{'content-length'} < 400"
otd_createRoute(props)
```

- The following command creates a route named `images-route` in the virtual server `bar` of the configuration `foo`, to send requests for which the URI path matches the pattern `/images` to the origin-server pool `images-repo`.

```
props = {}
props['configuration'] = 'foo'
props['virtual-server'] = 'bar'
props['route'] = 'images-route'
props['origin-server-pool'] = 'images-repo'
props['condition'] = '$path=/images/*'
otd_createRoute(props)
```

- The following command creates a route named `subnet-route` in the virtual server `bar` of the configuration `foo`, to send requests from any client in the subnet `130.35.46.*` to the origin-server pool `dedicated-osp`.

```
props = {}
props['configuration'] = 'foo'
props['virtual-server'] = 'bar'
props['route'] = 'subnet-route'
props['origin-server-pool'] = 'dedicated-osp'
props['condition'] = '$ip=130.35.46.*'
otd_createRoute(props)
```

- The following command creates a route named `body-route` in the virtual server `bar` of the configuration `foo`, to route requests to the origin-server pool `dedicated-osp` if the request body contains the word *alpha*.

```
props = {}
props['configuration'] = 'foo'
props['virtual-server'] = 'bar'
props['route'] = 'body-route'
props['origin-server-pool'] = 'dedicated-osp'
props['condition'] = '$body = 'alpha''
otd_createRoute(props)
```

Note that the value of the `condition` property should be a regular expression. For information about building condition expressions, see *Using Variables, Expressions, Wildcards, and String Interpolation* in *Configuration File Reference for Oracle Traffic Director*.

- To view a list of the routes defined for a virtual server, run the `otd_listRoutes` command, as shown in the following example:

```
props = {}
props['configuration'] = 'foo'
props['virtual-server'] = 'bar'
otd_listRoutes(props)
```

- To view the properties of a route, run the `otd_getRouteProperties` command, as shown in the following example:

```
props = {}
props['configuration'] = 'foo'
props['virtual-server'] = 'bar'
props['route'] = 'loan-route'
otd_getRouteProperties(props)

keep-alive-timeout=15
sticky-cookie=JSESSIONID
condition="$uri = '/loan'"
validate-server-cert=true
always-use-keep-alive=false
origin-server-pool=origin-server-pool-1
sticky-param=jsessionId
route-header=Proxy-jroute
rewrite-headers=location,content-location
use-keep-alive=true
route=loan-route
log-headers=false
route-cookie=JROUTE
timeout=300
```

- To change the properties of a route, run the `otd_setRouteProperties` command.

Examples:

- The following command changes the websocket idle timeout setting for the route named `route-1` in the virtual server `bar` of the configuration `foo` to 1200 seconds.

```
props = {}
props['configuration'] = 'foo'
props['virtual-server'] = 'bar'
props['route'] = 'route-1'
props['websocket-idle-timeout'] = '1200'
otd_setRouteProperties(props)
```

- The following command enables logging of the headers that Oracle Traffic Director sends to, and receives from, the origin servers associated with the route named `default-route` in the virtual server `bar` of the configuration `foo`.

```
props = {}
props['configuration'] = 'foo'
props['virtual-server'] = 'bar'
props['route'] = 'default-route'
props['log-headers'] = 'true'
otd_setRouteProperties(props)
```

- To disable WebSocket support, run the `otd_setRouteProperties` command with the `websocket-upgrade-enabled` property, as shown in the following example:

```
props = {}
props['configuration'] = 'foo'
props['virtual-server'] = 'bar'
props['route'] = 'default-route'
props['websocket-upgrade-enabled'] = 'false'
otd_setRouteProperties(props)
```

- To delete a route, run the `otd_deleteRoute` command, as shown in the following example:

```
props = {}
props['configuration'] = 'foo'
props['virtual-server'] = 'bar'
props['route'] = 'route-1'
otd_deleteRoute(props)
```

Copying a Virtual Server

When you want to create a virtual server that is similar to an existing virtual server, you can copy the existing configuration and make the required changes later.

Note:

When you copy a virtual server, you are, in effect, modifying a configuration. So for the new virtual server to take effect in the Oracle Traffic Director instances, you should redeploy the configuration as described in [Activating Configuration Changes](#).

You can copy a virtual server by using either Fusion Middleware Control or the WLST as described in the following topics:

Topics

- [Copying a Virtual Server Using Fusion Middleware Control](#)
- [Copying a Virtual Server Using WLST](#)

Copying a Virtual Server Using Fusion Middleware Control

To copy a virtual server by using the Fusion Middleware Control, do the following:

1. Log in to Fusion Middleware Control, as described in [Displaying Fusion Middleware Control](#).
2. Click the WebLogic Domain button at the upper left corner of the page.
3. Select Administration > OTD Configurations.

A list of the available configurations is displayed.

4. Select the configuration for which you want to copy virtual server.
5. Click the Traffic Director Configuration In the Common Tasks pane.
6. Select Administration > virtual server.

The Virtual Servers page is displayed. It shows a list of the virtual servers defined for the configuration.

7. Click the **Duplicate** icon for the virtual server that you want to copy.

The Duplicate Virtual Server dialog box is displayed.

8. Enter a name for the new virtual server, and click **OK**.

A message is displayed confirming that the new virtual server was created.

Copying a Virtual Server Using WLST

To copy a virtual server, run the `otd_copyVirtualServer` command.

For example, the following command creates a copy (`baz`) of the virtual server `bar`.

```
props = {}
props['configuration'] = 'foo'
props['source-virtual-server'] = 'bar'
props['dest-virtual-server'] = 'baz'
otd_copyVirtualServer(props)
```

Deleting a Virtual Server

You can delete virtual server instances that are no longer required. To delete virtual server instances, run the `otd_deleteVirtualServer` command.

You can delete instances of virtual server by using either Fusion Middleware Control or the WLST as described in the following topics:

Topics

- [Deleting a Virtual Server Using Fusion Middleware Control](#)
- [Deleting a Virtual Server Using WLST](#)

Deleting a Virtual Server Using Fusion Middleware Control

To delete a virtual server by using the Fusion Middleware Control, do the following:

1. Log in to Fusion Middleware Control, as described in [Displaying Fusion Middleware Control](#).
2. Click the WebLogic Domain button at the upper left corner of the page.
3. Select Administration > OTD Configurations.

A list of the available configurations is displayed.

4. Select the configuration for which you want to delete virtual server.
5. Click the Traffic Director Configuration In the Common Tasks pane.
6. Select Administration > virtual server.

The Virtual Servers page is displayed. It shows a list of the virtual servers defined for the configuration.

7. Click the **Delete** icon for the virtual server that you want to delete.

A prompt to confirm the deletion is displayed.

8. Click **OK**.

A message is displayed in the Console Message pane confirming that the virtual server was deleted.

Deleting a Virtual Server Using WLST

To delete a virtual server, run the `otd_deleteVirtualServer` command, as shown in the following example:

```
props = {}
props['configuration'] = 'foo'
props['virtual-server'] = 'bar'
otd_deleteVirtualServer(props)
```

Caching in Oracle Traffic Director

Caching frequently requested data reduces the time that clients have to wait for responses. In addition, when frequently accessed objects (response body and headers) are stored in memory, the load on the origin servers is significantly reduced.

To enable caching, you must configure caching rules.

- Both static and dynamically generated content from origin servers are cached.
- Only Successful responses (response code: 200) are cached.
- Responses to only HTTP GET and HEAD requests are cached.
- Oracle Traffic Director caches the response body and all of the response headers except `Dest-IP`, `Proxy-Agent`, `Proxy-Connection`, `Server`, `Set-Cookie`, `State-Info`, and `Status`.
- Oracle Traffic Director honors `Cache-Control` directives from origin servers, including directives to revalidate content and to not cache certain headers.
- You can configure one or more caching rules specific to each virtual server, subject to the overall limits—maximum heap space, maximum entries, and maximum object size—specified for the configuration.

You can configure the caching rules to be applicable either to all requests or to only those requests that match a specified condition.

- Cached data is held in the process memory (heap), separately for each virtual server. When the instance is stopped or restarted, the cache becomes empty.
- WebSocket upgrade requests are not cached.

When a client first requests an object, Oracle Traffic Director sends the request to an origin server. This request is a *cache miss*. If the requested object matches a caching rule, Oracle Traffic Director caches the object. For subsequent requests for the same object, Oracle Traffic Director serves the object from its cache to the client. Such requests are *cache hits*.

The caching behavior in Oracle Traffic Director is consistent with the specification in section 13 of RFC 2616. See <http://www.w3.org/Protocols/rfc2616/rfc2616-sec13.html>.

Reviewing Cache Settings and Metrics for an Instance

Viewing Caching Settings

- To view the current caching settings for a configuration, run the `otd_getCacheProperties` command, as shown in the following example:

```
props = {}
props['configuration'] = 'foo'
otd_getCacheProperties(props)

enabled=true
max-entries=1024
replacement=lru
max-heap-object-size=524288
max-heap-size=10485760
```

- To view a list of the caching rules defined for a virtual server, run the `otd_listCacheRules` command, as shown in the following example:

```
props = {}
props['configuration'] = 'foo'
props['virtual-server'] = 'bar'
otd_listCacheRules(props)

cache-rule-1
cache-rule-2
```

- To view the current settings of a virtual server-specific caching rule, run the `otd_getCacheRuleProperties` command, as shown in the following example:

```
props = {}
props['configuration'] = 'foo'
props['virtual-server'] = 'bar'
props['cache-rule'] = 'cache-rule-1'
otd_getCacheRuleProperties(props)

condition="$uri = '^/images'"
enabled=true
max-reload-interval=3600
min-reload-time=0
last-modified-factor=0
min-object-size=1
cache-https-response=true
rule=cache-rule-2
query-maxlen=0
compression=true
cache-http-response=false
```

Viewing Caching Metrics

You can view the current cache-hit rate, the cache heap usage, and the rate of successful revalidation of cache entries in the plain-text `perfdump` report, as shown in the following example:

```
Proxy Cache:
-----
Proxy Cache Enabled          yes
Object Cache Entries         42
Cache lookup (hits/misses)   183/79
Requests served from Cache    22
Revalidation (successful/total) 30/38 ( 78.95%)
Heap space used               16495
```

- Proxy Cache Enabled indicates whether caching is enabled for the instance.

- `Object Cache Entries` is the number of entries (URIs) currently in the cache.
- `Cache lookup (hits/misses)`
 - The first number is the number of times an entry was found in the cache for the requested URI.
 - The second number is the number of times the requested URI was not found in the cache.
- `Requests served from Cache` is the number of requests that Oracle Traffic Director served from the cache.
- `Revalidation (successful/total)`
 - The first number is the number of times revalidation of cached content was successful.
 - The second number is the total number of times Oracle Traffic Director attempted to revalidate cached content.
 - The percentage value is the ratio of successful revalidations to the total number of revalidation attempts.
- `Heap space used` is the amount of cache heap space that is currently used.

Tunable Caching Parameters

Caching can be considered effective in reducing the response time for clients when the cache-hit rate is high; that is, a relatively large number of requests are served from the cache instead of being sent to origin servers. For a high cache-hit rate, there should be sufficient memory to store cacheable responses from origin servers and the entries in the cache should be validated regularly.

 **Note:**

Dynamic content is generally not cacheable. So if the application or content being served by the origin servers consists mostly of dynamic content, the cache-hit rate is bound to be low. In such cases, enabling and tuning caching might not yield a significant performance improvement.

To improve the cache-hit rate, you can tune the following caching parameters:

- **Cache-entry replacement method**

When the cache becomes full—that is, the number of entries reaches the maximum entries limit, or the cache heap size reaches the maximum cache heap space—further entries in the cache can be accommodated only if existing entries are removed. The cache-entry replacement method specifies how Oracle Traffic Director determines the entries that can be removed from the cache.

- The default replacement method is Least Recently Used (`lru`). When the cache is full, Oracle Traffic Director discards the least recently used entries first.
- The other available method is Least Frequently Used (`lfu`). When the cache is full, Oracle Traffic Director discards the least frequently used entry first.

In either method, every time Oracle Traffic Director serves content from the cache, it needs to track usage information—the time the content was served in the case of the `lru` replacement method, and the number of times the content was served in the case of `lfu`. So the time saved by serving content directly from the cache instead of sending the request to the origin server, is offset to a certain extent by the latency caused by the need to track usage information. Between the two methods, `lru` requires marginally lower computing resources.

You can disable cache-entry replacement by specifying `false` as the replacement method.

- **Maximum cache heap space**

If only a small portion of the available heap space is used, it is possible that responses are not being cached because the virtual server-specific caching rules are defined too narrowly.

The optimal cache heap size depends upon how much system memory is free. With a large cache heap, Oracle Traffic Director can cache more content and therefore obtain a better hit ratio. However, the heap size should not be so large that the operating system starts paging cached content.

- **Maximum number of entries in the cache**

If the number of entries in the cache, as shown in the `perfdump` report, is consistently near, or at, the maximum number of entries, it is an indication that the cache might not be large enough. Consider increasing the maximum number of entries.

If the number of entries in the cache is very low when compared with the maximum allowed entries, it is possible that responses are not being cached because the virtual server-specific caching rules are defined too narrowly.

- **Maximum size of cacheable object**

To conserve system resources, you can limit the size of objects that are cached, even if the objects fulfill other caching rules.

If you observe that objects that are larger than the maximum cached object size are requested frequently, consider increasing the limit.

In a caching rule for a specific virtual server, you can specify the following parameters:

- Minimum and maximum size of objects that can be cached
- Minimum and maximum interval between cache-validation checks
- Maximum number of characters in a query string that can be cached
- Whether to compress content before caching
- Whether to cache HTTPS responses

Configuring Caching Parameters

You can configure caching settings by using either Fusion Middleware Control or the WLST.

Configuring Caching Settings Using Fusion Middleware Control

To configure caching settings by using the Fusion Middleware Control, do the following:

1. Log in to Fusion Middleware Control, as described in [Displaying Fusion Middleware Control](#).
2. Click the WebLogic Domain button at the upper left corner of the page.
3. Select Administration > OTD Configurations.
A list of the available configurations is displayed.
4. Select the configuration for which you want to modify.
5. Click the Traffic Director Configuration In the Common Tasks pane.
6. Select Administration > Virtual Servers.
The Virtual Servers page is displayed.
7. In the navigation pane, expand **Virtual Servers**, expand the name of the virtual server for which you want to configure cache, and select **Caching**.
The Cache Rules page is displayed. It lists the Cache rules that are currently defined for the virtual server.
8. In the navigation pane, select **Advanced Settings**.
The Advanced Settings page is displayed.
9. Specify the caching parameters that you want to change.
On-screen help and prompts are provided for all of the parameters.
When you change the value in a field or tab out of a text field that you changed, the **OK** button near the upper right corner of the page is enabled.
At any time, you can discard the changes by clicking the **Cancel** button.
10. After making the required changes, click **OK**.
 - A message, confirming that the updated configuration was saved, is displayed in the Console Messages pane.

Configuring Virtual Server-Specific Caching Rules Using Fusion Middleware Control

To create virtual server-specific caching rules by using the Fusion Middleware Control, do the following:

1. Log in to Fusion Middleware Control, as described in [Displaying Fusion Middleware Control](#).
2. Click the WebLogic Domain button at the upper left corner of the page.
3. Select Administration > OTD Configurations.
A list of the available configurations is displayed.
4. Select the configuration for which you want to create virtual server-specific caching rules.
5. Click the Traffic Director Configuration In the Common Tasks pane.
6. Select Administration > Virtual Servers.
The Virtual Servers page is displayed.
7. In the navigation pane, expand **Virtual Servers**, expand the name of the virtual server for which you want to create caching rules, and select **Caching**.

The Caching page is displayed. It lists the caching rules that are currently defined for the virtual server, and indicates whether the rules are enabled.

Creating a Caching Rule

- a. Click **New Caching Rule**.

The New Cache Rule dialog box is displayed.

In the **Name** field, enter a name for the new caching rule.

- b. Click **Ok**.

The caching rule that you just created is displayed on the Caching page.

Editing a Caching Rule

To enable or disable a caching rule, or to change the settings of a rule, do the following:

- a. Click the **Name** of the caching rule that you want to edit.

The Edit Cache Rule dialog box is displayed.

 **Note:**

To access the condition builder to edit conditions, select **Requests satisfying the condition** and click **Edit**. The condition builder enables you to delete old expressions and add new ones.

- b. Specify the parameters that you want to change.

On-screen help and prompts are provided for all of the parameters.

For information about building condition expressions, click the help button near the Condition field or see *Using Variables, Expressions, and String Interpolation* command in the *Configuration File Reference for Oracle Traffic Director* .

When you change the value in a field or tab out of a text field that you changed, the **Save** button near the upper right corner of the page is enabled.

At any time, you can discard the changes by clicking the **Reset** button.

- c. After making the required changes, click **Save**.

A message, confirming that the updated configuration was saved, is displayed in the Console Messages pane.

Deleting a Caching Rule

To delete a caching rule, click the **Delete** button. At the confirmation prompt, click **OK**.

Configuring Caching Settings Using WLST

- To change the caching properties for a configuration, run the `otd_setCacheProperties` command.

For example, the following command changes the maximum cache heap space to 20 MB.

```
props = {}  
props['configuration'] = 'foo'
```

```
props['max-heap-space'] = '20971520'
otd_setCacheProperties(props)
```

- To create a caching rule for a virtual server, run the `otd_createCacheRule` command.

For example, the following command creates a rule named `cache-rule-images` for the virtual server `bar` in the configuration `foo`, to cache the requests for which the expression `$uri='^/images'` evaluates to true.

```
props = {}
props['configuration'] = 'foo'
props['virtual-server'] = 'bar'
props['cache-rule'] = 'cache-rule-images'
props['condition'] = '$uri='^/images''
otd_createCacheRule(props)
```

Note that the value of the `condition` property should be a regular expression. For information about building condition expressions, see *Using Variables, Expressions, and String Interpolation* command in the *Configuration File Reference for Oracle Traffic Director* .

- To change a caching rule, run the `otd_setCacheRuleProperties` command.

For example, the following command disables compression of content for the caching rule `cache-rule-images`.

```
props = {}
props['configuration'] = 'foo'
props['virtual-server'] = 'bar'
props['cache-rule'] = 'cache-rule-images'
props['compression'] = 'false'
otd_setCacheRuleProperties(props)
```

- To delete a caching rule, run the `otd_deletecacheRule` command, as shown in the following example.

```
props = {}
props['configuration'] = 'foo'
props['virtual-server'] = 'bar'
props['cache-rule'] = 'cache-rule-1'
otd_deleteCacheRule(props)
```

Content Serving

Oracle Traffic Director supports static content serving by managing content-rules. No dynamic content serving is supported. Content-rule is created based on URI-prefix and URI-prefix should be unique across all the content-rule. Oracle Traffic Director admin supports static content serving only for content-rule, mime types and file cache.

You can configure content serving by using either Fusion Middleware Control or WLST as described in the following topics:

Topics

- [Content Serving Using Fusion Middleware Control](#)
- [Configuring Content Serving Using WLST](#)

Content Serving Using Fusion Middleware Control

To configure content serving by using the Fusion Middleware Control, do the following:

1. Log in to Fusion Middleware Control, as described in [Displaying Fusion Middleware Control](#).
2. Click the WebLogic Domain button at the upper left corner of the page.
3. Select Administration > OTD Configurations.

A list of the available configurations is displayed.

4. Select the configuration for which you want to configure content serving.
5. Click the Traffic Director Configuration In the Common Tasks pane.
6. Select Administration > Virtual Servers.

The Virtual Servers page is displayed.

7. In the navigation pane, expand **Virtual Servers**, expand the name of the virtual server for which you want to configure content serving, and select **Content serving**.

The content serving page is displayed. It lists the content serving that are currently defined for the virtual server.

8. Creating a Content serving

- a. Click **Create**.

The New Content Serving dialog box is displayed.

- b. In the **Name** field, enter a name for the new content rule.
- c. In the **URI Prefix** field, enter the specified URI for that content rule.
- d. In the **Directory Path** field, enter the specified directory where all the new content rules are available.

Click **OK**.

The Content Serving that you just created is displayed on the Content Serving page.

Editing a Content Serving

To change the settings of a Content Serving, do the following:

- a. Click the **Name** and select Edit button for the Content Serving.

The Content Serving Settings page is displayed.

- b. Specify the parameters that you want to change.

When you change the value in a field or tab out of a text field that you changed, the **OK** button near the upper right corner of the page is enabled.

At any time, you can discard the changes by clicking the **Cancel** button.

- c. After making the required changes, click **OK**.

The updated configuration is saved.

Deleting a Content Serving

To delete a Content Serving rule, click the **Delete** button. At the confirmation prompt, click **OK**.

Configuring Content Serving Using WLST

- To create a content rule, run the `otd_createContentRule` command, as shown in the following example:

```
props = {}
props['configuration'] = 'foo'
props['virtual-server'] = 'bar'
props['uri-prefix'] = '/baz'
props['directory-path'] = '/qux'
props['content-rule'] = 'content-rule-1'
otd_createContentRule(props)
```

- To view the list of content rules defined for a virtual server, run the `otd_listContentRules` command, as shown in the following example:

```
props = {}
props['configuration'] = 'foo'
props['virtual-server'] = 'bar'
otd_listContentRules(props)
```

- To view the content rule properties run the `otd_getContentRuleProperties` command, as shown in the following example:

```
props = {}
props['configuration'] = 'foo'
props['virtual-server'] = 'bar'
props['content-rule'] = 'content-rule-1'
otd_getContentRuleProperties(props)
```

- To set content rule properties run the `otd_setContentRuleProperties` command, as shown in the following example:

```
props['configuration'] = 'foo'
props['virtual-server'] = 'bar'
props['content-rule'] = 'content-rule-1'
props['index-files'] = 'home.htm'
otd_setContentRuleProperties(props)
```

- To delete a content rule, run the `otd_deleteContentRule` command, as shown in the following example:

```
props = {}
props['configuration'] = 'foo'
props['virtual-server'] = 'bar'
props['content-rule'] = 'content-rule-1'
otd_deleteContentRule(props)
```

- To create a mime type, run the `otd_createMimeType` command, as shown in the following example:

```
props = {}
props['configuration'] = 'foo'
props['content-type'] = 'bar'
props['extensions'] = 'baz'
otd_createMimeType(props)
```

- To view the list of mime types, run the `otd_listMimeType` command, as shown in the following example:

```
props = {}  
props['configuration'] = 'foo'  
otd_listMimeTypes(props)
```

- To delete a mime type, run the `otd_deleteMimeType` command, as shown in the following example:

```
props = {}  
props['configuration'] = 'foo'  
props['content-type'] = 'bar'  
props['extensions'] = 'baz'  
otd_createMimeType(props)
```

- To view the file cache properties run the `otd_getFileCacheProperties` command, as shown in the following example:

```
props = {}  
props['configuration'] = 'foo'  
otd_getFileCacheProperties(props)
```

- To set File Cache properties run the `otd_setFileCacheProperties` command, as shown in the following example:

```
props = {}  
props['configuration'] = 'foo'  
props['max-age'] = '1200'  
otd_setFileCacheProperties(props)
```

10

Managing TCP Proxies

A TCP Proxy handles TCP requests through TCP listeners for traffic tunnelling. While a TCP Proxy can have several TCP listeners associated with it, a TCP listener can be associated with only one TCP Proxy.

This chapter describes how to create, view, modify, and delete TCP proxies. It contains the following topics:

- [Creating a TCP Proxy](#)
- [Viewing a List of TCP Proxies](#)
- [Modifying a TCP Proxy](#)
- [Deleting a TCP Proxy](#)

Creating a TCP Proxy

You can create TCP proxies to handle requests through TCP listeners. To create a TCP proxy, run the `otd_createTcpProxy` command.

Note:

When you create a TCP Proxy, you are, in effect, modifying a configuration. So for the new TCP Proxy settings to take effect in the Oracle Traffic Director instances, you should redeploy the configuration as described in [Activating Configuration Changes](#).

Before You Begin

Before you begin creating a TCP Proxy, decide the following:

- A unique name for the proxy. Choose the name carefully. After creating a proxy, you cannot change its name.
- A unique IP address (or host name) and port number combinations for the listener.

You can define multiple TCP listeners with the same IP address combined with different port numbers, or with a single port number combined with different IP addresses. So each of the following IP address and port number combinations would be considered a unique listener:

```
10.10.10.1:80
10.10.10.1:81
10.10.10.2:80
10.10.10.2:81
```

- The name of the origin-server pool to which the TCP Proxy should forward requests. See [Managing Origin Server Pools](#).

You can create a TCP proxy by either using Fusion Middleware Control or the WLST as described in the following topics:

Topics

- [Creating a TCP Proxy Using Fusion Middleware Control](#)
- [Creating a TCP Proxy Using WLST](#)

Creating a TCP Proxy Using Fusion Middleware Control

To create a TCP Proxy by using the Fusion Middleware Control, do the following:

1. Log in to Fusion Middleware Control, as described in [Graphical User Interface-Fusion Middleware Control](#).
2. Click the WebLogic Domain button at the upper left corner of the page.
3. Select Administration > OTD Configurations.
A list of the available configurations is displayed.
4. Select the configuration for which you want to create a TCP proxy.
5. Click the Traffic Director Configuration In the Common Tasks pane.
6. Select Administration > TCP proxies.
7. In the Common Tasks pane, click **Create**.
The New TCP Proxy wizard appears.
8. Follow the on-screen prompts to complete creation of the TCP Proxy by using the details—proxy name, listener name, IP address, port, and so on—that you decided earlier.

Note:

- Select **Enable FTP** option if you want to enable FTP support on a TCP proxy.
- If the TCP traffic on the port is over SSL, for example T3S, then select the **SSL/TLS** check box on the first screen of the New TCP Proxy wizard and select the certificate to be used. See [Configuring SSL on a HTTP/TCP Listener](#).

After the proxy is created, the Results screen of the New TCP Proxy wizard displays a message confirming successful creation of the proxy.

9. Click **Close** on the Results screen.
 - The details of the TCP Proxies that you just created are displayed on the TCP proxies page.
 - In addition, the **Deployment Pending** message is displayed at the top of the main pane. You can either deploy the updated configuration immediately by clicking **Deploy Changes**, or you can do so later after making further changes, as described in [Activating Configuration Changes](#).

Creating a TCP Proxy Using WLST

To create a TCP proxy with a set of initial values, run the `otd_createTcpProxy` command.

For example, the following command creates a TCP Proxy named `bar` for the configuration `foo` with the origin-server-pool as `tcp-origin-server-pool-1`.

```
props = {}
props['configuration'] = 'foo'
props['tcp-proxy'] = 'bar'
props['origin-server-pool-name'] = 'tcp-origin-server-pool-1'
otd_createTcpProxy(props)
```

For example, the following command creates a TCP Proxy named `bar` for the configuration `foo` with the origin-server-pool as `tcp-origin-server-pool-1` and protocol as `ftp`.

```
props = {}
props['configuration'] = 'foo'
props['tcp-proxy'] = 'bar'
props['protocol'] = 'ftp'
props['origin-server-pool'] = 'tcp-origin-server-pool-1'
otd_createTcpProxy(props)
```

The FTP configuration is enabled for the tcp proxy with properties `ssl-termination`, `origin-explicit-fts` and `client-explicit-fts` being `false` and `true` respectively. These properties can be modified later using `otd_setTcpProxyProperties`.

Viewing a List of TCP Proxies

After creating a TCP proxy, you can view the current state of the proxies. to view a list of TCP proxies, run the `otd_listTcpProxies` command.

You can view a list of TCP proxies by using either Fusion Middleware Control or the WLST as described in the following topics:

Topics

- [Viewing a List of TCP Proxies Using Fusion Middleware Control](#)
- [Viewing a List of TCP Proxies Using WLST](#)

Viewing a List of TCP Proxies Using Fusion Middleware Control

To view a list of TCP proxies by using the Fusion Middleware Control, do the following:

1. Log in to Fusion Middleware Control, as described in [Displaying Fusion Middleware Control](#).
2. Click the WebLogic Domain button at the upper left corner of the page.
3. Select Administration > OTD Configurations.
A list of the available configurations is displayed.
4. Select the configuration for which you want to view a TCP proxy.

5. Click the Traffic Director Configuration In the Common Tasks pane.
6. Select Administration > TCP proxies.

The TCP Proxies page is displayed. It shows a list of the TCP proxies defined for the configuration.

Viewing a List of TCP Proxies Using WLST

To view a list of TCP proxies, run the `otd_listTcpProxies` command, as shown in the following example:

```
props = {}
props['configuration'] = 'foo'
otd_listTcpProxies(props)
```

```
tcp_proxy1
tcp_proxy2
```

You can view the properties of a TCP Proxy in detail by running the `otd_getTcpProxyProperties` command.

Modifying a TCP Proxy

After you create a TCP Proxy instance, you may want to change some of the settings such as port range, client/server FTP settings, and so on. To modify TCP Proxy settings, use the `otd_setTcpProxyProperties` command.

You can modify TCP Proxy settings by using either Fusion Middleware Control or the WLST as described in the following topics:

Topics

- [Modifying a TCP Proxy Using Fusion Middleware Control](#)
- [Modifying a TCP Proxy Using WLST](#)

Modifying a TCP Proxy Using Fusion Middleware Control

To modify a TCP Proxy by using the Fusion Middleware Control, do the following:

1. Log in to Fusion Middleware Control, as described in [Displaying Fusion Middleware Control](#).
2. Click the WebLogic Domain button at the upper left corner of the page.
3. Select Administration > OTD Configurations.

A list of the available configurations is displayed.

4. Select the configuration for which you want to modify a TCP proxy.
5. Click the Traffic Director Configuration In the Common Tasks pane.
6. Select Administration > TCP proxies.

The TCP Proxies page is displayed. It shows a list of the TCP proxies defined for the configuration.

7. Click the name of the TCP Proxy that you want to modify.

The TCP Proxy Settings page is displayed. On this page, you can do the following:

- Enable and disable the TCP Proxy.
- Change the origin server pool and idle timeout.
- Add and remove TCP listeners. For information about creating TCP listeners, see [Creating a Listener](#).
- Modify port ranges for active and passive FTP connections.
- View the client FTP settings.

Client Explicit SSL is enabled. This means that SSL is enabled on request for the client connection. This can only be disabled if all the associated TCP listeners have SSL enabled.

- Modify the server FTP settings.

Server Explicit SSL is enabled. This means that SSL is enabled on request for the origin server connection.

Server FTP settings cannot be changed because SSL is not enabled on the server pool. Click **Edit** to navigate to the server pool edit page and enable SSL.

8. Specify the parameters that you want to change.

On-screen help and prompts are provided for all of the parameters.

When you change the value in a field or tab out of a text field that you changed, the **OK** button near the upper right corner of the page is enabled.

At any time, you can discard the changes by clicking the **Cancel** button.

9. After making the required changes, click **OK**.

A message, confirming that the updated proxy was saved, is displayed in the Console Messages pane.

Modifying a TCP Proxy Using WLST

To change the properties of a TCP proxy, run the `otd_setTcpProxyProperties` command.

- For example, the following command changes the session idle timeout of the proxy `bar` in the configuration `foo` to 1200.

```
props = {}
props['configuration'] = 'foo'
props['tcp-proxy'] = 'bar'
props['session-idle-timeout'] = '1200'
otd_setTcpProxyProperties(props)
```

- For example, the following command enables FTP configuration for the TCP proxy with properties `'ssl-termination'`, `'origin-explicit-fts'` and `'client-explicit-fts'` as `'false'`, `'true'` and `'true'` respectively.

```
props = {}
props['configuration'] = 'foo'
props['tcp-proxy'] = 'bar'
props['client-explicit-fts'] = 'true'
otd_setTcpProxyProperties(props)
```


Deleting a TCP Proxy

You can delete TCP Proxy instances that are no longer required. To delete TCP Proxy instances, run the `otd_deleteTcpProxy` command.

You can delete TCP Proxy instances by using either Fusion Middleware Control or the WLST as described in the following topics:

Topics

- [Deleting a TCP Proxy Using Fusion Middleware Control](#)
- [Deleting a TCP Proxy Using WLST](#)

Deleting a TCP Proxy Using Fusion Middleware Control

To delete a TCP Proxy by using the Fusion Middleware Control, do the following:

1. Log in to Fusion Middleware Control, as described in [Displaying Fusion Middleware Control](#).
2. Click the WebLogic Domain button at the upper left corner of the page.
3. Select Administration > OTD Configurations.

A list of the available configurations is displayed.

4. Select the configuration for which you want to delete a TCP proxy.
5. Click the Traffic Director Configuration In the Common Tasks pane.
6. Select Administration > TCP proxies.

The TCP Proxies page is displayed. It shows a list of the TCP proxies defined for the configuration.

7. Click the **Delete** icon for the TCP Proxy that you want to delete.

A prompt to confirm deletion of the proxy is displayed. If the proxy is associated with any listeners, the prompt shows the names of those listeners.

8. To proceed with the deletion, click **Yes**.

A message is displayed in the Console Message pane confirming that the TCP Proxy was deleted.

In addition, the **Deployment Pending** message is displayed at the top of the main pane. You can either deploy the updated configuration immediately by clicking **Deploy Changes**, or you can do so later after making further changes, as described in [Activating Configuration Changes](#).

Deleting a TCP Proxy Using WLST

To delete a TCP Proxy, run the `otd_deleteTcpProxy` command, as shown in the following example:

```
props = {}
props['configuration'] = 'foo'
props['tcp-proxy'] = 'bar'
otd_deleteTcpProxy(props)
```

11

Managing Listeners

Connections between the clients and Oracle Traffic Director instances are created through HTTP and TCP listeners. Each listener is a unique combination of an IP address (or host name) and a port number.

This chapter describes how to create, view, modify, and delete listeners. It contains the following topics:

- [Creating a Listener](#)
- [Viewing a List of Listeners](#)
- [Modifying a Listener](#)
- [Deleting a Listener](#)
- [Configuring Status Listener](#)

Creating a Listener

You can create listeners to bind them to a virtual server and configure them to forward requests to different origin-server pools. To create a HTTP listener, run the `otd_createHttpListener` command, and to create a TCP listener, run the `otd_createTcpListener` command.

Before You Begin

Before you begin creating an listener, decide the following:

- A unique name for the listener. Choose the name carefully. After creating a listener, you cannot change its name.
- A unique IP address (or host name) and port number combinations for the listener.

You can define multiple listeners with the same IP address combined with different port numbers, or with a single port number combined with different IP addresses. So each of the following IP address and port number combinations would be considered a unique listener:

```
10.10.10.1:80
10.10.10.1:81
10.10.10.2:80
10.10.10.2:81
```

- For HTTP listeners:

- The default virtual server for the listener.

Oracle Traffic Director routes requests to the default virtual server if it cannot match the `Host` value in the request header with the host patterns specified for any of the virtual servers bound to the listener. For information about specifying the host patterns for virtual servers, see [Creating Virtual Servers](#).

- The server name to be included in any URLs that are generated automatically by the server and sent to the client. This server name should be the virtual host name, or the alias name if your server uses an alias. If a colon and port

number are appended to the server name then that port number is used in the autogenerated URLs.

- For TCP listeners: TCP proxy for the listener.

A TCP proxy handles TCP requests through TCP listeners for traffic tunnelling. A TCP proxy can have several TCP listeners associated with it. You can associate TCP listeners and configure TCP proxy settings from this page. See [Creating a TCP Proxy](#).

You can create listeners by using either Fusion Middleware Control or WLST as described in the following topics:

Topics

- [Creating a Listener Using Fusion Middleware Control](#)
- [Creating a Listener Using WLST](#)

Creating a Listener Using Fusion Middleware Control

To create an HTTP listener by using the Fusion Middleware Control, do the following:

1. Log in to Fusion Middleware Control, as described in [Graphical User Interface-Fusion Middleware Control](#).
2. Click the WebLogic Domain button at the upper left corner of the page.
3. Select Administration > OTD Configurations.
A list of the available configurations is displayed.
4. Select the configuration for which you want to create a HTTP Listener.
5. Click the Traffic Director Configuration In the Common Tasks pane.
6. Select Administration > Listener.
7. In the Common Tasks pane, click **Create** under HTTP Listener.
The New HTTP Listener wizard appears.
8. Follow the on-screen prompts to complete creation of the HTTP listener by using the details—listener name, IP address, port, and so on—that you decided earlier.

Note:

If certificates are available in the configuration, in the second screen of the wizard, an **SSL/TLS** check box will be available. If you want the new listener to receive HTTPS requests, click the check box to enable **SSL/TLS** and then select the appropriate certificate from the drop-down list.

After the HTTP listener is created, the Results screen of the New HTTP Listener wizard displays a message confirming successful creation of the listener.

9. Click **OK** on the Results screen.
 - The details of the listener that you just created are displayed on the Listeners page.

Creating a TCP Listener Using Fusion Middleware Control

To create a TCP listener by using the Fusion Middleware Control, do the following:

1. Log in to Fusion Middleware Control, as described in [Graphical User Interface-Fusion Middleware Control](#).
2. Click the WebLogic Domain button at the upper left corner of the page.
3. Select Administration > OTD Configurations.
A list of the available configurations is displayed.
4. Select the configuration for which you want to create a TCP Listener.
5. Click the Traffic Director Configuration In the Common Tasks pane.
6. Select Administration > Listener.
7. In the Common Tasks pane, click **Create TCP Listener**.
The New TCP Listener wizard appears.
8. Follow the on-screen prompts to complete creation of the TCP listener by using the details—listener name, IP address, port, and so on—that you decided earlier.

Note:

If certificates are available in the configuration, in the second screen of the wizard, an **SSL/TLS** check box will be available. If you want the new listener to receive T3S requests, click the check box to enable **SSL/TLS** and then select the appropriate certificate from the drop-down list.

After the TCP listener is created, the Results screen of the New TCP Listener wizard displays a message confirming successful creation of the listener.

9. Click **OK** on the Results screen.
The details of the listener that you just created are displayed on the Listeners page.

Creating a Listener Using WLST

- To create an HTTP listener, run the `otd_createHttpListener` command.

For example, the following command creates an HTTP listener named `http-listener-1` for the configuration `foo` with the port as `23456` and the default virtual server as `bar`.

```
props = {}
props['configuration'] = 'foo'
props['http-listener'] = 'http-listener-1'
props['port'] = '23456'
props['server-name'] = 'example.com'
props['default-virtual-server-name'] = 'bar'
otd_createHttpListener(props)
```

- To create a TCP listener, run the `otd_createTcpListener` command.

For example, the following command creates a TCP listener named `tcp_listener_1` for the configuration `foo` with the port as `34567` and the TCP proxy as `tcp_proxy-1`.

```
props = {}
props['configuration'] = 'foo'
props['tcp-listener'] = 'tcp-listener-1'
props['port'] = '34567'
props['tcp-proxy-name'] = 'tcp-proxy-1'
otd_createTcpListener(props)
```

Viewing a List of Listeners

After creating listeners, you can view the current list of listeners. To view the list of listeners, run the `otd_listHttpListeners` or the `otd_listTcpListeners` commands.

You can view the list of listeners by using either Fusion Middleware Control or WLST as described in the following topics:

Topics

- [Viewing a List of Listeners Using Fusion Middleware Control](#)
- [Viewing a List of Listeners Using WLST](#)

Viewing a List of Listeners Using Fusion Middleware Control

To view a list of HTTP or TCP listeners by using the Fusion Middleware Control, do the following:

1. Log in to Fusion Middleware Control, as described in [Displaying Fusion Middleware Control](#).
2. Click the WebLogic Domain button at the upper left corner of the page.
3. Select Administration > OTD Configurations.
A list of the available configurations is displayed.
4. Select the configuration for which you want to view a HTTP or TCP Listener.
5. Click the Traffic Director Configuration In the Common Tasks pane.
6. Select Administration > Listener.

The Listeners page is displayed. It shows a list of the listeners defined for the configuration.

Note:

HTTP and TCP listeners can also be identified by their icons.

You can view the properties of a listener in detail by clicking on its name.

Viewing a List of Listeners Using WLST

- To view a list of HTTP listeners, run the `otd_listHttpListeners` command, as shown in the following example:

```
props = {}
props['configuration'] = 'foo'
otd_listHttpListeners(props)
```

```
listener-1
listener-2
```

You can view the properties of an HTTP listener in detail by running the `otd_getHttpListenerProperties` command.

- To view a list of TCP listeners, run the `otd_listTcpListeners` command, as shown in the following example:

```
props = {}
props['configuration'] = 'foo'
otd_listTcpListeners(props)
```

```
listener-1
listener-2
```

You can view the properties of a TCP listener in detail by running the `otd_getTcpListenerProperties` command.

Modifying a Listener

After you create a listener, you may need to modify some of the settings such as, listener port number, IP address, protocol family, and so on.

You can modify the listener settings by using either Fusion Middleware Control or WLST as described in the following topics:

Topics

- [Modifying a Listener Using Fusion Middleware Control](#)
- [Modifying a Listener Using WLST](#)

Modifying a Listener Using Fusion Middleware Control

To modify an HTTP or TCP listener by using the Fusion Middleware Control, do the following:

1. Log in to Fusion Middleware Control, as described in [Displaying Fusion Middleware Control](#).
2. Click the WebLogic Domain button at the upper left corner of the page.
3. Select Administration > OTD Configurations.
A list of the available configurations is displayed.
4. Select the configuration for which you want to modify a HTTP or TCP Listener.
5. Click the Traffic Director Configuration In the Common Tasks pane.

6. Select Administration > Listener.

The Listeners page is displayed. It shows a list of the HTTP or TCP listeners defined for the configuration.

7. Click the name of the listener that you want to modify.

The Listener Settings page is displayed. On this page, you can do the following:

- Enable and disable the listener.
- Change the listener port number and IP address.
- For HTTP listeners: Change the server name and the default virtual server.
- For TCP listeners: Change the TCP proxy.
- If server certificates have been created for the configuration, you can enable SSL/TLS and configure SSL/TLS settings for the listener. See [Configuring SSL on a HTTP/TCP Listener](#).
- Change the protocol family—IPv4, IPv6, or SDP—for which the listener should accept requests.
- For HTTP listeners: Configure parameters to tune the performance of the virtual server—the number of acceptor threads, the listen queue size, receive buffer size, and so on. See [Tuning HTTP Listener Settings](#).

8. Specify the parameters that you want to change.

On-screen help and prompts are provided for all of the parameters.

When you change the value in a field or tab out of a text field that you changed, the **Save** button near the upper right corner of the page is enabled.

At any time, you can discard the changes by clicking the **Reset** button.

9. After making the required changes, click **Save**.

- A message, confirming that the updated listener was saved, is displayed in the Console Messages pane.
- In addition, the **Deployment Pending** message is displayed at the top of the main pane. You can either deploy the updated configuration immediately by clicking **Deploy Changes**, or you can do so later after making further changes as described in [Activating Configuration Changes](#).

Modifying a Listener Using WLST

- To change the properties of an HTTP listener, run the `otd_setHttpListenerProperties` command. For example, the following command changes the maximum requests per connection of the listener `http-listener-1` in the configuration `foo` to 1024.

```
props = {}
props['configuration'] = 'foo'
props['http-listener'] = 'http-listener-1'
props['max-requests-per-connection'] = '1024'
otd_setHttpListenerProperties(props)
```

To change the SSL/TLS settings of an HTTP listener, run the `otd_setHttpListenerSslProperties` command. For example, the following

command disables TLS 1.0 support for the listener `http-listener-1` in the configuration `foo`.

- ```
props = {}
props['configuration'] = 'foo'
props['http-listener'] = 'http-listener-1'
props['tls10'] = 'false'
otd_setHttpListenerSslProperties(props)
```

To change the properties of a TCP listener, run the `otd_setTcpListenerProperties` command. For example, the following command changes the maximum requests per connection of the listener `tcp-listener-1` in the configuration `foo` to 1024.

- ```
props = {}
props['configuration'] = 'foo'
props['tcp-listener'] = 'tcp-listener-1'
props['max-requests-per-connection'] = '1024'
otd_setTcpListenerProperties(props)
```

To change the SSL/TLS settings of an TCP listener, run the `otd_setTcpListenerSslProperties` command. For example, the following command disables TLS 1.0 support for the listener `tcp-listener-1` in the configuration `foo`.

- ```
props = {}
props['configuration'] = 'foo'
props['tcp-listener'] = 'tcp-listener-1'
props['tls10'] = 'false'
otd_setTcpListenerSslProperties(props)
```

## Deleting a Listener

You can delete listeners that are no longer required. To delete listeners, run the `otd_deleteHttpListener` or the `otd_deleteTcpListener` command.

You can delete listeners by using either Fusion Middleware Control or WLST as described in the following topics:

### Topics

- [Deleting a Listener Using Fusion Middleware Control](#)
- [Deleting a Listener Using WLST](#)

## Deleting a Listener Using Fusion Middleware Control

To delete an HTTP or TCP listener by using the Fusion Middleware Control, do the following:

1. Log in to Fusion Middleware Control, as described in [Displaying Fusion Middleware Control](#).
2. Click the WebLogic Domain button at the upper left corner of the page.
3. Select Administration > OTD Configurations.  
A list of the available configurations is displayed.
4. Select the configuration for which you want to delete an HTTP or TCP Listener.



5. Click the Traffic Director Configuration In the Common Tasks pane.
6. Select Administration > Listener.  
The Listeners page is displayed. It shows a list of the HTTP/TCP listeners defined for the configuration.
7. Click the **Delete** icon for the listener that you want to delete.  
A prompt to confirm deletion of the listener is displayed.

 **Note:**

If the HTTP listener is associated with virtual servers, the prompt shows the names of those virtual servers.

8. Click **Yes** to proceed with the deletion.  
A message is displayed in the Console Message pane confirming that the HTTP/TCP listener was deleted.

## Deleting a Listener Using WLST

- To delete an HTTP listener, run the `otd_deleteHttpListener` command, as shown in the following example:

```
props = {}
props['configuration'] = 'foo'
props['http-listener'] = 'http-listener-1'
otd_deleteHttpListener(props)
```

- To delete an TCP listener, run the `otd_deleteTcpListener` command, as shown in the following example:

```
props = {}
props['configuration'] = 'foo'
props['tcp-listener'] = 'tcp-listener-1'
otd_deleteTcpListener(props)
```

## Configuring Status Listener

You can now configure dedicated status listeners to check the status of Oracle Traffic Director instances. Using a dedicated port to serve Oracle Traffic Director status ensures that Oracle Traffic Director is available to service status requests even when it is loaded.

In addition, you can secure the status listener by configuring SSL settings for the port.

You can configure listeners by using either Fusion Middleware Control or the WLST.

## Configuring Status Listener using Fusion Middleware Control

To configure status listener by using the Fusion Middleware Control, do the following:

1. Log in to Fusion Middleware Control, as described in [Graphical User Interface-Fusion Middleware Control](#).

2. If Certificates are not present, this section will ask you to generate certificates. Click the Weblogic Domain button at the upper right corner of the page.
3. Select Administration > OTD Configurations.  
A list of the available configurations is displayed.
4. Select the configuration for which you want to configure a status listener.
5. Click the Traffic Director Configuration in the Common Tasks pane.
6. Select Advanced Configuration > Status Listener.
7. In the **General Settings** section, enable status listener by checking the **Enabled** check box and specify the details like port, IP address, and so on - that you decided earlier.
8. Click **Apply**.
  - The details of the status listener that you just configured are displayed on the **Status Listener** page.
  - The **SSL/TLS Settings** and **Advanced Settings** sections are displayed.
9. In the **SSL/TLS Settings** section, enable SSL for the listener.  
If certificates are present, this section will allow you to select a certificate and enable SSL for the listener. To generate a certificate, click **Manage Certificates**.
  - Select the **SSL/TLS** check box to enable certificates
  - In the **RSA Certificate** and **ECC Certificate** fields, select the certificates that you want to use to authenticate the server.
10. The **Advanced Settings** pane displays the SSL/TLS advanced settings. SSL/TLS Settings are available if the configuration has certificates.  
In the **SSL/TLS Settings** section, specify settings for the SSL or TLS security protocols- that you decided earlier.
11. After entering the details, click **Apply**.  
A message confirming that a status listener was configured, is displayed in the Console Messages pane.  
When you change the value in a field or tab out of a text field that you changed, the **Apply** button near the upper right corner of the page is enabled.  
At any time, you can discard the changes by clicking the **Revert** button.
12. Restart the instances of the configuration by clicking **Start Instances/Restart Instances** in the Oracle Traffic Director Configuration pane.

## Configuring Status Listener Using WLST

- Enabling/changing properties of a status listener

To enable a status listener or to change the non-SSL properties of a status listener, run the `otd_enableStatusListener` command, as shown in the example:

- To enable a status listener

```
props = {}
props['configuration'] = 'foo'
props['port'] = '12345'
otd_enableStatusListener(props)
```

- To change the non-SSL properties of a status listener

Consider a status listener which was enabled as shown in the previous example. This means that the `ip` and `family` values are `*` and `default` respectively. To re-configure the IP address and port for this status listener, run the `otd_enableStatusListener` command, as shown in the example:

```
props = {}
props['configuration'] = 'foo'
props['ip'] = '127.0.0.1'
props['port'] = '2016'
otd_enableStatusListener(props)
```

- Disabling a status listener

To disable a status listener, run the `otd_disableStatusListener` command, as shown in the example:

```
props = {}
props['configuration'] = 'foo'
otd_disableStatusListener(props)
```

- Viewing properties of status listener properties

To view the status listener properties, run the `otd_getStatusListenerProperties` command, as shown in the example:

```
props = {}
props['configuration'] = 'foo'
otd_getStatusListenerProperties(props)
```

- Changing the SSL properties of status listener

To change the SSL properties of status listener, run the `otd_setStatusListenerSslProperties` command, as shown in the example:

To disable SSL on a status listener, set the `enabled` property to `false`.

```
props = {}
props['configuration'] = 'foo'
props['enabled'] = 'false'
otd_setStatusListenerSslProperties(props)
```

- Viewing the status listener SSL properties

To view the SSL properties of a status listener, run the `otd_getStatusListenerSslProperties` command, as shown in the example:

```
props = {}
props['configuration'] = 'foo'
otd_getStatusListenerSslProperties(props)
```

# 12

## Managing Security

You can secure access to the Oracle Traffic Director administration server and enable SSL/TLS for Oracle Traffic Director virtual servers. In addition, you can configure client authentication and use Oracle Traffic Director to secure access to origin servers. This chapter contains the following sections:

- [SSL/TLS Concepts](#)
- [Managing Certificates](#)
- [Configuring SSL/TLS on Oracle Traffic Director](#)
- [Managing Certificate Revocation Lists](#)

### SSL/TLS Concepts

Before you begin administering Oracle Traffic Director security, you need to understand some basic concepts about Oracle Traffic Director security management, the set of security standards supported by Oracle Traffic Director, and the tasks involved in securing an Oracle Traffic Director domain.

This section provides basic information about security-related concepts. It contains the following topics:

- [About SSL](#)
- [About Ciphers](#)
- [About Keys](#)
- [About Certificates](#)

### About SSL

Secure Socket Layer (SSL) is a protocol for securing Internet communications and transactions. It enables secure, confidential communication between a server and clients through the use of digital certificates. Oracle Traffic Director supports Transport Layer Security (TLS) v1, v1.1, and v1.2.

In a 2-way HTTP over SSL (HTTPS) connection, each party—say a browser or a web server—first verifies the identity of the other. This phase is called the SSL/TLS handshake. After the identities are verified, the connection is established and data is exchanged in an encrypted format. The following are the steps in the SSL/TLS handshake between an SSL-enabled browser and an SSL-enabled server:

1. The browser attempts to connect to the server by sending a URL that begins with `https://` instead of `http://`.
2. The server sends its digital certificate (see [About Certificates](#)) and public key to the client.
3. The client checks whether the server's certificate is current (that is, it has not expired) and is issued by a certificate authority (CA) that the client trusts.

4. If the certificate is valid, the client generates a one-time, unique session key and encrypts it with the server's public key, and then sends the encrypted session key to the server.
5. The server decrypts the message from the client by using its private key and retrieves the session key.

At this point, the client has verified the identity of the server; and only the client and the server have a copy of the client-generated, unique session key. Till the session is terminated, the client and the server use the session key to encrypt all communication between them.

## About Ciphers

A cipher is an algorithm, a mathematical function, used for encrypting and decrypting data. Some ciphers are stronger and more secure than others. Usually, the more bits a cipher uses, the harder it is to decrypt the data encrypted using that cipher.

Clients and servers support different *cipher suites* (sets of ciphers), depending on factors such as the supported protocol, organizational policies on encryption strength, and government restrictions on export of encrypted software.

In any 2-way encryption process, the client and the server must use the same cipher suite. During the SSL/TLS handshake process, the server and client negotiate the cipher suite—typically, the strongest one—that they use to communicate.

## Cipher Suites Supported by Oracle Traffic Director

During the SSL/TLS handshake, Oracle Traffic Director and clients negotiate the cipher suites to be used. The cipher suites supported in Oracle Traffic Director are listed. You can view this list by running the `otd_getVirtualServerSslProperties WLST` command.

The name of each cipher suite indicates the key-exchange algorithm, the hashing algorithm, and the encryption algorithm, as depicted in the table.

- **Protocols supported**
  - TLS: TLS 1.0, 1.1, 1.2
- **Key exchange algorithms supported**
  - RSA
  - RSA\_EXPORT
  - RSA\_EXPORT1024
  - RSA\_FIPS
  - ECDHE\_RSA
  - ECDH\_RSA
  - ECDH\_ECDSA
  - ECDHE\_ECDSA
- **Encryption algorithms supported**
  - AES\_256\_CBC: 256-bit key

- 3DES\_EDE\_CBC: 168-bit key
- AES\_128\_CBC: 128-bit key
- RC4\_128: 128-bit key
- DES\_CBC: 56-bit key
- RC4\_40 and RC2\_CBC\_40: 128-bit key but only 40 bits have cryptographic significance
- NULL: No encryption
- **Message Authentication Code (MAC) algorithms supported**
  - SHA: 160-bit hash
  - NULL: No hashing

#### **Cipher Suites Supported in Oracle Traffic Director**

- SSL\_RSA\_WITH\_3DES\_EDE\_CBC\_SHA
- TLS\_ECDHE\_RSA\_WITH\_RC4\_128\_SHA
- TLS\_ECDHE\_RSA\_WITH\_AES\_256\_CBC\_SHA
- TLS\_ECDHE\_ECDSA\_WITH\_AES\_256\_CBC\_SHA
- TLS\_RSA\_WITH\_AES\_128\_CBC\_SHA
- TLS\_RSA\_WITH\_AES\_256\_CBC\_SHA
- TLS\_RSA\_WITH\_AES\_128\_CBC\_SHA256
- TLS\_RSA\_WITH\_AES\_256\_CBC\_SHA256
- TLS\_RSA\_WITH\_AES\_128\_GCM\_SHA256
- TLS\_RSA\_WITH\_AES\_256\_GCM\_SHA384
- TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_CBC\_SHA256
- TLS\_ECDHE\_ECDSA\_WITH\_AES\_256\_CBC\_SHA384
- TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_GCM\_SHA256
- TLS\_ECDHE\_ECDSA\_WITH\_AES\_256\_GCM\_SHA384
- TLS\_ECDHE\_RSA\_WITH\_AES\_128\_CBC\_SHA256
- TLS\_ECDHE\_RSA\_WITH\_AES\_128\_GCM\_SHA256
- TLS\_ECDHE\_RSA\_WITH\_AES\_256\_GCM\_SHA384

## About Keys

Encryption using ciphers, by itself, does not ensure data security. A key must be used with the encrypting cipher to produce the actual encrypted result, or to decrypt previously encrypted information. The encryption process uses two keys—a public key and a private key. The keys are mathematically related; so information that is encrypted using a public key can be decrypted only using the associated private key, and vice versa. The public key is published by the owner as part of a certificate (see [About Certificates](#)); only the associated private key is safeguarded.

## About Certificates

A certificate is a collection of data that uniquely identifies a person, company, or other entity on the Internet. It enables secure, confidential communication between two entities. Personal certificates are used by individuals; server certificates are used to establish secure sessions between the server and clients over SSL.

Certificates can be self-signed (by the server), signed by a trusted third party called Certification Authority (CA) or one that you created. The holder of a certificate can present the certificate as proof of identity to establish encrypted, confidential communication. The CA could be a third-party vendor or an internal department responsible for issuing certificates for an organization's servers.

Certificates are based on public-key cryptography, which uses a pair of *keys* (very long numbers) to encrypt information so that it can be read only by its intended recipient. The recipient then decrypts the information using one of the keys.

A certificate binds the owner's public key to the owner's identity. In addition to the public key, a certificate typically includes information such as the following:

- The name of the holder and other identification, such as the URL of the server using the certificate
- The name of the CA that issued the certificate
- The digital signature of the issuing CA
- The validity period of the certificate

## RSA and ECC Certificates

Oracle Traffic Director supports generation of the traditional RSA-type keys and the more advanced Elliptic Curve Cryptography (ECC) keys. ECC offers equivalent security with smaller key sizes, which results in faster computations, lower power consumption, and memory and bandwidth savings.

### Key size for RSA and ECC

- RSA keys: you can specify 2048, 4096 bits. Long keys provide better encryption, but Oracle Traffic Director would need more time to generate them.
- ECC keys: you should specify the curve for generating the key pair. Oracle Traffic Director supports the following curves: 163 (sect163r2), 192 (secp192r1), 224(secp224r1), 233(sect233k1), 256(secp256r1), 283(sect283k1), 384(secp384r1), 409(sect409k1), 521(secp521r1), 571(sect571k1).

## Managing Certificates

To obtain a digital certificate, you must generate a Certificate Signing Request (CSR) and issue it to a reputable CA. The CA returns a digital certificate that is signed with the CA's private key and that is used for establishing identity. You then import the digital certificate for identity into your identity keystore.

This section contains the following topics:

- [Obtaining a Certificate](#)

- [Generating a Keypair](#)
- [Generating a Certificate Signing Request \(CSR\)](#)
- [Importing a Certificate](#)
- [Viewing a List of Certificates](#)
- [Deleting a Certificate](#)

## Obtaining a Certificate

To enable SSL/TLS for an Oracle Traffic Director instance, you must associate an RSA or ECC certificate, or both with an Oracle Traffic Director listener or a virtual server. The certificate can be a self-signed certificate, demo CA signed certificate or a certificate issued by a third-party Certificate Authority (CA) like Verisign.

### Demo CA signed Certificate

To obtain a demo CA signed certificate, you must generate a keypair from Oracle Traffic Director. See section [Generating a Keypair](#) for information on generating a keypair. The keypair generated is signed by a demonstration CA by default. Use this keypair if you do not need your certificate to be signed by a CA. In addition, you can use this keypair if you want to test the SSL/TLS implementation while the CA is in the process of signing your certificate request.

### Third party CA signed certificate

To obtain a certificate signed by a CA, generate a keypair from Oracle Traffic Director. See section [Generating a Keypair](#). Then, generate a Certificate Signing Request (CSR) for the generated keypair, submit it to the CA, and follow the process to obtain the signed certificate. The Certificate Signing Request (CSR) is a digital file, a block of encrypted text in Base-64 encoded PEM format containing information such as your server name, organization name, and country. It also contains the public key that is included in the certificate. See section, [Generating a CSR Using Fusion Middleware Control](#) for generating the CSR from a keypair.

After obtaining the CA-signed certificate in response to your CSR, import the certificate into the appropriate configuration, as described in section [Importing a CA Signed Certificate](#).

### Self-signed Certificate

You cannot create a self-signed certificate from Oracle Traffic Director. Use tools like `orapki`, `keytool`, and `openssl`. The certificate can then be imported into Oracle Traffic Director as described here. The following commands create a self-signed certificate and import the same into Oracle Traffic Director.

#### Using openssl and java keytool

```
openssl req -x509 -newkey rsa:4096 -keyout key.pem -out cert.pem -days 365
openssl pkcs12 -export -name myservercert -in cert.pem -inkey key.pem -out
keystore.p12
keytool -importkeystore -destkeystore mykeystore.jks -srckeystore
keystore.p12 -srcstoretype pkcs12 -alias myservercert
```

```
Importing the certificate using wlst command
svc = getOpssService("KeyStoreService")
```



```
svc.importKeyStore(appStripe='OTD', name='foo', password='<keystore
password>', keypasswords='key passwords', type='OracleWallet',
permission=true, filepath='<directory containing wallet>')
```

### Using orapki

```
cd <ORACLE_HOME>/oracle_common/bin
./orapki wallet create -wallet ./myservercert -pwd <password>
./orapki wallet add -wallet ./myservercert -keysize 1024 -
dn "CN=Custom_Root_CA,O=oracle,C=US" -self_signed -validity 3650 -pwd <password>

Importing the certificate using wlst command
svc = getOpssService("KeyStoreService")
svc.importKeyStore(appStripe='OTD', name='foo', password='<keystore password>',
keypasswords='<key passwords>', type='OracleWallet', permission=true,
filepath='<directory containing wallet>')
```

## Generating a Keypair

You can generate a keypair if you do not need your certificate to be signed by a CA, or if you want to test the SSL/TLS implementation while the CA is in the process of signing your demo CA certificate.

Note that if you use a keypair to enable SSL/TLS for an Oracle Traffic Director virtual server, when a client accesses the `https://` URL of the virtual server, an error message is displayed indicating that the signing CA is unknown and not trusted. To proceed with the connection, the client can choose to trust the self-signed certificate.

You can generate a keypair by using either Fusion Middleware Control or the WLST.

## Before You Begin

Before you begin generating a keypair, decide the following:

- The nickname of the keypair (required only for generating a keypair).
- The key type—RSA or ECC.

See [RSA and ECC Certificates](#).

## Generating a Keypair Using Fusion Middleware Control

To generate a keypair by using the Fusion Middleware Control, do the following:

1. Log in to Fusion Middleware Control, as described in [Graphical User Interface-Fusion Middleware Control](#).
2. Click the WebLogic Domain button at the upper left corner of the page.
3. Select Administration > OTD Configurations.
4. Select the configuration for which you want to create an self-signed certificate.
5. Click the Traffic Director Configuration In the Common Tasks pane.
6. Select Security > Manage Certificates
7. Click the **Generate Keypair** button in the common task bar.

The **Generate Keypair** wizard appears.

**Generate Keypair**

\* Alias

\* Common name

Subject Alternate Names

Organizational Unit

Organization

City

State

Country

Key Type RSA

Key Size 2048

OK Cancel

8. Specify the Alias, Common Name and Subject Alternate Names.

 **Note:**

- Alternate names for the server can be specified using the **Subject Alternate Names** field as a comma separated list of domain names. For example, test.com,www.test.com,my.test.com
- Since many modern browsers ignore the **Common Name** field when matching hostnames, you may need to specify the domain name mentioned in the **Common Name** field as a Subject Alternate Name as well to ensure browser interoperability.

**9. Click OK.**

The new certificate is displayed in the certificate list.

- 10. View the certificate details by clicking on the certificate alias**
- The key pair is wrapped in a demonstration CA signed certificate and stored in the truststore. If your applications not using the truststore, then you must import the demonstration CA certificate to a custom keystore.

## Generating a Keypair using WLST

To generate a keypair, run the `generateKeyPair` command, as shown in the following example:

```
svc = getOpssService("KeyStoreService")
svc.generateKeyPair(appStripe='OTD', name='myconfig', password='', alias='mycert',
keypassword='', dn='CN=test.com', OU=Webtier, O='\Oracle Corporation',
ST=California, C=US', keysize='1024',
ext_san="DNS:test.com,DNS:www.test.com,DNS:my.test.com")
```

For the updated configuration to take effect, you should deploy it to the Oracle Traffic Director instances by using the `activate` command.

See `generateKeyPair` in the *WebLogic Scripting Tool Command Reference for Oracle Traffic Director*.

 **Note:**

- Alternate domain names for the server (also known as Subject Alternate Names or SAN) can be added to the keypair using the `ext_san` parameter. Ensure that you specify the correct syntax for specifying domain names.
- Since many modern browsers ignore the CN (Common Name) field when matching hostnames, you may need to add the domain name mentioned in the CN as an alternate domain name as well using the `ext_san` parameter to ensure browser interoperability.

## Generating a Certificate Signing Request (CSR)

The CSR is a digital file—a block of encrypted text in Base-64 encoded PEM format—containing information such as your server name, organization name, and country. It also contains the public key that will be included in the certificate.

You can generate a Certificate Signing Request (CSR) by using either Fusion Middleware Control or the WLST.

### Generating a CSR Using Fusion Middleware Control

To generate a CSR using the Fusion Middleware Control, do the following:

1. Log in to Fusion Middleware Control, as described in [Displaying Fusion Middleware Control](#).
2. Click the WebLogic Domain button at the upper left corner of the page.
3. Select Administration > OTD Configurations.

A list of the available configurations is displayed.

4. Select the configuration for which you want to create a CSR.
5. Click the Traffic Director Configuration In the Common Tasks pane.
6. Select Security > Manage Certificates

A new page displays on screen.

7. A list of the available certificates are displayed. Select the certificate for which you want to generate CSR.

8. Click the **Generate CSR** button in the common pane.

It opens a new window where you can export **Generate CSR**.

9. Follow the on-screen prompts to **Export CSR**, Click on **Export CSR** to have a copy of it, and click **Close**.

### Generating a CSR Using WLST

To generate a CSR, run the `exportKeyStoreCertificateRequest` command, as shown in the following example:

```
generate the CSR and put it in to a text file
svc.exportKeyStoreCertificateRequest(appStripe='OTD', name='myconfig', password='',
alias='mycert', keypassword='', filepath='/scratch/certreq.crt')
```

This command generates a CSR and displays the encrypted text of the CSR as shown in *Generating a Keypair using Fusion Middleware Control*.

For the updated configuration to take effect, you should deploy it to the Oracle Traffic Director instances by using the `activate` command. After obtaining the CA-signed certificate in response to your CSR, you should import the certificate in the appropriate configuration, as described in [Importing a Certificate](#).

## Importing a Certificate

You can import a generated keypair or CA-signed certificate by using Fusion Middleware Control or the WLST.

This section contains the following topics:

- Importing a Certificate Using Fusion Middleware Control
- Importing a Certificate Using WLST

## Importing a CA Signed Certificate

### Before you begin

The CA signed certificate that must be imported should be a base-64 encoded PEM formatted file containing all the certificates that form the certificate chain starting from server certificate, intermediate certificate and the root certificates. OTD also supports importing a certificate chain that is in base-64 encoded PEM formatted PKCS7 file (.p7b file).

### Sample PEM formatted certificate chain file

```
-----BEGIN CERTIFICATE-----
(Server SSL certificate)
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
(Intermediate certificate)
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
(Root certificate)
-----END CERTIFICATE-----
```

### Sample pkcs7 certificate chain file (.p7b file)

```
-----BEGIN PKCS7-----
[... certificate content here ...]
-----END PKCS7-----
```

### Importing a CA Signed Certificate Using Fusion Middleware Control

1. Login to Fusion Middleware Control, as described in [Displaying Fusion Middleware Control](#).
2. Click the WebLogic Domain button at the upper left corner of the page.
3. Select Administration > OTD Configurations.  
A list of the available configurations is displayed.
4. Select the configuration for which you want to import a certificate.
5. Click the Traffic Director Configuration In the Common Tasks pane.
6. Select Security > Manage Certificates  
A new page displays on screen.
7. Click the **Import** button.

The Import Certificate wizard opens.

8. Choose **Certificate Chain** from the **Certificate Type** drop-down.
9. Choose an existing (created while generating the keypair) from the **Alias** from the drop-down.
10. Specify the certificate source. If you are using the **Paste** option, then copy and paste the certificate directly into the text field. If you are using the **Select a file** option, then click **Browse** to choose the file from the operating system.
11. Click **OK**. The imported certificate or trusted certificate is displayed in the list of certificates.

## Importing a CA Signed Certificate Using WLST

To Import a Certificate Chain, run the `importKeyStoreCertificate` command, as shown in the following example. See *Importing a Certificate in Securing Applications with Oracle Platform Security Services*.

```
svc = getOpssService("KeyStoreService")
svc.importKeyStoreCertificate(appStripe='OTD', name='<configuration name>',
password='<key store password>', alias='test_cert', keypassword='<key password>',
type='CertificateChain', filepath='/export/home/testCertChain.crt')
```

## Importing an Existing Certificate

Existing certificates could be self signed certificates or certificates with keypair generated outside the current Oracle Traffic Director installation.

A self signed certificate is generated using tools like `orapki`, `keytool` and `openssl`. There is no option in Fusion Middleware Control to import these certificates, as the private key exists outside the current Oracle Traffic Director installation. This can be

done using WLST by using the `importKeyStore` command. Oracle Traffic Director supports importing KeyStores of type JKS or Oracle Wallet.

 **Note:**

Note that except for the self signed certificates, the KeyStore must contain the entire certificate chain including the root certificate in the same alias as well as the private key.

### Importing an Existing Certificate Using WLST

- **Import JKS keystore:** The following commands show how the CA signed certificate, the private key and CA certificate can be added to a JKS keystore which can then be imported into the Oracle Traffic Director configuration using WLST commands.

```
Convert the certificates and key into a pkcs12 file
servercert.pem - Base-64 encoded PEM formatted file containing containing CA
signed certificate obtained from the CA.
serverkey.pem - Private key generated while creating the certificate signing
request
cacert.pem - Base-64 encoded PEM formatted file containing all the
certificates that are needed to form the certificate chain starting from
intermediate certificates (if any) and the root certificate
cat intermediate.pem root.pem > cacert.pem

openssl pkcs12 -export -name myservercert -in servercert.pem -inkey
serverkey.pem -certfile cacert.pem -out keystore.p12

Convert pkcs12 file into jks key store
keytool -importkeystore -destkeystore mykeystore.jks -srckeystore keystore.p12 -
srcstoretype pkcs12 -alias myservercert

Import the JKS key store created above into the OTD configuration using WLST:
svc = getOpssService("KeyStoreService")

svc.importKeyStore(appStripe='OTD', name='foo', password='<keystore password>',
keypasswords='key passwords', aliases='myservercert', type='JKS',
permission=true, filepath='mykeystore.jks')
```

- **Import Oracle Wallet:** The certificate can also be imported from an Oracle wallet provided the wallet contains the CA signed certificate, the private key, the CA certificate and any intermediate certificates that are needed to form the certificate chain.

```
svc = getOpssService("KeyStoreService")
svc.importKeyStore(appStripe='OTD', name='foo', password='<keystore
password>',keypasswords='key passwords', type='OracleWallet', permission=true,
filepath='<directory containing wallet>')
```

## Importing a Trusted Certificate

When SSL/TLS is enabled for an Oracle Traffic Director virtual server and you access the `https://` URL of the virtual server, an error message is displayed indicating that the signing CA is unknown and not trusted if the signing CA is not part of trusted certificates in Oracle Traffic Director. To proceed with the connection, the client can

choose to trust the certificate or import the certificate of the signing CA as a trusted certificate. The Trusted certificate that must be imported should be a base-64 encoded PEM formatted file.

```
-----BEGIN CERTIFICATE-----
(Trusted certificate)
-----END CERTIFICATE-----
```

### Importing a Trusted Certificate Using Fusion Middleware Control

1. Log in to Fusion Middleware Control, as described in [Graphical User Interface-Fusion Middleware Control](#).
2. Click the WebLogic Domain button at the upper left corner of the page.
3. Select Administration > OTD Configurations.  
A list of the available configurations is displayed.
4. Select the configuration for which you want to import a certificate.
5. Click the Traffic Director Configuration In the Common Tasks pane.
6. Select Security > Manage Certificates  
A new page displays on screen.
7. Click the **Import** button.  
The Import Certificate wizard opens.

**Import Certificate**

Certificate Type: Trusted Certificate ▾

\* Alias: foo

Certificate Source:  Paste Certificate or Certificate Chain

\* Paste Certificate String here

Select a file that contains the Certificate or Certificate Chain

File Name

Choose File No file chosen

OK Cancel

8. Choose **Trusted Certificate** from the **Certificate Type** drop-down.
9. Specify an **Alias** name for the imported certificate from the drop-down.
10. Specify the certificate source. If using the **Paste** option, then copy and paste the certificate directly into the text field. If using the **Select a file** option, then click **Browse** to choose the file from the operating system.



11. Click **OK**. The imported certificate or trusted certificate is displayed in the list of certificates.

## Importing a Trusted Certificate Using WLST

To import a trusted certificate, run the `importKeyStoreCertificate` command, as shown in the following example.

```
svc = getOpssService("KeyStoreService")
svc.importKeyStoreCertificate(appStripe='OTD', name='<configuration name>',
password='<key store password>', alias='test_cert', keypassword='<key password>',
type='TrustedCertificate', filepath='/export/home/trustedCertificate.crt')
```

## Viewing a List of Certificates

You can view a list of the certificates installed in a configuration by using either Fusion Middleware Control or the WLST

### Viewing a List of Certificates Using Fusion Middleware Control

To view a list of the certificates installed in a configuration by using the administration console, do the following:

1. Log in to Fusion Middleware Control, as described in [Displaying Fusion Middleware Control](#)
2. Click the WebLogic Domain button at the upper left corner of the page.
3. Select Administration > OTD Configurations.  
A list of the available configurations is displayed.
4. Select the configuration for which you want to view certificates.
5. Click the Traffic Director Configuration In the Common Tasks pane.
6. Select Security > Manage Certificates  
A new page displays on screen.
7. Below the common task bar, certificates are listed.

### Viewing a List of Certificates Using WLST

- To view a list of the certificates installed in a configuration, run the `otd_listCertificates` command, as shown in the following examples.

The following command displays a list of the server certificates in the configuration.

```
props = {}
props['configuration'] = 'foo'
otd_listCertificates(props)
```

- To view the properties of a certificate, run the `getKeyStoreCertificates` command, as shown in the following example.

```
svc = getOpssService("KeyStoreService")
svc.getKeyStoreCertificates(appStripe='OTD', name='myconfig',
password='', alias='mycert')
```

## Deleting a Certificate

You can delete certificates in a configuration by using either Fusion Middleware Control or the WLST.

### Deleting a Certificate using Fusion Middleware Control

To delete a certificate using Fusion Middleware Control, do the following:

1. Log in to Fusion Middleware Control, as described in [Graphical User Interface-Fusion Middleware Control](#).
2. Click the WebLogic Domain button at the upper left corner of the page.
3. Select Administration > OTD Configurations.

A list of the available configurations is displayed.

4. Select the configuration for which you want to delete certificates.
5. Click the Traffic Director Configuration In the Common Tasks pane.
6. Select Security > Manage Certificates

A new page displays on screen.

7. Below the common task bar it displays a list of certificates.

8. Click the **Delete** button for the certificate that you want to delete.

- If one or more listeners are associated with the certificate that you are deleting, a message is displayed indicating that the certificate cannot be deleted.
- If the certificate that you are deleting is not associated with any listener, a prompt to confirm deletion of the certificate is displayed.

Click **OK** to proceed.

A message is displayed in the Console Messages pane, confirming that the certificate has been deleted.

### Deleting a Certificate Using WLST

#### Deleting a Certificate Using WLST

To delete a certificate, run the `deleteKeyStoreEntry` command.

#### Example:

```
svc = getOpssService("KeyStoreService")
svc.deleteKeyStoreEntry(appStripe='OTD', name='myconfig', password='',
alias='mycert', keypassword='')
```

If the certificate that you are deleting is associated with one or more listeners, the following message is displayed.

```
OTD-64309 Certificate 'rsa-1' is being referred by listeners: listener1,listenerN
```

You can delete the certificate forcibly by including the `--force` option.

For the updated configuration to take effect, you should deploy it to the Oracle Traffic Director instances by using the `activate` command.

## Configuring SSL/TLS on Oracle Traffic Director

Learn how to configure Oracle Traffic Director to use Secure Sockets Layer (SSL)/Transport Layer Security (TLS).

### Topics

- [Configuring SSL/TLS Between Oracle Traffic Director and Clients](#)
- [Configuring SSL/TLS Between Oracle Traffic Director and Origin Servers](#)
- [Configure SSL Termination At a Hardware Load Balancer front-ending Oracle Traffic Director](#)
- [Configure WebLogic to receive SSL information from Web Tier / Traffic Director](#)
- [Configure SSL Pass through on Oracle Traffic Director](#)

## Configuring SSL/TLS Between Oracle Traffic Director and Clients

This section describes how you can use SSL/TLS to secure communication between clients and Oracle Traffic Director instances.

To enable SSL/TLS for an OTD instance, you must associate an RSA or ECC certificate, or both, with one or more listeners of the instance.

When an HTTPS request is received, the certificate that Oracle Traffic Director sends to the client during the SSL/TLS handshake could be one of the following:

- A certificate associated with a virtual server bound to a configured HTTP listener.
- A certificate associated with the HTTP/TCP listener

## Configuring SSL on a HTTP/TCP Listener

You can configure a listener to receive HTTPS or TCP requests by using either Fusion Middleware Control or the WLST.

### Configuring SSL/TLS for a Listener Using Fusion Middleware Control

1. Log in to Fusion Middleware Control, as described in [Displaying Fusion Middleware Control](#).
2. Click the WebLogic Domain button at the upper left corner of the page.
3. Select Administration > OTD Configurations.  
A list of the available configurations is displayed.
4. Select the configuration for which you want to configure SSL/TLS-enabled listeners.
5. Click the Traffic Director Configuration In the Common Tasks pane.
6. Select Administration > Listeners.
7. Select the listener for which you want to enable and configure SSL/TLS among HTTP listeners/TCP Listeners.

The Listener Settings page is displayed.

8. Select Settings > SSL/TLS Settings.

The SSL/TLS settings page is displayed.

9. In the SSL Settings section, select the **SSL Enabled** check box.
10. In the **RSA Certificate** and **ECC Certificate** fields, select the certificates that you want to use to authenticate the server.

If you associate a listener with an RSA certificate *and* with an ECC certificate, the certificate that the server eventually presents to the client is determined during the SSL/TLS handshake, based on the cipher suite that the client and the server negotiate to use.

You can also specify the following advanced SSL/TLS settings in the Advanced Settings section of the Listener Settings page:

**Configuring Client Authentication:** Client authentication is the verification of a client by the HTTP/TCP Listener based on the certificate that the client provides. Client authentication is not enabled by default. To configure client authentication, do the following:

- a. Select Settings > Advanced Settings > SSL/TLS Settings.  
The SSL/TLS settings page is displayed.
- b. Select the required SSL/TLS Client Authentication mode.
  - Required: The server requests the client for a certificate; if the client does not provide a certificate, the connection is closed.
  - Optional: 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.
  - Disabled (default): Client authentication is disabled.
- c. Specify the Authentication Timeout and Maximum Authentication Data Parameters.

**Configuring Ciphers:** To enable and disable SSL and TLS ciphers, do the following:

- a. Select Settings > Advanced Settings > SSL/TLS Settings.  
The SSL/TLS settings page is displayed
- b. In the SSL Settings section, you can select the Ciphers from the available ciphers for the listener.

 **Note:**

You can select ciphers only if SSL/TLS is enabled for the Listener.

11. The on-screen help and prompts are provided for all parameters. When you change the value in a field or tab out of a text field that you changed, the **Apply** button near the upper right corner of the page is enabled. At any time, you can discard the changes by clicking the **Revert** button.
12. After making the required changes, click **Apply**.

A message, confirming that the updated listener was saved, is displayed in the Console Messages pane.

- Restart the instances of the configuration by clicking **Start/Restart Instances** in the Common Tasks pane.

## Configuring SSL/TLS for a Listener Using WLST

- To view the SSL/TLS properties of an HTTP or TCP listener, run the `otd_getHttpListenerSslProperties` or `otd_getTcpListenerSslProperties` command, as shown in the following example:

```
props = {}
props['configuration'] = 'foo'
props['http-listener'] = 'http-listener-1'
otd_getHttpListenerSslProperties(props)

enabled=false
client-auth=false
client-auth-timeout=60
max-client-auth-data=1048576
tls11=true
tls12=true
override-cipher-order=false
...
```

- To configure SSL/TLS for an HTTP or TCP listener, run the `otd_setHttpListenerSslProperties` or `otd_setTcpListenerSslProperties` command, as shown in the following example:

```
props['configuration'] = 'foo'
props['http-listener'] = 'http-listener-1'
props['tls10'] = 'false'
otd_setHttpListenerSslProperties(props)
```

- Configuring Client Authentication:** Client authentication is the verification of a client by the HTTP/TCP Listener based on the certificate that the client provides. Client authentication is not enabled by default. To configure client authentication for a listener, run the `otd_setHttpListenerSslProperties/ otd_setTcpListenerSslProperties` command.

```
props = {}
props['configuration'] = 'foo'
props['http-listener'] = 'bar'
props['client-auth'] = 'false'
otd_setHttpListenerSslProperties(props)
```

- Configuring Ciphers:** To view the ciphers that are currently enabled for a listener, run the `otd_getHttpListenerSslProperties/ otd_getTcpListenerSslProperties` command, as shown in the following example:

```
props = {}
props['configuration'] = 'foo'
props['http-listener'] = 'bar'
otd_getTcpListenerSslProperties(props)
```

A comma-separated list of ciphers that are currently enabled is returned in the cipher property.

- To enable or disable specific ciphers for a listener, run the `otd_setHttpListenerSslProperties/otd_setTcpListenerSslProperties` command and specify the ciphers to be enabled in the ciphers property.

- The list of supported ciphers will be listed as apart of a property "supported-ciphers", when you run the `otd_getHttpListenerSslProperties/otd_getTcpListenerSslProperties` command.

## Configuring SSL On a Virtual Server

### Configuring SSL On a Virtual Server Using Fusion Middleware Control

1. Log in to Fusion Middleware Control, as described in [Displaying Fusion Middleware Control](#).
2. Click the WebLogic Domain button at the upper left corner of the page.
3. Select Administration > OTD Configurations.  
A list of the available configurations is displayed.
4. Select the configuration for which you want to configure SSL/TLS-enabled listeners.
5. Click the Traffic Director Configuration In the Common Tasks pane.
6. Select Administration > Virtual Servers.  
The Virtual Servers page is displayed. It shows a list of the virtual servers defined for the configuration  
You can view the properties of a virtual server by clicking on its name.
7. Select the listener for which you want to enable and configure SSL/TLS.  
The Listener Settings page is displayed.
8. Select Settings > SSL/TLS Settings.  
The SSL/TLS settings page is displayed.
9. In the SSL Settings section, select the **SSL Enabled** check box.
10. In the **RSA Certificate** and **ECC Certificate** fields, select the certificates that you want to use to authenticate the server.

If you associate a listener with an RSA certificate *and* with an ECC certificate, the certificate that the server eventually presents to the client is determined during the SSL/TLS handshake, based on the cipher suite that the client and the server negotiate to use.

You can also specify the following advanced SSL/TLS settings in the Advanced Settings section of the Listener Settings page:

**Configuring Client Authentication:** Client authentication is the verification of a client by the OTD virtual server based on the certificate that the client provides. Client authentication is not enabled by default. To configure client authentication, do the following:

- a. Select Settings > Advanced Settings > SSL/TLS Settings.  
The SSL/TLS settings page is displayed.
- b. Select the required SSL/TLS Client Authentication mode.
  - Required: The server requests the client for a certificate; if the client does not provide a certificate, the connection is closed.

- Optional: 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.
  - Disabled (default): Client authentication is disabled.
- c. Specify the Authentication Timeout and Maximum Authentication Data Parameters.

**Configuring Ciphers:** To enable and disable SSL and TLS ciphers, do the following:

- a. Select Settings > Advanced Settings > SSL/TLS Settings.  
The SSL/TLS settings page is displayed
- b. In the SSL Settings section, you can select the Ciphers from the available ciphers for the listener.

 **Note:**

You can select ciphers only if SSL/TLS is enabled for the Listener.

11. The on-screen help and prompts are provided for all parameters. When you change the value in a field or tab out of a text field that you changed, the **Apply** button near the upper right corner of the page is enabled. At any time, you can discard the changes by clicking the **Revert** button.
12. After making the required changes, click **Apply**.  
A message, confirming that the updated listener was saved, is displayed in the Console Messages pane.
13. Restart the instances of the configuration by clicking **Start/Restart Instances** in the Common Tasks pane.

## Configuring SSL On a Virtual Server Using WLST

- To view the certificates that are currently associated with a virtual server, run the `otd_getVirtualServerSslProperties` command, as shown in the following example:

```
props = {}
props['configuration'] = 'foo'
props['virtual-server'] = 'bar'
otd_getVirtualServerSslProperties(props)
```

```
max-client-auth-data=1048576
```

```
client-auth=false
```

```
tls12=true
```

```
tls11=true
```

```
override-cipher-order=false
```

```
tls10=true
```

```
enabled=false
```

```
server-cert-alias=None
```

```
client-auth-timeout=60
```

```
...
```

- To associate a certificate with a virtual server, run the `otd_setVirtualServerSslProperties` command, as shown in the following example:

```
props = {}
props['configuration'] = 'foo'
props['virtual-server'] = 'bar'
props['server-cert-alias'] = 'cert-1'
otd_setVirtualServerSslProperties(props)
```

Ensure that a certificate of the same type—ECC or RSA—that you want to associate with the virtual server, is also associated with the listeners to which the virtual server is bound.

- Client authentication is the verification of a client by the OTD virtual server based on the certificate that the client provides. Client authentication is not enabled by default. To enable client authentication for an virtual server, run the `otd_setVirtualServerSslProperties`

```
props = {}
props['configuration'] = 'foo'
props['virtual-server'] = 'bar'
props['client-auth'] = 'false'
otd_setVirtualServerSslProperties(props)
```

- To view the ciphers that are currently enabled for a Virtual Server, run the `otd_getVirtualServerSslProperties` command, as shown in the following example:

```
props = {}
props['configuration'] = 'foo'
props['virtual-server'] = 'bar'
otd_getVirtualServerSslProperties(props)
```

A comma-separated list of ciphers that are currently enabled is returned in the `cipher` property.

- To enable or disable specific ciphers for a virtual server, run the `otd_setVirtualServerSslProperties` command and specify the ciphers to be enabled in the `ciphers` property.
- The list of supported ciphers will be listed as part of a property "supported-ciphers", when you run the `otd_getVirtualServerListenerSslProperties` command.

## Configuring SSL/TLS Between Oracle Traffic Director and Origin Servers

This section describes how to use SSL/TLS to secure connections between Oracle Traffic Director instances and origin servers that are Oracle WebLogic Server and Oracle HTTP Server instances. It contains the following topics:

- [About One-Way and Two-Way SSL/TLS](#)
- [Configuring One-Way SSL/TLS Between Oracle Traffic Director and Origin Servers](#)



- [Configuring Two-Way SSL/TLS Between Oracle Traffic Director and Origin Servers](#)
- [Configuring Ciphers On an Origin Server Pool](#)
- [Configure SSL Pass through on Oracle Traffic Director](#)

## About One-Way and Two-Way SSL/TLS

The connections between Oracle Traffic Director and origin servers in the back end can be secured using one-way or two-way SSL/TLS.

- **One-way SSL/TLS:** The SSL/TLS-enabled origin server presents its certificate to the Oracle Traffic Director instance. The Oracle Traffic Director instance is not configured to present any certificate to the origin server during the SSL/TLS handshake.
- **Two-way SSL/TLS:** The SSL/TLS-enabled origin server presents its certificate to the Oracle Traffic Director instance. The Oracle Traffic Director instance too presents its own certificate to the origin server. The origin server verifies the identity of the Oracle Traffic Director instance before establishing the SSL/TLS connection. Additionally, either end of the SSL/TLS connection—Oracle Traffic Director and/or origin servers—can be configured to verify the host name while exchanging certificates.

## Configuring One-Way SSL/TLS Between Oracle Traffic Director and Origin Servers

1. Import a trusted certificate. For information about importing a certificate as trusted certificate into Oracle Traffic Director, see [Importing a Trusted Certificate](#).
2. Configure Oracle Traffic Director to verify the host name in the origin server's certificate by using the command `otd_setOriginServerPoolSslProperties`, if required.

Syntax:

```
otd_setOriginServerPoolSslProperties(props)
```

Example:

```
props = {}
props['configuration'] = 'foo'
props['origin-server-pool'] = 'bar'
props['validate-server-cert-hostname'] = 'true'
otd_setOriginServerPoolSslProperties(props)
```

 **Note:**

Since Oracle Traffic Director is configured to validate the host name in the origin server's certificate during the SSL/TLS handshake, the following must be done to avoid failure. When the origin server presents its certificate, Oracle Traffic Director cannot validate the host name in the certificate, and so the SSL/TLS handshake fails.

- Ensure that the server name (CN) in the origin server's certificate matches the origin server's hostname as specified in the origin-server pool of the OTD configuration. See [Managing Origin-Server Pools](#)
- Ensure that dynamic discovery is disabled (default setting). See [Configuring an Oracle WebLogic Server Cluster as an Origin-Server Pool](#)

## Configuring Two-Way SSL/TLS

To configure two-way SSL/TLS between Oracle Traffic Director and origin servers, do the following:

1. Perform the procedure for configuring one-way SSL/TLS, as described in [Configuring One-Way SSL/TLS Between Oracle Traffic Director and Origin Servers](#).
2. Obtain a CA-issued server certificate for Oracle Traffic Director, as described in [Obtaining a Certificate](#).
3. Install the CA-issued server certificate in the Oracle Traffic Director configuration, as described in [Importing a Certificate](#).
4. Configure the required Oracle Traffic Director route with the certificate that Oracle Traffic Director should present to the origin server, by using the `otd_setOriginServerPoolSslProperties WLST`.

### Syntax:

```
otd_setOriginServerPoolSslProperties(props)
```

### Example:

```
props = {}
props['configuration'] = 'foo'
props['origin-server-pool'] = 'bar'
props['client-cert-alias'] = 'baz'
otd_setOriginServerPoolSslProperties(props)
```

5. Export the root certificate of the CA that signed the certificate for the OTD, from the keystore.

### Syntax

```
svc = getOpssService("KeyStoreService")
svc.exportKeyStoreCertificate(appStripe='<stripe>', name='<keystore>',
password='<password>', alias='<alias>', type='<entrytype>',
filepath='<absolute_file_path>')
```

`svc` is the service command object obtained through a call to `getOpssService()`, `appstripe` is the name of the stripe containing the keystore, `name` is the name of the keystore, `password` is the keystore password

`alias` is the alias of the entry to be exported

`type` is the type of keystore entry to be exported. Valid values are **Certificate**, **Trusted Certificate**, and **Certificate Chain**

`filepath` is the absolute path of the file where certificate, trusted certificate or certificate chain is exported

Example:

```
svc = getOpssService("KeyStoreService")
svc.exportKeyStoreCertificate(appStripe='OTD', name='myconfig', password='',
alias='rootca', keypassword='', type='TrustedCertificate', filepath='/scratch/
rootcert.txt')
```

For more information about the `exportKeyStoreCertificate` command, run the following command:

```
svc = getOpssService("KeyStoreService")
svc.help('exportKeyStoreCertificate')
```

6. Import the root certificate that you exported in the previous step into the trust keystore of the origin servers.
7. Configure the origin servers to require Oracle Traffic Director to present its client certificate during the SSL/TLS handshake.
  - **For Oracle WebLogic Server origin servers:**

Perform the procedure described in *Configure two-way SSL* in the *Oracle WebLogic Server Fusion Middleware Control Online Help*.

 **Note:**

By default, host name verification is enabled in Oracle WebLogic Server. For information about disabling host name verification, see *Disable host name verification* in the *Oracle WebLogic Server Fusion Middleware Control Online Help*.

- **For Oracle HTTP Server origin servers:**

Add the following directive in the `httpd.conf` file: `SSLVerifyClient require`

## Configuring Ciphers On an Origin Server Pool

You can configure Ciphers on an Origin Server Pool by using either Fusion Middleware Control or WLST, as specified in the topics below:

### Configuring Ciphers On an Origin Server Pool Using Fusion Middleware Control

1. Log in to Fusion Middleware Control, as described in [Graphical User Interface-Fusion Middleware Control](#).
2. Click the WebLogic Domain button at the upper left corner of the page.
3. Select Administration > OTD Configurations.

A list of the available configurations is displayed.

4. Select the configuration for which you want to configure ciphers.
5. Click the Traffic Director Configuration In the Common Tasks pane.
6. Select Administration > Origin Server Pool.

The Origin Server Pools page is displayed. It shows a list of the Origin Server Pools defined for the configuration.

7. Select the Origin Server Pool for which you want to configure ciphers.  
The Origin Server Pool Settings page is displayed.
8. Select Settings > Advanced Settings > SSL/TLS Settings.

The SSL/TLS settings page is displayed. You can select ciphers only if SSL/TLS is enabled for the Origin Server Pool.

9. In the SSL Settings section, you can manage the Certificates.
10. When you change the value in a field or tab out of a text field that you changed, the **OK** button near the upper right corner of the page is enabled.

At any time, you can discard the changes by clicking the **Cancel** button.

11. After making the required changes, click **OK**.

A message, confirming that the updated listener was saved, is displayed in the Console Messages pane.

### Configuring Ciphers On an Origin Server Pool Using WLST

- To view the ciphers that are currently enabled for a Origin Server Pool, run the `otd_getOriginServerPoolSslProperties` command, as shown in the following example:

```
props = {}
props['configuration'] = 'foo'
props['origin-server-pool'] = 'bar'
otd_getOriginServerPoolSslProperties(props)
```

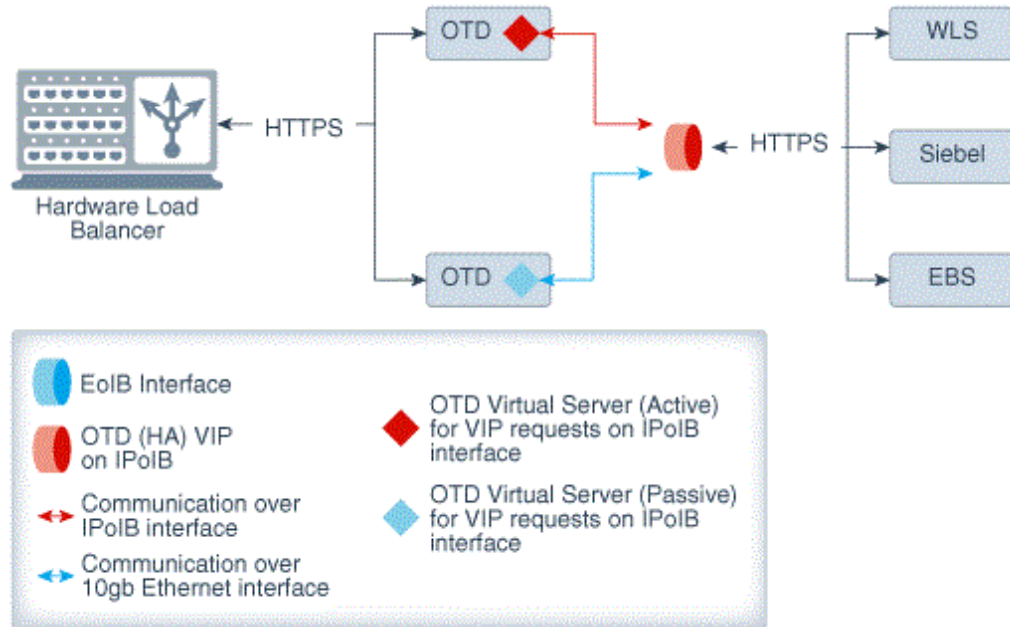
A comma-separated list of ciphers that are currently enabled is returned in the `ciphers` property.

- To enable or disable specific ciphers for a virtual server, run the `otd_setOriginServerPoolSslProperties` command and specify the ciphers to be enabled in the `ciphers` property.
- The list of supported ciphers are listed as apart of a property, `supported-ciphers`, when you run the `otd_getOriginServerPoolSslProperties` command.

## Configure SSL Termination At a Hardware Load Balancer front-ending Oracle Traffic Director

This section lists the configuration steps necessary to handle this use case where customers want to terminate SSL at an external hardware load balancer front ending Oracle Traffic Director.

In a typical production deployment topology (as shown below), customers terminate SSL within the (external) hardware load balancer and have only HTTP communication within the internal load balancer (reverse proxy) and application tiers.



In this scenario, customers need to perform a few additional steps within the internal load balancer / reverse proxy solution as well as the application tiers so that any request redirect can happen correctly. This section will focus on the steps necessary within the internal load balancer (Oracle Traffic Director) / reverse proxy solution.

1. Configure OTD to pass SSL information.
  - a. Log in to Fusion Middleware Control, as described in [Displaying Fusion Middleware Control](#).
  - b. Click the WebLogic Domain button at the upper left corner of the page.
  - c. Select Administration > OTD Configurations.  
A list of the available configurations is displayed.
  - d. Select the configuration for which you want to configure ciphers.
  - e. Click the Traffic Director Configuration In the **Common Tasks** pane.
  - f. Select Administration > Virtual Servers.  
The **Virtual Servers** page is displayed, with a list of the Virtual Servers defined for the configuration.
  - g. Select the Virtual Server for which you want to configure ciphers.  
The **Virtual Server Settings** page is displayed.
  - h. Click on **Routes** tab and select the route that needs to be changed.
  - i. Select **Advanced Settings**, and do the following:
    - Update Route.
    - Disable OTD from rewriting any 'Location' or 'Content-Location' headers by changing 'Rewrite-headers' to be empty. This is important when SSL termination is on at an external load balancer.
    - Uncheck all SSL related headers sent to the origin server.

- j. After making the required changes, click **OK**.

A message, confirming that the updated listener was saved, is displayed in the **Console Messages** pane.

At the end of the above step, OTD now does NOT rewrite any headers (typically this happens when the application tier redirects the request to another location) and can pass the external hardware load balancer provided SSL information, via the incoming request headers within the incoming requests to origin servers.

2. Configure F5-BigIP to send SSL information via headers to OTD.

If you choose to terminate SSL within the hardware load balancer such as F5-BigIp, then you will need to configure this hardware load balancer to explicitly send a specific header - WL-Proxy-SSL - to OTD and WLS.

This procedure explains how to configure F5-BigIp to send this header (Source: F5 Big IP product documentation).

- a. On the Main tab, expand Local Traffic, and then click Profiles. The HTTP Profiles screen opens.
- b. In the upper right portion of the screen, click the Create button. The New HTTP Profile screen opens.
- c. In the Name box, type a name for this profile. In our example, we type bea-ssl.
- d. From the Parent Profile list, select http-acceleration if you are using the WebAccelerator. If you are not using the WebAccelerator, select http-wan-optimized-compression-caching.
- e. In the Request Header Insert row, check the Custom box. In the box, type: WL-Proxy-SSL:true.
- f. In the Redirect Rewrite row, check the Custom box. From the Redirect Rewrite list, select Match.
- g. Modify any of the other settings as applicable for your network. In our example, we leave the settings at their default levels.
- h. Click **Finish**.

If you are also using OTD to front-end Oracle Access Manager (OAM/IDM), then you will need to configure your hardware load balancer to additionally insert header 'IS\_SSL:ssl' to the incoming request headers.

## Configure WebLogic to receive SSL information from Web Tier / Traffic Director

When Oracle Traffic Director is terminating a connection, the attribute 'WebLogic Plug-In Enabled' in WebLogic Server Fusion Middleware Control can be set to 'true'. This way, Oracle Traffic Director communicates the certificate information to the applications deployed on the WebLogic Server. An application can then validate for specific information in the certificate, such as key size or cipher, before allowing the clients to access the application. This flag can be tuned on either within a specific WebLogic Managed Server or at WebLogic cluster level. This ensures that WebLogic is able to appropriately rewrite the headers when SSL termination happens within Traffic Director / Web Tier.

1. Open `http://<wls-admin-hostname>:7001/console` and login with weblogic user/ password.
2. Within WLS Managed Server, 'Configuration->General' settings tab, expand 'Advanced Settings' and scroll down to check "Weblogic Plugin Enabled" to 'Yes' and click "Save".
3. If you have Weblogic cluster enabled, then we recommend doing this at cluster level.

## Configure SSL Pass through on Oracle Traffic Director

Oracle Traffic Director when configured for TCP load balancing with a TCP proxy and TCP listeners can be used for SSL pass-through for any TCP based protocol including HTTPS. In this case, the requests are handled as generic TCP connections and SSL will be terminated at the origin servers for HTTPS/LDAPS/FTPS/T3S, and so on.

## Managing Certificate Revocation Lists

A Certificate Revocation List (CRL) is a list that a CA publishes to inform users about certificates that the CA has decided to revoke before they expire. CRLs are updated periodically; the updated CRLs can be downloaded from the CA's website.

To ensure that Oracle Traffic Director servers do not trust server certificates that have been revoked by CA, you should download the latest CRLs from the CAs' websites regularly and install them in your Oracle Traffic Director configurations.

You can install CRLs manually. You can also configure Oracle Traffic Director to take the downloaded CRLs from a specified directory and install them automatically at specified intervals.

### Related Topics

- [Installing CRLs Manually Using Fusion Middleware Control](#)
- [Update CRLs Automatically](#)

## Installing and Deleting CRLs Manually

You can install and delete CRLs manually by using either Fusion Middleware Control or the WLST.

- [Installing CRLs Manually using Fusion Middleware Control](#)
- [Installing and Deleting CRLs Manually Using WLST](#)

## Installing CRLs Manually Using Fusion Middleware Control

To install a downloaded CRL by using the Fusion Middleware Control, do the following:

1. Log in to Fusion Middleware Control, as described in [`](#).
2. Click the WebLogic Domain button at the upper left corner of the page.
3. Select Administration > OTD Configurations.

A list of the available configurations is displayed.

4. Select the configuration for which you want to install a certificate.
5. Click the Traffic Director Configuration In the Common Tasks pane.
6. Select Security > Certificate Revocation List  
A new page displays on screen.
7. Click the Install CRL button.  
The Install Certificate Revocation List dialog box is displayed.
8. Specify the location of the downloaded CRL file, and click Install CRL.
  - A message, confirming successful installation of the CRL, is displayed in the Console Messages pane.
  - The CRL that you just installed is displayed on the Certificate Authorities page.

## Installing and Deleting CRLs Manually Using WLST

- To install a downloaded CRL, run the `otd_installCrl` command, as shown in the following example:

```
props = {}
props['configuration'] = 'foo'
props['file-path'] = '/export/ServerSign.crl'
otd_installCrl(props).
```

- To view a list of the installed CRLs in a configuration, run the `otd_listCrls` command, as shown in the following example:

```
props = {}
props['configuration'] = 'foo'
otd_listCrls(props)

"Class 1 Public Primary Certification Authority" "Sat Apr 15 16:59:59 PDT 2000"
"VeriSign Class 3 Code Signing 2010 CA" "Mon Aug 29 14:00:03 PDT 2011"
"VeriSign Class 3 Organizational CA" "Sun May 18 13:48:16 PDT 2014"
```

- To delete a CRL, run the `otd_deleteCrl` command, as shown in the following example:

```
props = {}
props['configuration'] = 'foo'
props['issuer'] = 'CN=GlobalSign ServerSign CA,OU=ServerSign CA,O=GlobalSign nv-
sa,C=BE'
otd_deleteCrl(props)
```

When you delete a CRL, it is removed from the Oracle Traffic Director configuration *and* from the directory in which the downloaded CRL was stored.

For the updated configuration to take effect, you should deploy it to the Oracle Traffic Director instances by using the `activate` command.

## Update CRLs Automatically

You can configure Oracle Traffic Director to periodically take downloaded CRL files from a specified directory and install them automatically by using either Fusion Middleware Control or the WLST.

At the specified interval, Oracle Traffic Director looks for updated CRL files in the specified directory.



- If Oracle Traffic Director detects new CRL files, it installs them in the configuration and logs a message in the server log.
- If existing CRL files have been changed, Oracle Traffic Director installs the updated CRL files in the configuration and logs a message in the server log.
- If Oracle Traffic Director detects that previously installed CRL files have been removed from the directory, it deletes the CRLs from the configuration and logs a message in the server log.
- If no changes are detected in the CRL directory, Oracle Traffic Director does not perform any update.

## Configuring Oracle Traffic Director to Install CRLs Automatically Using Fusion Middleware Control

To configure Oracle Traffic Director to install CRLs automatically using the Fusion Middleware Control, do the following:

1. Log in to Fusion Middleware Control, as described in [Graphical User Interface-Fusion Middleware Control](#)
2. Click **WebLogic Domain**.
3. Select Administration > OTD Configurations.
4. Click the name of the configuration that you want to set up to install CRLs automatically.
5. Click the Traffic Director Configuration In the Common Tasks pane.
6. Select Security > Certificate Revocation List .
7. Click the **Install CRL** button.
8. Specify the location of the downloaded CRL file, and click **Install CRL**.
  - A message, confirming successful installation of the CRL, is displayed in the Console Messages pane.
  - The CRL that you just installed is displayed on the Certificate Authorities page.
9. Go to the **Advanced Settings** section of the page.
10. In the **CRL Update Event** field, enter the absolute path to the directory that contains the updated CRL files.
11. Click **New Event**
12. Specify the interval or time of the day at which the CRLs should be updated, and then click **OK**.
  - A message, confirming creation of the event, is displayed in the Console Messages pane.
  - The new event is displayed in the CRL Update Events list.
    - New events are enabled by default. To change the status, select the **Enable/Disable** check box.
    - To delete an event, click the **Delete** button.

## Configuring Oracle Traffic Director to Install CRLs Automatically Using WLST

To configure Oracle Traffic Director to install CRLs automatically using WLST, do the following:

Schedule an event for Oracle Traffic Director to take the downloaded CRLs from the specified directory and install them automatically, by using the `otd_createEvent` command.

For example, the following command specifies that the CRLs for the configuration `foo` should be updated after every 12:00.

```
props = {}
props['configuration'] = 'foo'
props['event'] = 'event-1'
props['command'] = 'update CRL'
props['time'] = '12:00'
otd_createEvent(props)
```

See the `otd_createEvent` command in the *WebLogic Scripting Tool Command Reference for Oracle Traffic Director*.

# 13

## Managing Logs

Oracle Traffic Director records data about server events such as configuration changes, instances being started and stopped, errors while processing requests, and so on in log files. You can use the logs to diagnose server problems, evaluate server usage patterns, and tune the system for improved performance.

This chapter contains the following sections:

- [About the Oracle Traffic Director Logs](#)
- [Viewing Logs](#)
- [Configuring Log Preferences](#)
- [About Log Rotation](#)
- [Rotating Logs Manually](#)
- [Configuring Oracle Traffic Director to Rotate Logs Automatically](#)

### About the Oracle Traffic Director Logs

Each Oracle Traffic Director instance, including the administration server, has two logs—an access log and a server log. You can enable access and server logs for each virtual server in the instance.

The default location of the access log and server log for an Oracle Traffic Director instance is the `DOMAIN_HOME/servers/instance-name/logs` directory.

In addition to the access and server logs, there are instance logs that are enabled by default and initialized when the instance is started for the first time.

This section provides an overview of the access and server logs. For information about changing log settings, including the name and location of log files, see [Configuring Log Preferences](#).

### Access Log

The access log contains information about requests to, and responses from, the server. The default name of the access log file is `access.log`.

The following example shows the first three lines in a typical access log:

```
format=%Ses->client.ip% - %Req->vars.auth-user% [%SYSDATE%] "%Req->reqpb.clf-request
%" %Req->srvhdrs.clf-status% %Req->srvhdrs.content-length% %Req->vars.ecid%
10.177.243.207 - - [28/Aug/2011:23:28:30 -0700] "GET / HTTP/1.1" 200 4826 -
10.177.243.207 - - [28/Aug/2011:23:28:31 -0700] "GET / HTTP/1.1" 200 916 -
```

The first line indicates the access log format. The second and third lines are the actual entries.

You can change the access log format, file name, and location. You can also disable the access log. See [Configuring Log Preferences](#).

## Server Log

The server log contains data about lifecycle events—server start-up, shut down, and restart; configuration updates; and so on. It also contains errors and warnings that the server encountered. The default name of the server log file is `server.log`.

The following line is an example of an entry in a server log.

```
[2011-10-03T02:04:59.000-07:00] [net-soa] [NOTIFICATION] [OTD-10358] []
 [pid: 11722] http-listener-1: http://example.com:1904 ready to accept requests
```

The default server-log level is `NOTIFICATION:1`, at which only major lifecycle events, warnings, and errors are logged.

You can change the log level, the log file name, and the log file location. See [Configuring Log Preferences](#).

[Table 13-1](#) lists the log levels that you can specify for the server log.

**Table 13-1 Server Log Levels**

| Log Level                                                              | Description                                                                                             |
|------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------|
| <code>INCIDENT_ERROR:1</code>                                          | A serious problem caused by unknown reasons. You should contact Oracle for support.                     |
| <code>ERROR:1</code><br><code>ERROR:16</code><br><code>ERROR:32</code> | A serious problem that requires your immediate attention.                                               |
| <code>WARNING:1</code>                                                 | A potential problem that you should review.                                                             |
| <code>NOTIFICATION:1 (default)</code>                                  | A major lifecycle event, such as a server being started or restarted.                                   |
| <code>TRACE:1</code><br><code>TRACE:16</code><br><code>TRACE:32</code> | Trace or debug information to help you or Oracle Support diagnose problems with a particular subsystem. |

The number following each log level indicates the severity of the logged event on the scale 1–32. An `ERROR:1` message is of higher severity than an `ERROR:16` message.

`TRACE:32` is the most verbose log level and `INCIDENT_ERROR:1` is the least verbose. Enabling the `TRACE` log levels might affect performance, because of the high volume of messages that are logged. Therefore, avoid enabling verbose log levels in production systems, except in situations when you need more detailed information to debug issues.

## Viewing Logs

After creating logs for virtual servers, you can view the list of logs for each virtual server. To view a list of logs, run the `displayLogs` command.

You can view the access and server logs of Oracle Traffic Director instances and virtual servers by using either Fusion Middleware Control or the WLST.

 **Note:**

- Besides using WLST or Fusion Middleware Control, you can also use the standard operating-system commands such as `ls` and `more` to list and view the log files.
- The Log Viewer in Fusion Middleware Control and the `displayLogs` CWLST command display only the log entries that currently exist in the access log file, TCP access log, and error log on the disk. They do not display items from the access-log buffer (see [Configuring Access-Log Buffer Settings](#))).

## Viewing Logs Using Fusion Middleware Control

View log data for a node, an instance, or virtual server within an instance by using the Fusion Middleware Control,

1. Log in to Fusion Middleware Control, as described in [Displaying Fusion Middleware Control](#).
2. Click the **WebLogic Domain** button at the upper left corner of the page..
3. Select Administration > OTD Configurations.  
A list of the available configurations is displayed.
4. Select the configuration for which you want to view logs.
5. Click the Traffic Director Configuration In the Common Tasks pane.
6. Select Administration > Logging.
7. The Oracle Traffic Director Log Viewer window is displayed.
8. A list of the **Server Log Preferences, Access Log Preference, and TCP Access Log Preferences** Tabs are displayed.
  - To view the server log, select the **Server Log Preference** tab.
  - To view the access log, select the **Access Log Preference** tab.
  - To view the TCP access log, select the **TCP Access Log Preference** tab.

## Viewing Logs Using WLST

To view the access log for an instance or a virtual server, run the `displayLogs` command.

For example, the following command displays the access-log records for the instance of the configuration `foo`.

```
displayLogs(target="sc:otd_foo_machine1")
```

## Configuring Log Preferences

When you create a configuration, the server and access logs are enabled with certain default settings. You can change the server log level, file name, and location. You can

change the access log format, file name, and location. You can also disable the access log. If you change the location of the server log, you should restart the instance for the change to take effect.

The log preferences defined in a configuration are applicable to all the virtual servers in the configuration. At the virtual-server level, you can define the access-log location and format, and the server-log location.

You can configure log preferences for Oracle Traffic Director instances by using either Fusion Middleware Control or the WLST as described in the following topics:

## Configuring Log Preferences Using Fusion Middleware Control

Configure log preferences for a configuration or a virtual server by using the Fusion Middleware Control

1. Log in to Fusion Middleware Control, as described in [Displaying Fusion Middleware Control](#).
2. Click the WebLogic Domain button at the upper left corner of the page.
3. Select Administration > OTD Configurations.  
A list of the available configurations is displayed.
4. Select the configuration for which you want to configure log preferences.
5. Click the Traffic Director Configuration In the Common Tasks pane.
6. Select Administration > Logging.
7. The Oracle Traffic Director Log Viewer window is displayed.
8. A list of the **Server Log Preferences, Access Log Preference, and TCP Access Log Preferences** Tabs are displayed.
  - To view the server log, select the **Server Log Preference** tab.
  - To view the access log, select the **Access Log Preference** tab.
  - To view the TCP access log, select the **TCP Access Log Preference** tab.
9. Specify the parameters that you want to change in the each Tab.  
On-screen help and prompts are provided for all of the parameters.  
For information about specifying a custom access-log format, see *Using the Custom Access-Log Format* in the *Configuration File Reference for Oracle Traffic Director* .  
When you change the value in a field or tab out of a text field that you changed, the **Apply** button near the upper right corner of the page is enabled.  
At any time, you can discard the changes by clicking the **Revert** button.
10. After making the required changes, click **Apply**.

## Configuring Log Preferences Using WLST

- To view the current access-log preferences for a configuration or a virtual server, run the `getConfigurationAccessLogProperties` Or `otd_getVirtualServerAccessLogProperties` commands.

For example, the following command displays the access-log preferences for the configuration `foo`.

```
props = {}
props['configuration'] = 'foo'
otd_getConfigurationAccessLogProperties(props)

log-file=${DOMAIN_HOME}/servers/${INSTANCE_NAME}/logs/access.log
format=%Ses->client.ip% - %Req->vars.auth-user% %SYSDATE% "%Req->reqpb.clf-
request%" %Req->srvhdrs.clf-status% %Req->srvhdrs.content-length% %Req->vars.ecid
% %Req->vars.origin-server%
default-access-log-format=%Ses->client.ip% - %Req->vars.auth-user% %SYSDATE%
"%Req->reqpb.clf-request%" %Req->srvhdrs.clf-status% %Req->srvhdrs.content-length
% %Req->vars.ecid% %Req->vars.origin-server%
```

- To set or change access-log preferences for a configuration or a virtual server, run the `setConfigurationAccessLogProperties` or `otd_setVirtualServerAccessLogProperties` commands.

For example, the following command changes the location of the access log for the configuration `foo` to `logs/access.log`.

```
props = {}
props['configuration'] = 'foo'
props['log-file'] = 'logs/access.log'
otd_setConfigurationAccessLogProperties(props)
```

For information about specifying a custom access-log format, see *Using the Custom Access-Log Format* in the *Configuration File Reference for Oracle Traffic Director*.

- To disable the access log for a virtual server, run the `otd_disableVirtualServerAccessLog` command, as shown in the following example:

```
props = {}
props['configuration'] = 'foo'
props['virtual-server'] = 'bar'
otd_disableVirtualServerAccessLog(props)
```

- To view the current server-log preferences for a configuration, run the `otd_getLogProperties` command.

For example, the following command displays the server-log preferences for the configuration `soa`.

```
props = {}
props['configuration'] = 'foo'
otd_getLogProperties(props)

log-stdout=true
log-stderr=true
log-virtual-server-name=false
create-console=false
log-to-console=true
log-to-syslog=false
log-level=NOTIFICATION:1
log-file=../logs/server.log
```

- To set or change server-log preferences for a configuration, run the `otd_setLogProperties` command. Note that if you change the location of the server log, you should restart the instance for the change to take effect.

For example, the following command changes the server-log level for the configuration `foo` to `TRACE:32`.

```
props = {}
props['configuration'] = 'foo'
props['log-level'] = 'TRACE:32'
otd_setLogProperties(props)
```

- To view the current access-log preferences for a configuration or a virtual server, run the `getConfigurationAccessLogProperties` or `otd_getVirtualServerAccessLogProperties` commands.

For example, the following command displays the access-log preferences for the configuration `foo`.

```
props = {}
props['configuration'] = 'foo'
otd_getConfigurationAccessLogProperties(props)

log-file=${DOMAIN_HOME}/servers/${INSTANCE_NAME}/logs/access.log
format=%Ses->client.ip% - %Req->vars.auth-user% %SYSDATE% "%Req->reqpb.clf-
request%" %Req->srvhdrs.clf-status% %Req->srvhdrs.content-length% %Req->vars.ecid
% %Req->vars.origin-server%
default-access-log-format=%Ses->client.ip% - %Req->vars.auth-user% %SYSDATE%
"%Req->reqpb.clf-request%" %Req->srvhdrs.clf-status% %Req->srvhdrs.content-length
% %Req->vars.ecid% %Req->vars.origin-server%
```

- To set or change access-log preferences for a configuration or a virtual server, run the `setConfigurationAccessLogProperties` or `otd_setVirtualServerAccessLogProperties` commands.

For example, the following command changes the location of the access log for the configuration `foo` to `logs/access.log`.

```
props = {}
props['configuration'] = 'foo'
props['log-file'] = 'logs/access.log'
otd_setConfigurationAccessLogProperties(props)
```

For information about specifying a custom access-log format, see *Using the Custom Access-Log Format* in the *Configuration File Reference for Oracle Traffic Director*.

- To disable the access log for a virtual server, run the `otd_disableVirtualServerAccessLog` command, as shown in the following example:

```
props = {}
props['configuration'] = 'foo'
props['virtual-server'] = 'bar'
otd_disableVirtualServerAccessLog(props)
```

- To view the current server-log preferences for a configuration, run the `otd_getLogProperties` command.

For example, the following command displays the server-log preferences for the configuration `soa`.

```
props = {}
props['configuration'] = 'foo'
otd_getLogProperties(props)
```

```
log-stdout=true
```



```
log-stderr=true
log-virtual-server-name=false
create-console=false
log-to-console=true
log-to-syslog=false
log-level=NOTIFICATION:1
log-file=../logs/server.log
```

- To set or change server-log preferences for a configuration, run the `otd_setLogProperties` command. Note that if you change the location of the server log, you should restart the instance for the change to take effect.

For example, the following command changes the server-log level for the configuration `foo` to `TRACE:32`.

```
props = {}
props['configuration'] = 'foo'
props['log-level'] = 'TRACE:32'
otd_setLogProperties(props)
```

## About Log Rotation

You can configure Oracle Traffic Director to automatically rotate (archive) the logs at specified intervals. You can also rotate the logs manually whenever required.

When the logs are rotated, the old log files are renamed with a suffix indicating the rotation date (in the `yyyymmdd` format) and 24-hour time (in the `hhmm` format). For example, the file name of the server log archive created at 11 p.m. on August 25, 2017 would be `server-201108252300.log`.

After log rotation, the server and access logs are re-initialized.

### Note:

Rotate Access Log event will also rotate TCP access logs.

## Rotating Logs Manually

You can rotate the logs manually whenever require. The server saves the old log files and marks the saved files with a name that includes the date and time when they were saved.

You can rotate the server and access logs of Oracle Traffic Director instances manually by using either Fusion Middleware Control or the WLST as described in the following topics:

## Rotating Logs Manually Using Fusion Middleware Control

Rotate logs by using the Fusion Middleware Control

1. Log in to Fusion Middleware Control, as described in [Displaying Fusion Middleware Control](#)..

2. Click the WebLogic Domain button at the upper left corner of the page.
3. Select Administration > OTD Configurations.
4. Select the configuration for which you want to rotate logs.
5. Click the Traffic Director Configuration In the Common Tasks pane.
6. Select Administration > Log Rotation.
7. If you want Oracle Traffic Director to run a specific command on the rotated log files, specify the absolute path to the required command in the **Archive Command** field
  - a. Log in to Fusion Middleware Control, as described in [Displaying Fusion Middleware Control](#).
  - b. Click the WebLogic Domain button at the upper left corner of the page.
  - c. Select Administration > OTD Configurations.

A list of the available configurations is displayed.
  - d. Select the configuration for which you want to rotate logs.
  - e. Click the Traffic Director Configuration In the Common Tasks pane.
  - f. Select Administration > Log Rotation.
  - g. The Oracle Traffic Director Log Rotation window is displayed.
  - h. If you want Oracle Traffic Director to run a specific command on the rotated log files, specify the absolute path to the required command in the **Archive Command** field
    - i. For example, if you specify `/usr/bin/gzip` as the archive command, after rotating the logs, Oracle Traffic Director compresses the rotated log files by running the following commands:

```
$ /usr/bin/gzip access-yyyyymmddhhmm.log
$ /usr/bin/gzip server-yyyyymmddhhmm.log
```
    - ii. Click **Rotate Logs Now**.

The server and access logs, including any virtual server-specific logs, for all the instances of the configuration are archived.

To rotate logs for a specific instance of the selected configuration, do the following:

      - i. In the navigation pane, select **Instances**.

The Instances page is displayed.
      - ii. Click the **Rotate Logs** button for the required instance.

The server and access logs, including any virtual server-specific logs, for the selected instance are archived.

A message is displayed in the Console Messages pane confirming that the logs were rotated.

## Rotating Logs Manually Using WLST

To rotating logs manually using WLST, do the following:

To rotate logs for an instance, run the `otd_rotateLog` command. For example, the following command rotates the access and server logs for the `otd_foo_machine1` instance.

```
props = {}
props['instance'] = 'otd_foo_machine1'
otd_rotateLog(props)
```

#### Note:

If you want Oracle Traffic Director to run a specific command on the rotated log files, specify the absolute path to the required command by running the `otd_setLogProperties` command and specifying the `archive-command` property, as shown in the following example:

```
props = {}
props['configuration'] = 'foo'
props['archive-command'] = '/usr/bin/gzip'
otd_setLogProperties(props)
```

In this example, after rotating the logs, Oracle Traffic Director compresses the rotated log files by running the following commands:

```
$ /usr/bin/gzip access-yyyyymmddhhmm.log
$ /usr/bin/gzip server-yyyyymmddhhmm.log
```

## Configuring Oracle Traffic Director to Rotate Logs Automatically

You can configure Oracle Traffic Director to rotate logs automatically at specified times or intervals by creating log-rotation events.

You can create log-rotation events by using either Fusion Middleware Control or the WLST as described in the following topics:

### Creating Log-Rotation Events Using Fusion Middleware Control

To create log-rotation events by using the Fusion Middleware Control, do the following:

1. Log in to Fusion Middleware Control, as described in [Graphical User Interface-Fusion Middleware Control](#).
2. Click the WebLogic Domain button at the upper left corner of the page.
3. Select Administration > OTD Configurations.  
A list of the available configurations is displayed.
4. Select the configuration for which you want to rotate logs.
5. Click the Traffic Director Configuration In the Common Tasks pane.
6. Select Administration > Log Rotation.
7. The Oracle Traffic Director Log Rotation window is displayed.

8. If you want Oracle Traffic Director to run a specific command on the rotated log files, specify the absolute path to the required command in the **Archive Command** field.

For example, if you specify `/usr/bin/gzip` as the archive command, after rotating the logs, Oracle Traffic Director compresses the rotated log files by running the following commands:

```
$ /usr/bin/gzip access-yyyyymmddhhmm.log
$ /usr/bin/gzip server-yyyyymmddhhmm.log
```

9. Click **Create**.

The New Log Rotation Event dialog box is displayed.

10. Specify whether the event is for the server log or the access log.
11. Specify the interval or time of the day at which the log should be updated, and then click **OK**.
  - A message, confirming creation of the event, is displayed in the Console Messages pane.
  - The new event is displayed in the Log Rotation Events list.
    - New events are enabled by default. To change the status, select the **Enable/Disable** check box.
    - To delete an event, click the **Delete** button.

## Creating Log-Rotation Events Using WLST

To create log-rotation events, run the `otd_createEvent` command.

For example, the following commands configure Oracle Traffic Director to rotate the access logs and server logs for all instances of the configuration `foo` at 12pm.

```
props = {}
props['configuration'] = 'foo'
props['event'] = 'event-1'
props['command'] = 'rotate-log'
props['time'] = '12:00'
otd_createEvent(props)

props = {}
props['configuration'] = 'foo'
props['event'] = 'event-1'
props['command'] = 'rotate-access-log'
props['time'] = '12:00'
otd_createEvent(props)
```

# Managing Event Notifications

Oracle Traffic Director allows users to subscribe to notifications about events that it detects. To receive such notifications, you can provide an HTTP endpoint URL of your choice. Oracle Traffic Director sends information about the events that it has detected through a HTTP POST message to this URL.

The HTTP POST request contains event information in JSON format. Oracle Traffic Director supports notifications for the following two events:

- [Origin server status change event](#)
- [Request limit exceeded event](#)

## Origin server status change event

Learn about how Oracle Traffic Director sends notifications to configured HTTP endpoint URL when it detects a change in the origin server status.

The origin server status change event is said to occur when one of the two events occur:

- OTD marks the origin server as `offline`.
- OTD marks the origin server back as `online` from `offline`.

A notification message is sent when there is a status change of any origin server that is part of the configuration. You cannot receive notifications for a particular origin server of interest while subscribing to the notifications.

Multiple notifications can be sent when there is a change in the status of an origin server that is part of multiple origin server pools or multiple Oracle Traffic Director instances.

## Subscribing to origin server status event using Fusion Middleware Control

1. Log in to Fusion Middleware Control for Traffic Director, as described in [Graphical User Interface-Fusion Middleware Control](#).
2. Click the WebLogic Domain button at the upper left (or) right corner of the page.
3. Select Administration > OTD Configurations.  
A list of available OTD Configurations is displayed.  
Select a configuration for which Event Subscriptions will be enabled.
4. Select Configuration > Advanced Configuration > Event Subscriptions.  
The **Events Subscriptions** page is displayed.
5. In the Common Tasks pane, click **Create** under Event Subscription.  
The New Event Subscription wizard opens.

Follow the on-screen prompts to complete the creation of the Event Subscription by using the details- Name of the subscription, URL, and so on - that you decided earlier.

6. Click **OK** on the Results screen.

After the Event Subscription is created, the Results screen of the New Event Subscription wizard displays a message confirming the successful creation of the Event Subscription.

## Subscribing to origin server status change event Using WLST

- Creating an event subscription

To create an event subscription, run the `otd_createEventSubscription` command, as shown in the example.

```
props = {}
props['configuration'] = 'foo'
props['event-subscription'] = 'bar'
props['url'] = 'http://example.com:7777/subscriber'
otd_createEventSubscription(props)
```

The first command subscribes to the URL: `http://example.com:7777/subscriber`

- Viewing a list of event subscriptions

To view a list of subscribed event subscriptions, run the `otd_listEventSubscriptions` command.

For example, to display the event subscriptions scheduled for instances of the configuration:

```
props = {}
props['configuration'] = 'foo'
otd_listEventSubscriptions(props)
```

- Deleting an event subscription

To delete an event subscription, run the `otd_deleteEventSubscription` command, as shown in the example.

```
props = {}
props['configuration'] = 'foo'
props['event-subscription'] = 'bar'
otd_deleteEventSubscription(props)
```

- Setting an event subscription properties

When you create an event subscription, it is enabled automatically.

The command `otd_setEventSubscriptionProperties` with 'enabled' as 'false' can be used to disable event subscriptions.

To disable an event, set the `enabled` property to false.

```
props = {}
props['configuration'] = 'foo'
props['event-subscription'] = 'bar'
props['enabled'] = 'true'
otd_setEventSubscriptionProperties(props)
```

- Getting an event subscription properties

The command `otd_getEventSubscriptionProperties` with `'enabled'` as `'true'` must be used to get the event subscription properties.

To enable an event, set the `enabled` property to `true`:

```
props = {}
props['configuration'] = 'foo'
props['event-subscription'] = 'bar'
otd_getEventSubscriptionProperties(props)
```

To know the details of the command and its each parameter/option, use `help`, as shown below:

```
help('otd_createEventSubscription')
```

## Notification format

When a notification to a subscribed URL is sent, the Content-type header value is set to `application/json`. When an event occurs, the OTD sends HTTP POST to the endpoint with a message body that contains a JSON document with the name/value pairs as described in this section.

**Table 14-1 JSON properties common to all events**

| JSON property              | Description                                                              |
|----------------------------|--------------------------------------------------------------------------|
| <code>event-type</code>    | Type of the event. Value: <code>origin-server-status-change</code> .     |
| <code>domain-name</code>   | Name of the domain where Oracle Traffic Director instance is configured. |
| <code>instance-name</code> | Name of the Oracle Traffic Director instance.                            |
| <code>timestamp</code>     | The time when the event occurred and detected by OTD.                    |

**Table 14-2 JSON property specific to origin-server-status-change event**

| JSON property          | Description                                                                        |
|------------------------|------------------------------------------------------------------------------------|
| <code>pool-name</code> | Origin server pool name to which the origin server is associated with.             |
| <code>host</code>      | Origin server host for which the status being sent.                                |
| <code>port</code>      | Origin server port.                                                                |
| <code>status</code>    | Online or offline.                                                                 |
| <code>reason</code>    | Reason for Oracle Traffic Director marking the origin server as offline or online. |
| <code>protocol</code>  | Health check protocol used.                                                        |

## JSON Schema

```
{
 "$schema": "http://json-schema.org/draft-04/schema#",
 "id": "",
 "type": "object",
 "properties": {
 "v1.0": {
 "id": "/v1.0",
```

```
"type": "object",
"properties": {
 "event-type": {
 "id": "/v1.0/event-type",
 "type": "string"
 },
 "domain-name": {
 "id": "/v1.0/domain-name",
 "type": "string"
 },
 "instance-name": {
 "id": "/v1.0/instance-name",
 "type": "string"
 },
 "origin-server": {
 "id": "/v1.0/origin-server",
 "type": "object",
 "properties": {
 "pool-name": {
 "id": "/v1.0/origin-server/pool-name",
 "type": "string"
 },
 "host": {
 "id": "/v1.0/origin-server/host",
 "type": "string"
 },
 "port": {
 "id": "/v1.0/origin-server/port",
 "type": "integer"
 },
 "health-check": {
 "id": "/v1.0/origin-server/health-check",
 "type": "object",
 "properties": {
 "protocol": {
 "id": "/v1.0/origin-server/health-check/protocol",
 "type": "string"
 },
 "status": {
 "id": "/v1.0/origin-server/health-check/status",
 "type": "string"
 },
 "reason": {
 "id": "/v1.0/origin-server/health-check/reason",
 "type": "string"
 }
 }
 },
 "required": [
 "status"
]
 }
 },
 "required": [
 "host",
 "port",
 "health-check"
]
},
"timestamp": {
 "id": "/v1.0/timestamp",
 "type": "string"
}
```



```

 }
 },
 "required": [
 "event-type",
 "domain-name",
 "instance-name",
 "origin-server",
 "timestamp"
]
}
},
"required": [
 "v1.0"
]
}

```

## Example

Content-Type: application/json

Content:

```

{
 "v1.0": {
 "event-type": "origin-server-status-change",
 "domain-name": "base_domain",
 "instance-name": "otdl",
 "origin-server": {
 "pool-name": "testpool",
 "host": "slc06cdz",
 "port": 7777,
 "health-check": {
 "protocol": "HTTP",
 "status": "offline",
 "reason": "Server not reachable"
 }
 }
 },
 "timestamp": "Mon, 18 Apr 2016 04:34:23 -07:00"
}

```

## Error handling

When a subscriber responds with an error code, Oracle Traffic Director:

- Logs the following **warning** message to the server log:
  - Event Dispatcher: response code <http\_response\_code> received from subscriber <subscription\_url>
  - Event Dispatcher: unable to post event <json\_notification\_data> to subscriber <subscription\_url>
  - Discards the event.

When a subscriber is not reachable, Oracle Traffic Director:

- Tries re posting the event three times
- Logs the following **warning** message to the server log:

- Event Dispatcher: unable to receive response from subscriber <subscription\_url>
- Event Dispatcher: unable to post event <json\_notification\_data> to subscriber <subscription\_url>
- Discards the event.

## Request limit exceeded event

A request limit exceeded event occurs when a request limit configured for a virtual server is exceeded. OTD checks if a request limit was exceeded in the specified interval of time, `event-notification-interval` and sends a notification message to the configured HTTP endpoint.

The `event-notification-interval` is a configurable parameter set by the user for a request limit, while subscribing to notifications from it. A notification message is sent for every request limit that exceeds the configured request limit. A request limit is identified using the name provided while configuring the request limit.

If the request limit uses the `monitor` attribute to monitor requests, the notification message will contain a JSON array of all monitors that exceeded the thresholds.

When the virtual server exceeds the specified request limit, OTD rejects all subsequent requests that match the monitor attribute. The notification message will include information on the number of requests rejected for each monitor.

### Note:

When a request limit is exceeded, OTD does not send a notification immediately.

When there is a high burst of traffic, request limit may exceed multiple times at close intervals of time. If OTD sends a notification message every time the request limit exceeds, there would be many notification messages. To avoid this, OTD checks if a request limit was exceeded in an interval of time and sends a notification message if it detects that the limit was exceeded.

## Subscribing to Request Limit Exceeded Event Using WLST

- Create an event subscription. See [Subscribing to origin server status change event Using WLST](#).
- Enabling request limit exceeded event

To enable events for a specified request limit, run the `otd_enableRequestLimitEvents` command, as shown in the example.

```
props = {}
props['configuration'] = 'foo'
props['virtual-server'] = 'bar'
props['request-limit'] = 'request-limit-1'
props['event-notification-interval'] = '60'
otd_enableRequestLimitEvents(props)
```

- Disabling request limit exceeded event

To disable events for a specified request limit, run the `otd_disableRequestLimitEvents` command, as shown in the example.

```
props = {}
props['configuration'] = 'foo'
props['virtual-server'] = 'bar'
props['request-limit'] = 'request-limit-1'
otd_disableRequestLimitEvents(props)
```

- Viewing request limit exceeded event properties

To view the properties of a specified request limit, run the `otd_getRequestLimitProperties` command.

## Notification format

POST messages sent to the endpoint has a message body that contains a JSON document with the name/value pairs described in this section.

**Table 14-3 JSON properties common to all events**

| JSON property | Description                                                              |
|---------------|--------------------------------------------------------------------------|
| event-type    | Type of the event. Value: <code>request-limit-exceeded</code> .          |
| domain-name   | Name of the domain where Oracle Traffic Director instance is configured. |
| instance-name | Name of the Oracle Traffic Director instance.                            |

**Table 14-4 JSON properties specific to notifications from request-limit-exceeded**

| JSON property         | Description                                                                                                                                                                                                                                                                                                                                                                                                           |
|-----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| time-begin            | Timestamp that indicates the begin of the event notification interval.                                                                                                                                                                                                                                                                                                                                                |
| time-end              | Timestamp that indicates the end of the event notification interval.                                                                                                                                                                                                                                                                                                                                                  |
| virtual-server        | Name of the virtual server for which request-limiting is enabled.                                                                                                                                                                                                                                                                                                                                                     |
| request-limit-rule    | Identifies the request limit rule that generated this notification message.                                                                                                                                                                                                                                                                                                                                           |
| monitor               | Value of the request attribute that is being monitored.<br>For example, if <code>monitor=\$ip</code> is specified in the <code>-request-limit</code> rule, the JSON property <code>monitor</code> will be set to the value of the <code>"\$ip"</code> variable, the Client IP address.<br>If the <code>monitor</code> is not specified in the request-limit rule, this property will be set to <code>unnamed</code> . |
| total-queue-overflows | Total requests rejected due to queue overflow.                                                                                                                                                                                                                                                                                                                                                                        |
| total-queue-timeouts  | Total requests rejected due to timeout while waiting in the queue.                                                                                                                                                                                                                                                                                                                                                    |

## JSON schema

```
{
 "$schema": "http://json-schema.org/draft-04/schema#",
 "id": "/",
 "type": "object",
 "properties": {
 "v1.0": {
 "id": "v1.0",
 "type": "object",
 "properties": {
 "event-type": {
 "id": "event-type",
 "type": "string"
 },
 "domain-name": {
 "id": "domain-name",
 "type": "string"
 },
 "instance-name": {
 "id": "instance-name",
 "type": "string"
 },
 "event-notification-interval": {
 "id": "event-notification-interval",
 "type": "object",
 "properties": {
 "time-begin": {
 "id": "time-begin",
 "type": "string"
 },
 "time-end": {
 "id": "time-end",
 "type": "string"
 }
 }
 },
 "virtual-server": {
 "id": "virtual-server",
 "type": "string"
 },
 "request-limit-rule": {
 "id": "request-limit-rule",
 "type": "string"
 },
 "monitors": {
 "id": "monitors",
 "type": "array",
 "items": {
 "id": "0",
 "type": "object",
 "properties": {
 "monitor": {
 "id": "monitor",
 "type": "string"
 },
 "total-rejects": {
 "id": "total-rejects",
 "type": "object",

```

```

 "properties": {
 "total-queue-overflows": {
 "id": "total-queue-overflows",
 "type": "integer"
 },
 "total-queue-timeouts": {
 "id": "total-queue-timeouts",
 "type": "integer"
 }
 }
 },
 "required": [
 "event-type",
 "domain-name",
 "instance-name",
 "event-notification-interval",
 "virtual-server",
 "request-limit-rule",
 "monitors"
]
},
"required": [
 "v1.0"
]
}

```

## Example

```

{
 "v1.0": {
 "event-type": "request-limit-exceeded",
 "domain-name": "base domain",
 "instance-name": "otd1",
 "event-notification-interval": {
 "time-begin": "Mon, 18 Apr 2016 04:34:23 -07:00",
 "time-end": "Mon, 18 Apr 2016 04:35:23 -07:00"
 },
 "virtual-server": "1.example.com",
 "request-limit-rule": "request-limit-1",
 "monitors": [
 {
 "monitor": "16.181.76.89",
 "total-rejects": {
 "total-queue-overflows": 2,
 "total-queue-timeouts": 3
 }
 }
]
 }
}

```

# 15

## Managing Failover Groups

Learn about how to configure the Oracle Traffic Director instances in a failover group in the active-passive and active-active modes.

This section contains the following topics:

- [Creating Failover Groups](#)
- [Managing Failover Groups](#)

### Creating Failover Groups

Learn how to implement a high available pair of Oracle Traffic Director instances by creating failover groups.

#### Before You Begin

- Decide the unique **VIP address** that you want to assign to the failover group.
  - The VIP addresses should belong to the same subnet as that of the nodes in the failover group.
  - The VIP addresses must be accessible to clients.
- Identify the instances that would be part of the failover group. The nodes should be in the same subnet.
- Identify the **network interface** for each node.

A network interface, on the node where instance is running, upon which the VIP must be managed. The first network interface that results in a match is used as the network interface for the VIP.

For this comparison, depending on whether the VIP specified for the failover group is an IPv4 or IPv6 address, the administration server considers only those network interfaces on the host that are configured with an IPv4 or IPv6 address, respectively.

- You can bind to a VIP IP address within the HTTP listener by performing a system configuration that allows you to bind to a non-existing address, as a sort of forward binding. Perform one of the following system configurations:

```
echo 1 > /proc/sys/net/ipv4/ip_nonlocal_bind
```

or,

```
sysctl net.ipv4.ip_nonlocal_bind=1 (change in /etc/sysctl.conf to keep after a reboot)
```

Make sure that the IP addresses of the listeners in the configuration for which you want to create a failover group are either an asterisk (\*) or the same address as the VIP. Otherwise, requests sent to the VIP will not be routed to the virtual servers.

- Ensure that the router ID for each failover group is unique.

If the router ID is not specified, the default router ID will be a random number between 1 and 255.

## Creating Failover Groups Using Fusion Middleware Control

To create a failover group by using the Fusion Middleware Control, do the following:

1. Log in to Fusion Middleware Control, as described in [Graphical User Interface-Fusion Middleware Control](#).
2. Click the WebLogic Domain button at the upper left corner of the page.
3. Select Administration > OTD Configurations.

A list of the available configurations is displayed.

4. Select the configuration for which you want to create a failover group.
5. Click the Traffic Director Configuration In the Common Tasks pane.
6. Select Administration > Failover Groups.

The Failover Groups page is displayed. It shows a list of the Failover Groups defined for the configuration.

7. Click **Create**.

The New Failover Group wizard is displayed.

8. Follow the on-screen prompts to complete creation of the failover group by using the details—virtual IP address, network interface, primary, backup instances, and so on—that you decided earlier.

After the failover group is created, the Results screen of the New Failover Group wizard displays a message confirming successful creation of the failover group.

9. Click **Close** on the Results screen.

The details of the failover group that you just created are displayed on the Failover Groups page.

## Creating Failover Groups Using WLST

To create a failover group, run the `otd_createFailoverGroup` command.

For example, the following command creates an active-passive failover group with the following details:

- Configuration: `ha`
- Primary instance: `1.example.com`
- Backup instance: `2.example.com`
- Virtual IP address: `192.0.2.1`

```
props = {}
props['configuration'] = 'ha'
props['virtual-ip'] = '192.0.2.1'
props['primary-instance'] = '1.example.com'
props['backup-instance'] = '2.example.com'
props['primary-nic'] = 'eth0'
props['backup-nic'] = 'eth0'
```

```
props['failover-type'] = 'active-passive'
otd_createFailoverGroup(props)
```

 **Note:**

After creating a failover group you must run `otd_startFailover` on those machines as a `root` user. This is to manually start the failover. If this command is not executed, failover will not start and there will be no high availability.

For example, the following command creates an active-active failover group with the following details:

- Configuration: `ha`
- Primary instance: `1.example.com`
- Backup instance: `2.example.com`
- Virtual IP address: `192.0.2.1`

```
props = {}
props['configuration'] = 'ha'
props['virtual-ip'] = '192.0.2.1'
props['failover-type'] = 'active-active'
otd_createFailoverGroup(props)
```

## Managing Failover Groups

After creating failover groups, you can list them, view their settings, change the primary instance for a failover group, switch the primary and backup instances, and delete them.

To manage the failover groups, the failover daemon needs to run as a privileged user (typically `root`), `otd_startFailover` command should be executed as a privileged user on the machines on which the primary and backup instances of the failover group run. Similarly to stop the daemon, you should run the `otd_stopFailover`. The configuration parameters for the `keepalived` daemon are stored in a file named `keepalived.conf` in the `config` directory of each instance that is part of the failover group. See `otd_startFailover` and `otd_stopFailover` commands in the *WebLogic Scripting Tool Command Reference for Oracle Traffic Director*.

You can view, modify, and delete failover groups by using either Fusion Middleware Control or the WLST. Note that to change the VIP or any property of a failover group, you should delete the failover group and create it afresh.

### Managing Failover Groups Using Fusion Middleware Control

To view, modify, and delete failover groups by using the Fusion Middleware Control, do the following:

1. Log in to Fusion Middleware Control, as described in [Graphical User Interface-Fusion Middleware Control](#).
2. Click the WebLogic Domain button at the upper left corner of the page.



3. Select Administration > OTD Configurations.  
A list of the available configurations is displayed.
4. Select the configuration for which you want to manage failover groups.
5. Click the Traffic Director Configuration In the Common Tasks pane.
6. Select Administration > Failover Groups.
7. The Failover Groups page is displayed. It shows a list of the Failover Groups defined for the configuration.
  - To view the properties of a failover group, click its virtual IP.
  - To switch the hosts for the primary and backup nodes, click the **Toggle Primary** button. In the resulting dialog box, click **OK**.
  - To delete a failover group, click the **Delete** button. In the resulting dialog box, click **OK**.
8. If you add or remove a failover instance from an active-active failover group, ensure to stop and start the failover group on all nodes to see the changes.

 **Note:**

- If you want to assign a different primary or backup instance in a failover group, you should create the failover group afresh.
- There can be a maximum of 255 failover groups *across configurations*.

### Managing Failover Groups Using WLST

For example, run the `otd_listFailoverGroups` command, for list of failover groups:

```
props = {}
props['configuration'] = 'ha'
otd_listFailoverGroups(props)
```

For example, run the `otd_toggleFailovergroupPrimary` command, for toggle a failover group:

```
props = {}
props['configuration'] = 'ha'
props['virtual-ip'] = '10.128.67.44'
otd_toggleFailovergroupPrimary(props)
```

For example, run the `otd_getFailoverGroupProperties` command, for change properties of a failover group:

```
props = {}
props['configuration'] = 'ha'
props['primary-instance'] = '1.example.com'
otd_getFailoverGroupProperties(props)
```

For example, run the `otd_deleteFailoverGroup` command, for deleting a failover group:

```
props = {}
props['configuration'] = 'ha'
```

```
props['virtual-ip'] = '10.128.67.44'
otd_deleteFailoverGroup(props)
```

### WLST commands specific to active-active HA

For example, run the `otd_addFailoverInstance` command, for adding a failover instance:

```
props = {}
props['configuration'] = 'ha'
props['virtual-ip'] = '10.128.67.44'
props['instance'] = '1.example.com'
props['nic'] = 'eth0'
otd_addFailoverInstance(props)
```

For example, run the `otd_removeFailoverInstance` command, for removing a failover instance:

```
props = {}
props['configuration'] = 'ha'
props['virtual-ip'] = '10.128.67.44'
props['instance'] = '1.example.com'
otd_removeFailoverInstance(props)
```

For example, run the `otd_listFailoverInstances` command, for the list of failover instances:

```
props = {}
props['configuration'] = 'ha'
props['virtual-ip'] = '10.128.67.44'
otd_listFailoverInstances(props)
```

For example, run the `otd_setFailoverInstanceOrder` command, for changing the failover instance order.

```
props = {}
props['configuration'] = 'ha'
props['virtual-ip'] = '10.128.67.44'
props['instances'] = '1.example.com, 2.example.com'
otd_setFailoverInstanceOrder(props)
```

# 16

## Monitoring Oracle Traffic Director Instances

Learn about the monitoring capabilities of Oracle Traffic Director.

Oracle Traffic Director records statistics about server activity at different levels—instances, virtual servers, listeners, connections, and origin servers. For example, for each instance of a configuration, Oracle Traffic Director collects statistics about the duration for which the instance has been running, number of requests processed, volume of data received and sent, number of responses that the instance sent of each type, average load, and so on. Similarly for each virtual server in an instance, Oracle Traffic Director collects statistics about the number of requests processed, volume of data received and sent, and the number of responses of each type. For a full list of the metrics that Oracle Traffic Director collects, see [Metrics Tracked by Oracle Traffic Director](#).

This chapter contains the following sections:

- [Methods for Monitoring Oracle Traffic Director Instances](#)
- [Configuring Statistics-Collection Settings](#)
- [Configuring URI Access to Statistics Reports](#)
- [Viewing Statistics Using WLST](#)
- [Viewing stats-xml and perfdump Reports Through a Browser](#)
- [Monitoring Using SNMP](#)
- [Monitoring Using DMS](#)
- [Sample XML \(stats-xml\) Report](#)
- [Sample Plain-Text \(perfdump\) Report](#)

### Methods for Monitoring Oracle Traffic Director Instances

[Table 16-1](#) summarizes the methods that you can use to view statistical data about an instance of a configuration and about individual virtual servers within an instance.

**Table 16-1 Methods for Monitoring Oracle Traffic Director Instances**

| Monitoring Method                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  | Requirements                                                                                                            | Advantages                                                                                         |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------|
| <p><b>WLST</b></p> <ul style="list-style-type: none"> <li>View runtime statistics for various subsystems for an instance:<br/>To view in plain-text format:<br/><code>otd_getPerfDump</code><br/>To view in XML format:<br/><code>otd_getStatsXml</code></li> <li>Display runtime statistics about all instances, or a specific instance, from metric tables collected by DMS:<br/><code>displayMetricTables</code></li> </ul> <p>See <a href="#">Viewing Statistics using WLST</a>, and <a href="#">Monitoring Using DMS</a>.</p> | <p>Administration server must be running.</p>                                                                           | <p>Enabled by default.<br/>Accessible even when request-processing threads are hanging.</p>        |
| <p><b>Browser</b></p> <ul style="list-style-type: none"> <li>Detailed statistics for a specific virtual server in XML format</li> <li>Summary report for a specific virtual server in plain-text format</li> </ul> <p>See <a href="#">Viewing stats-xml and perfdump Reports Through a Browser</a>.</p>                                                                                                                                                                                                                            | <p>Must be enabled and configured explicitly.<br/>See <a href="#">Configuring URI Access to Statistics Reports</a>.</p> | <p>The administration server need not be running. It is sufficient if the instance is running.</p> |
| <p><b>SNMP</b></p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 | <p>Must be configured explicitly.<br/>See <a href="#">Monitoring Using SNMP</a>.</p>                                    | <p>Statistics available through network management systems.</p>                                    |

## Configuring Statistics-Collection Settings

When you create an Oracle Traffic Director configuration, statistics collection is enabled by default, with five seconds as the update interval. You can disable, enable, and configure statistics collection

### Configuring Statistics-Collection Settings Using Fusion Middleware Control

To configure statistics-collection settings by using the Fusion Middleware Control, do the following:

1. Log in to Fusion Middleware Control, as described in [Graphical User Interface-Fusion Middleware Control](#).
2. Click the WebLogic Domain button at the upper left corner of the page.
3. Select Administration > OTD Configurations.  
A list of the available configurations is displayed.
4. Select the configuration for which you want to configure statistics-collection settings.

5. Click the Traffic Director Configuration In the Common Tasks pane.
6. Select Administration > virtual server.  
The Virtual Servers page is displayed. It shows a list of the virtual servers defined for the configuration.
7. Select the name of the virtual server you want to configure.
8. Select Settings > Monitoring.
9. Go to the **Statistics Collection** section of the page.
10. Specify the parameters that you want to change.

**Note:**

When deciding the statistics-collection interval, remember that frequent collection of statistics affects performance.

On-screen help and prompts are provided for all of the parameters.

When you change the value in a field or tab out of a text field that you changed, the **Apply** button near the upper right corner of the page is enabled.

At any time, you can discard the changes by clicking the **Revert** button.

11. After making the required changes, click **Apply**.
  - A message, confirming that the updated configuration was saved, is displayed in the Console Messages pane.

**Configuring Statistics-Collection Settings Using WLST**

- To view the current statistics-collection properties, run the `otd_getStatsProperties` command, as shown in the following example:

```
props = {}
props['configuration'] = 'foo'
otd_getStatsProperties(props)
```

- To configure statistics-collection properties, run the `otd_setStatsProperties` command.

For example, the following command changes the interval at which statistics are updated for the configuration to 10 seconds.

```
props = {}
props['configuration'] = 'foo'
props['interval'] = '10'
otd_setStatsProperties(props)
```

## Configuring URI Access to Statistics Reports

As described in [Methods for Monitoring Oracle Traffic Director Instances](#), in addition to viewing activity statistics by using WLST, you can view the following reports through a URI.

- `stats-xml`: Detailed statistics in XML format. For a sample, see [Sample XML \(stats-xml\) Report](#).

- `perfdump`: A summary report in plain-text format containing a subset of the data in the `stats-xml` report. For a sample, see [Sample Plain-Text \(perfdump\) Report](#). Note that though you enable the `perf-dump` report at the virtual-server level, the data in the report is aggregated at the instance level.

### Relative Advantages of URI-Based and WLST Access to Statistics Reports

- The administration server need not be running for users to access the `stats-xml` and `perfdump` reports through URIs. When compared with accessing statistics by using WLST, accessing URI-based reports involves lower processing overhead.
- Access to statistics by using WLST is enabled by default, but to view statistics through the browser, you should explicitly enable URI-based reporting and specify the URIs at which users can access the reports.

You can configure URI-based reporting of statistics by using either Fusion Middleware Control or the WLST.

### Configuring URI Access to Statistics Using Fusion Middleware Control

To configure URI-based reporting by using the Fusion Middleware Control, do the following:

1. Log in to Fusion Middleware Control, as described in [Graphical User Interface-Fusion Middleware Control](#).
2. Click the WebLogic Domain button at the upper left corner of the page.
3. Select Administration > OTD Configurations.  
A list of the available configurations is displayed.
4. Select the configuration for which you want to configure URI-based reports.
5. Click the Traffic Director Configuration In the Common Tasks pane.
6. Select Administration > virtual server.  
The Virtual Servers page is displayed. It shows a list of the virtual servers defined for the configuration.
7. Select the name of the virtual server you want to configure.
8. Select Settings > Monitoring.
9. Select Settings > Advanced Settings
10. Go to the **Monitoring** section of the page.
  - To enable URI-based reporting in XML format, select the **XML Report** check box and specify a valid URI.
  - To enable URI-based reporting in plain-text format, select the **Plain Text Report** check box and specify a valid URI for the report.On-screen help and prompts are provided for all of the parameters.  
When you change the value in a field or tab out of a text field that you changed, the **Apply** button near the upper right corner of the page is enabled.  
At any time, you can discard the changes by clicking the **Revert** button.
11. After making the required changes, click **Apply**.
  - A message, confirming that the updated configuration was saved, is displayed in the Console Messages pane.

## Configuring URI Access to Statistics in XML Format Using WLST

- To view the current XML reporting settings, run the `otd_getStatsXmlProperties` command, as shown in the following example:

```
props = {}
props['configuration'] = 'foo'
props['virtual-server'] = 'bar'
otd_getStatsXmlProperties(props)
```

```
enabled=false
uri=/stats-xml
```

- To enable and configure URI-based XML reporting, run the `otd_enableStatsXml` command.

For example, the following command enables URI-based statistics reporting in XML format for the virtual server `bar` in the configuration `foo` and specifies that the report should be available at the URI `/stats`.

```
props = {}
props['configuration'] = 'foo'
props['virtual-server'] = 'bar'
props['uri'] = '/stats'
otd_enableStatsXml(props)
```

- To disable URI-based XML reporting, run the `otd_disableStatsXml` command, as shown in the following example:

```
props = {}
props['configuration'] = 'foo'
props['virtual-server'] = 'bar'
otd_disableStatsXml(props)
```

## Configuring URI Access to Statistics in Plain-Text Format Using WLST

- To view the plain-text reporting settings, run the `otd_getPerfDumpProperties` command, as shown in the following example:

```
props = {}
props['configuration'] = 'foo'
props['virtual-server'] = 'bar'
otd_getPerfDumpProperties(props)
```

- To enable and configure the plain-text reporting, run the `otd_enablePerfDump` command.

For example, the following command enables URI-based statistics reporting in plain-text format for the virtual server `bar` in the configuration `foo` and specifies that the report should be available at the URI `/perf`.

```
props = {}
props['configuration'] = 'foo'
props['virtual-server'] = 'bar'
props['uri'] = '/perf'
otd_enablePerfDump(props)
```

- To disable URI-based plain-text reporting, run the `otd_disablePerfDump` command, as shown in the following example:

```
props = {}
props['configuration'] = 'foo'
props['virtual-server'] = 'bar'
otd_disablePerfDump(props)
```

## Viewing Statistics Using WLST

By using WLST, you can view statistics for one or all instances of a configuration.

- To view detailed statistics for an instance in XML format, run the `otd_getStatsXml` command, as shown in the following example:

```
props = {}
props['instance'] = 'otd_foo_machine1'
otd_getStatsXml(props)
```

For a sample of the report, see [Sample XML \(stats-xml\) Report](#).

- To view a summary of the statistics for an instance in plain-text format, run the `otd_getPerfDump` command, as shown in the following example:

```
props = {}
props['instance'] = 'otd_foo_machine1'
otd_getPerfDump(props)
```

For a sample of the report, see [Sample Plain-Text \(perfdump\) Report](#).

- To view statistics for one or all instances of a configuration using metric tables collected from Oracle Dynamic Monitoring Service (DMS) for Oracle Traffic Director, run the `displayMetricTables` command, as shown in the following examples. For more details about Oracle Dynamic Monitoring Service (DMS), see [Monitoring Using DMS](#)

To view metrics for all Oracle Traffic Director instances:

```
displayMetricTables('OTD_*')
```

To view origin server metrics for all instances:

```
displayMetricTables('OTD_OriginServer')
```

To get a list of metric tables for a specific instance:

```
displayMetricTableNames(servers='/OTD/otd_test_myserver.example.com')
```

To view all metrics for a specific instance:

```
displayMetricTables(servers='/OTD/otd_test_myserver.example.com')
```

To view instance metrics for a specific instance:

```
displayMetricTables('OTD_Instance', servers='/OTD/otd_test_myserver.example.com')
```

## Viewing stats-xml and perfdump Reports Through a Browser

If you enable URI access to statistics as described in [Configuring URI Access to Statistics Reports](#), you can access the `stats-xml` and `perfdump` reports through a browser by using the following URL:

```
http://host:port/uri
```

`host` and `port` are the IP address (or host name) and port number of the virtual server for which you enabled URI access to statistics. `uri` is the location that you specified while enabling URI access. Note that if a virtual server is associated with multiple



listeners, you can use the address `host:port` of any of the listeners to access the URI-based reports.

- For example, if `/perfdump` is the configured URI for the plain-text report for the virtual server `soa.example.com:1904`, the URL that you should use to access the report would be the following:

```
http://soa.example.com:1904/perfdump
```

In the URL, you can also specify the interval, in seconds, after which the browser should refresh the `perfdump` report automatically, as shown in the following example:

```
http://soa.example.com:1904/perfdump?refresh=5
```

- Similarly, if `/stats-xml` is the configured URI for the XML report for the virtual server `soa.example.com:1904`, the URL that you should use to access the XML report would be the following:

```
http://soa.example.com:1904/stats-xml
```

You can limit the data that the XML report provides by specifying a URL query string indicating the elements that should not be displayed. If you do not include a query string, all the elements in the XML report are displayed.

For example, the query string specified in the following URL suppresses display of the `virtual-server` and `server-pool` elements in the XML report.

```
http://soa.example.com:1904/stats-xml?virtual-server=0&server-pool=0
```

The following list shows the hierarchy of elements in the statistics XML report. Note that when you opt to suppress an element in the report, the child elements of that element are also suppressed.

```
stats
 server
 process
 connection-queue
 thread-pool
 dns
 keepalive
 thread
 request-bucket
 profile-bucket
 compression
 decompression
 origin-server-pool
 origin-server
 websocket
 service-queue
 virtual-server
 request-bucket
 websocket
 webapp-firewall
 profile-bucket
 route
 request-bucket
 cpu-info
 tcp-proxy
 cache
 failover
 partition
```

request-bucket  
ssl-session-cache

## Monitoring Using SNMP

Simple Network Management Protocol (SNMP) is a standard that enables management of devices in a network from a network management application running on a remote system. The network management application might, for example, show which servers in the network are running or stopped at any point in time, and the number and type of error messages received.

Oracle Traffic Director instances are monitored using SNMP, by default. To be able to do this, you should do the following:

- Configure the instances to support monitoring through SNMP.
- Configure the SNMP subagent on the nodes.
- Start the SNMP subagent on the nodes.

This section contains the following topics:

- [Configuring Oracle Traffic Director Instances for SNMP Support](#)
- [Configuring the SNMP Subagent](#)
- [Configuring SNMP v3 User](#)
- [Starting and Stopping the SNMP Subagent](#)
- [Viewing Statistics Using snmpwalk](#)

## Configuring Oracle Traffic Director Instances for SNMP Support

When you create a configuration, support for monitoring the instances through SNMP is enabled by default. You can disable, enable, and configure support for SNMP monitoring by using either Fusion Middleware Control or the WLST.

### Configuring SNMP Support Using Fusion Middleware Control

To enable SNMP support for a configuration by using the Fusion Middleware Control, do the following:

1. Log in to Fusion Middleware Control, as described in [Graphical User Interface-Fusion Middleware Control](#).
2. Click the WebLogic Domain button at the upper left corner of the page.
3. Select Administration > OTD Configurations.

A list of the available configurations is displayed.

4. Select the configuration for which you want to enable SNMP support.
5. Click the Traffic Director Configuration In the Common Tasks pane.
6. Select Advanced Configuration > Settings.

The Settings page is displayed, scroll down It shows a SNMP settings.

7. In the **SNMP** section of the page, select the **SNMP** check box Enabled. The other parameters in the section are optional.

On-screen help and prompts are provided for all of the parameters.

When you change the value in a field or tab out of a text field that you changed, the **Apply** button near the upper right corner of the page is enabled.

At any time, you can discard the changes by clicking the **Revert** button.

8. After making the required changes, click **Apply**.
  - A message, confirming that the updated configuration was saved, is displayed in the Console Messages pane.

### Configuring SNMP Support Using WLST

To enable SNMP using the WLST commands, do the following:

- To view the current SNMP settings for a configuration, run the `otd_getSnmpProperties` command, as shown in the following example:

```
props = {}
props['configuration'] = 'foo'
otd_getSnmpProperties(props)
```

- SNMP is enabled by default. To enable or disable SNMP support, run the `otd_setSnmpProperties` command, as shown in the following example:

```
props = {}
props['enabled'] = 'true'
props['organization'] = 'bar'
otd_setSnmpProperties(props)
```

## Configuring the SNMP Subagent

When you create an Oracle Traffic Director node, an SNMP *subagent* is created automatically. The SNMP subagent collects information about the instances running on the node.

The SNMP subagent's configuration settings, including the frequency at which the subagent updates statistics, the duration after which cached statistics are timed out, and the port through which the subagent process communicates, are stored in the following file:

- SNMP configuration file can be found at: `<domain-home>/config/fmwconfig/components/OTD/snmp/config/snmpagt.conf`
- SNMP persistent file can be found at: `<domain-home>/config/fmwconfig/components/OTD/snmp/store/snmpagt.conf`

You can configure the SNMP subagent's settings by editing the `snmpagt.conf` file. [SNMP Subagent Configuration Parameters](#) lists the key SNMP subagent parameters.

**Table 16-2** SNMP Subagent Configuration Parameters

| Parameter in <code>snmpagt.conf</code> | Description                                        | Default Value |
|----------------------------------------|----------------------------------------------------|---------------|
| <code>agentAddress</code>              | Ports at which the SNMP subagent receives requests | 11161         |
| <code>statInterval</code>              | Statistics update frequency (seconds)              | 5             |
| <code>cacheTimeOut</code>              | Cache timeout period (seconds)                     | 5             |

The syntax for entries in `snmpagt.conf` should be as described in the documentation for `snmpd.conf` at: <http://www.net-snmp.org/docs/man/snmpd.conf.html>.

After configuring the SNMP subagent on a node, you should start it. The subagent then begins collecting statistics about the Oracle Traffic Director instances on the node.

## SNMP v3 User configuration

On a system with fresh Oracle Traffic Director installation, SNMP v3 users and access control are not configured by default. A user must be added and access control configured before the server status can be queried via SNMP.

### Adding an SNMP v3 user

1. Stop the SNMP agent and add an SNMP v3 user by modifying the SNMP configuration files manually. Optionally enable encryption of all requests and responses.

- **Persistent file:** `<domain-home>/config/fmwconfig/components/OTD/snmp/store/snmpagt.conf`

A user can be added by adding an entry in the following format:

```
createUser <user-name> SHA <auth-password> [AES <crypt-password>]
```

SHA is the encryption for authentication password

DES and AES are the encryption protocols

privpassphrase is the password to encrypt the snmp request and response

- **Examples**

- Create user 'user1' and specify password used for authentication

```
createUser user1 SHA user1pwd
```

- Create user 'user2' and encrypt all requests and responses.

```
createUser user2 SHA user2pwd AES
```

- Create user 'user3' and encrypt all requests and responses with separate crypt password

```
createUser user3 SHA user3pwd AES cryptpwd
```

- **Configuration file:** `<domain-home>/config/fmwconfig/components/OTD/snmp/config/snmpagt.conf`

Access permissions for a SNMP user can be specified by adding an entry in the following format:

```
rouser username [noauth|auth|priv]
```

rouser is for read-only access. Authentication and encryption can be specified as:

noauth: To allow unauthenticated requests

auth: To enforce authentication of username with authpassphrase

priv: To enforce use of encryption

2. After adding the SNMP v3 user, the SNMP agent should be started and stopped once. When the SNMP agent is stopped, user authentication parameters provided in "createUser" line is encrypted and stored in the persistent file.
3. Ensure that the SNMP agent and one or more OTD instances are running. The SNMP agent can now be queried over SNMP v3 using <user-name> and <auth-password>.
4. Run the snmpwalk command. User can run the snmpcmd from remote host.

#### snmpcmd(snmpwalk/snmpget)

```
snmpwalk -m <oracle-home>/otd/lib/ORACLE-TRAFFICDIRECTOR-MIB.txt -v 3
-u username -l<authentication level> -a SHA -A authpassphrase -x (DES|
AES) -X privpassphrase <hostname>:11161 ORACLE-TRAFFICDIRECTOR-
MIB::originServer
```

#### snmpwalk example

```
snmpwalk -m <oracle-home>/otd/lib/ORACLE-TRAFFICDIRECTOR-MIB.txt -v 3
-u <user-name> -l priv -a SHA -A <auth-password> -x AES -X <crypt-
password> <hostname>:11161 ORACLE-TRAFFICDIRECTOR-
MIB::originServerTable
```

#### Deleting an SNMP v3 user

The user can be deleted by removing the relevant user entries from both the configuration file and persistent file and restarting the SNMP agent.

#### Integrating SNMP with Master Agent

Oracle Traffic Director SNMP can be integrated with the SNMP master agent running on the system on Linux and Solaris using the `agentx` protocol. This requires the SNMP agent to be started in `agentx` mode. When running in this mode, the Oracle Traffic Director SNMP agent cannot communicate via the SNMP protocol. All requests should be made to the SNMP master agent, which in-turn retrieves the requested information from the SNMP agent via the `agentx` protocol.

The default `agentx` transport specifier for Oracle Traffic Director SNMP is specified as below. The transport specifier can be changed as mentioned in the [man page](#) of `snmpd.conf`.

#### agentx transport specifier

```
agentxsocket tcp:127.0.0.1:705
```

1. Configure operating system master agent to communicate to sub-agents via `agentx` protocol. This requires the following changes to the master agent configuration (`/etc/snmp/snmpd.conf` on Linux and `/etc/net-snmp/snmp/snmpd.conf` on Solaris). The master agent needs to be restarted after making this change.

Directives that need to be added to the master agent configuration (`snmpd.conf`)

```
master agentx
agentxsocket tcp:127.0.0.1:705
```

- Changes to the master agent configuration requires super-user privileges.
- The `agentxsocket` directive should be the same on both the OTD SNMP agent and the OS master agent.

2. Start the SNMP agent in agentx mode.

WLST command to start OTD SNMP agent

Online

```
otd_startSnmpSubAgent({'machine':<machine_name>, 'agentx':'true'})
```

Offline

```
otd_startSnmpSubAgent({'domain-home':<otd_domain_directory_path>, 'agentx':'true'})
```

3. Send an SNMP request directly to the operating system master agent.

### Configuring TRAP Notifications

The Oracle Traffic Director SNMP agent can also generate an SNMP trap notification whenever an instance is started or stopped. This notification can be sent to one or more destinations that are specified in the configuration file at <domain-home>/config/fmwconfig/components/OTD/snmp/config/snmpagt.conf.

#### Examples for specifying trap destinations in snmpagt.conf

Send an SNMPv1 trap

```
trapsink 127.0.0.1:1127 trapComm
```

Send an SNMPv2 trap

```
trap2sink 192.168.1.99:1199 trapComm
```

Send an SNMPv2 inform which the remote side should confirm

```
informsink 192.1.68.88:1188 trapComm
```

Send an SNMPv3 trap without any auth

```
trapsess -v 3 -u v3user -l noAuth 127.0.0.1:11162
```

Send a SNMPv3 inform with full SNMPv3 security

```
trapsess -v 3 -r 0 -Ci -u myuser -n "" -l authPriv -a SHA -A <auth-password> -x AES -X <crypt-password> 192.168.1.77:1177
```

For more information on these directives see the [man page](#).

## Starting and Stopping the SNMP Subagent

You can start and stop the SNMP subagent on a node by using WLST commands.

### Starting and Stopping the SNMP Subagent Using WLST

- To start the SNMP subagent on one or more nodes, run the `otd_startSnmpSubAgent` command, as shown in the following example:

```
Online
props = {}
props['machine-name'] = 'abc123.example.com'
otd_startSnmpSubAgent(props)

Offline
props = {}
```

```
props['domain-home'] = '/export/domains/otd_domain'
otd_startSnmpSubAgent(props)
```

- To stop the SNMP subagent on one or more nodes, run the `sotd_stopSnmpSubAgent` command, as shown in the following example:

```
Online
props = {}
props['machine-name'] = 'host.example.com'
otd_stopSnmpSubAgent(props)

Offline
props = {}
props['domain-home'] = '/export/domains/otd_domain'
otd_stopSnmpSubAgent(props)
```

## Viewing Statistics Using `snmpwalk`

You can view statistics collected by the SNMP subagent, by using the `snmpwalk` command-line utility that is available in the Net-SNMP suite of applications (<http://www.net-snmp.org>).

### Note:

The prerequisites for using `snmpwalk` are as follows:

- **For Linux:** Make sure the contents `snmpwalk` package `net-snmp-utils-5.3.2.2-9.0.1.el5_5.1 RPM` or higher and standard MIBS package `net-snmp-5.3.2.2-9.0.1.el5_5.1 RPM` or higher are installed.
- **For Solaris:** Make sure the package located at `system/management/snmp/net-snmp` is installed. This package contains contents `snmpwalk` and standards MIBS.

### Simplifying Commands by Setting Defaults

Before using `snmpwalk`, if required, you can set most of the `snmpwalk` options in the `snmp.conf` file, located at `~/snmp/snmp.conf`.

The advantage of setting various options in `snmp.conf` is that after setting the options, you can run the `snmpwalk` command without specifying the options that are already set in `snmp.conf`. The `snmp.conf` enables you to set the following options:

| Default Options                | Description                                                                          |
|--------------------------------|--------------------------------------------------------------------------------------|
| <code>defSecurityNam</code>    | SNMPv3 user-name                                                                     |
| <code>defAuthType</code>       | SHA (authentication method)                                                          |
| <code>defSecurityLevel</code>  | security level for the user. i.e <code>authNoPriv</code> , <code>authPriv</code> etc |
| <code>defAuthPassphrase</code> | auth-password                                                                        |
| <code>defPrivType</code>       | privacy protocol to use. AES                                                         |
| <code>defPrivPassphrase</code> | privpassphrase                                                                       |

| Default Options | Description                               |
|-----------------|-------------------------------------------|
| defVersion      | 3                                         |
| defaultport     | 11161                                     |
| mibirds         | +<path to ORACLE-TRAFFICDIRECTOR-MIB.txt> |
| mibs            | +ORACLE-TRAFFICDIRECTOR-MIB               |

### snmpwalk with defaults

```
snmpwalk <hostname> ORACLE-TRAFFICDIRECTOR-MIB::originServerTable
```

### snmpwalk output

When you run the `snmpwalk` command, the output would be as follows:

```
ORACLE-TRAFFICDIRECTOR-MIB::originServerName.1.1.0 = STRING:
http://example.com:4000
ORACLE-TRAFFICDIRECTOR-MIB::originServerType.1.1.0 = STRING: generic
(Oracle-iPlanet-Web-Server/7.0)
ORACLE-TRAFFICDIRECTOR-MIB::originServerStatus.1.1.0 = INTEGER: online(1)
ORACLE-TRAFFICDIRECTOR-MIB::originServerDiscoveredStatus.1.1.0 =
INTEGER: false(2)
ORACLE-TRAFFICDIRECTOR-MIB::originServerRampedupStatus.1.1.0 = INTEGER:
true(1)
ORACLE-TRAFFICDIRECTOR-MIB::originServerBackupStatus.1.1.0 = INTEGER:
active(1)
ORACLE-TRAFFICDIRECTOR-MIB::originServerTimeOnline.1.1.0 = Counter64:
1566 seconds
ORACLE-TRAFFICDIRECTOR-MIB::originServerCountDetectedOffline.1.1.0 =
Counter32: 0
ORACLE-TRAFFICDIRECTOR-MIB::originServerCountBytesTransmitted.1.1.0 =
Counter64: 0
ORACLE-TRAFFICDIRECTOR-MIB::originServerCountBytesReceived.1.1.0 =
Counter64: 0
ORACLE-TRAFFICDIRECTOR-MIB::originServerCountActiveConnections.1.1.0 =
Gauge32: 0
ORACLE-TRAFFICDIRECTOR-MIB::originServerCountIdleConnections.1.1.0 =
Gauge32: 2
```



ORACLE-TRAFFICDIRECTOR-MIB::originServerCountActiveStickyConnections.1.1.0  
= Gauge32: 0

ORACLE-TRAFFICDIRECTOR-MIB::originServerCountClosedConnections.1.1.0 =  
Counter64: 100

ORACLE-TRAFFICDIRECTOR-  
MIB::originServerCountClosedConnectionsByOriginServer.1.1.0  
= Counter64: 0

ORACLE-TRAFFICDIRECTOR-MIB::originServerCountConnectAttempts.1.1.0 =  
Counter64: 102

ORACLE-TRAFFICDIRECTOR-MIB::originServerCountConnectFailures.1.1.0 =  
Counter64: 0

ORACLE-TRAFFICDIRECTOR-MIB::originServerCountRequestsAborted.1.1.0 =  
Counter64: 0

ORACLE-TRAFFICDIRECTOR-MIB::originServerCountRequestsTimedout.1.1.0 =  
Counter64: 0

ORACLE-TRAFFICDIRECTOR-MIB::originServerCountRequests.1.1.0 = Counter64: 0

ORACLE-TRAFFICDIRECTOR-MIB::originServerCountHealthCheckRequests.1.1.0 =  
Counter64: 102

ORACLE-TRAFFICDIRECTOR-MIB::originServerCountStickyRequests.1.1.0 =  
Counter64: 0

ORACLE-TRAFFICDIRECTOR-MIB::originServerDynamicWeight.1.1.0 = STRING:  
1.00

ORACLE-TRAFFICDIRECTOR-MIB::originServerSecondsKeepAliveTimeout.1.1.0 =  
Counter64: 0 seconds

ORACLE-TRAFFICDIRECTOR-  
MIB::originServerMillisecondsConnectionActiveAverage.1.1.0  
= STRING: milliseconds

ORACLE-TRAFFICDIRECTOR-MIB::originServerServerPoolName.1.1.0 = STRING:  
origin-server-pool-1

ORACLE-TRAFFICDIRECTOR-MIB::originServerInstanceName.1.1.0 = STRING:  
otd\_test\_partha-Latitude-E7440

ORACLE-TRAFFICDIRECTOR-MIB::originServerRequests1MinuteAverage.1.1.0 =  
STRING: 0

ORACLE-TRAFFICDIRECTOR-MIB::originServerRequests5MinuteAverage.1.1.0 =

STRING: 0

ORACLE-TRAFFICDIRECTOR-MIB::originServerRequests15MinuteAverage.1.1.0 =

STRING: 0

Each line in the output shows the value of a metric, but because the OID is shown in numeric format, it is difficult to identify the name of the specific metric. The `snmpwalk` utility can resolve numeric OIDs to textual names by using the management information base (MIB) definitions. For Oracle Traffic Director, the MIB definitions file is available in the following directory:

```
ORACLE_HOME/lib/snmp/ORACLE-TRAFFICDIRECTOR-MIB.txt
```

For a list of the SNMP MIB object names that you can use to query for specific statistics, see [Metrics Tracked by Oracle Traffic Director](#).

For more information about `snmpwalk`, see the documentation at: <http://www.net-snmp.org/docs/man/snmpwalk.html>.

## Monitoring Using DMS

The Oracle Dynamic Monitoring Service (DMS) provides a set of Java and C APIs that measure and report performance metrics, trace performance and provide a context correlation service for Fusion Middleware and other Oracle products. Apart from the APIs, DMS provides interfaces to enable application developers, support analysts, system administrators and others to measure application-specific performance information.

The DMS metrics for OTD are available as a set of metric tables (see [DMS Metrics Tables](#)). The monitoring data is exposed to DMS via a single Component Metric MBean. When DMS requests for monitoring data for an OTD instance, a plugin (`MetricsPlugin`) is invoked on the corresponding node manager to retrieve the statistics from the specified OTD instance. The plugin communicates with the OTD instance via native OTD interfaces to retrieve the monitoring data. The monitoring data returned from the node manager is cached on the administration server for a period of 5 seconds, during which time any request from DMS for monitoring data is satisfied from the cache.

You can view the DMS Metrics using a variety of interfaces including the DMS Spy Servlet and Oracle Fusion Middleware Control. You can also view metrics using the DMS custom WLST commands. See [DMS Custom WLST Commands](#) in *WLST Command Reference for Infrastructure Components*.

### Example 16-1 Viewing DMS Metrics Using Custom DMS WLST Commands

```
View metrics for all OTD instances
displayMetricTables('OTD_*')

View origin server metrics for all instances
displayMetricTables('OTD_OriginServer')

Get list of metric tables for a specific instance
displayMetricTableNames(servers='/OTD/otd_test_myserver.example.com')

View all metrics for a specific instance
displayMetricTables(servers='/OTD/otd_test_myserver.example.com')
```

```
View instance metrics for a specific instance
displayMetricTables('OTD_Instance', servers='/OTD/otd_test_myserver.example.com')

Dump all metrics for a specific instance
dumpMetrics(servers='/OTD/otd_test_myserver.example.com')
```

## Sample XML (stats-xml) Report

This section contains a sample statistics report in XML format, which you can view by using the `otd_getStatsXml` command or through a URI. See [Viewing Statistics Using WLST](#) and [Viewing stats-xml and perfdump Reports Through a Browser](#).

Note that the values shown in this sample report might not be meaningful. The sample report is provided here merely to indicate the metrics that the report includes and to give you a general idea about the format and structure of the report.

```
<?xml version="1.0" encoding="utf-8"?>
<stats versionMajor="1" versionMinor="3" flagEnabled="1">
<server id="otd_test_otd_Machine_1" versionServer="Oracle Traffic Director
12.2.1.2.0 B160911.182037 (Linux)" timeStarted="1475563593" secondsRunning="8209"
ticksPerSecond="1000" maxProcs="1" maxThreads="256" flagProfilingEnabled="1"
load1MinuteAverage="0.030000" load5MinuteAverage="0.020000"
load15MinuteAverage="0.050000" rateBytesTransmitted="7225" rateBytesReceived="1596"
requests1MinuteAverage="0.016667" requests5MinuteAverage="0.003333"
requests15MinuteAverage="0.000000" errors1MinuteAverage="0.000000"
errors5MinuteAverage="0.000000" errors15MinuteAverage="0.000000"
responseTime1MinuteAverage="1.000000" responseTime5MinuteAverage="1.000000"
responseTime15MinuteAverage="0.000000">
 <connection-queue id="cq1"/>
 <thread-pool id="thread-pool-0" name="NativePool"/>
 <profile id="profile-0" name="all-requests" description="All requests"/>
 <profile id="profile-1" name="default-bucket" description="Default bucket"/>
 <profile id="profile-2" name="cache-bucket" description="Cached responses"/>
 <process pid="19949" mode="active" timeStarted="1475563593"
countConfigurations="3" sizeVirtual="1191404" sizeResident="15788"
fractionSystemMemoryUsage="0.0022">
 <connection-queue connectionQueueId="cq1" countTotalConnections="9"
countQueued="0" peakQueued="1" maxQueued="1536" countOverflows="0"
countTotalQueued="9" ticksTotalQueued="1" count
Queued1MinuteAverage="0.000000" countQueued5MinuteAverage="0.000000"
countQueued15MinuteAverage="0.000000"/>
 <thread-pool threadPoolId="thread-pool-0" countIdleThreads="1"
countThreads="1" maxThreads="128" countQueued="0" peakQueued="0" maxQueued="0"/>
 <dns flagCacheEnabled="1" countCacheEntries="0" maxCacheEntries="1024"
countCacheHits="0" countCacheMisses="0" flagAsyncEnabled="0"
countAsyncNameLookups="0" countAsyncAddrLookups="0" countAsyncLookupsInProgress="0"/>
 <keepalive countConnections="1" maxConnections="3072" countHits="0"
countFlushes="0" countRefusals="0" countTimeouts="0" secondsTimeout="30"/>
 <compression countRequests="0" bytesInput="0" bytesOutput="0"
compressionRatio="0.000000" pageCompressionAverage="0.000000"/>
 <decompression countRequests="0" bytesInput="0" bytesOutput="0"/>
 <thread mode="idle" type="sync" asyncState="async-none"
timeStarted="1475563593" connectionQueueId="keep-alive">
 <request-bucket countRequests="0" countBytesReceived="0"
countBytesTransmitted="0" count2xx="0" count3xx="0" count4xx="0" count5xx="0"
countOther="0" count200="0" count302="0" count304="0" count400="0"
count401="0" count403="0" count404="0" count503="0"/>
 <profile-bucket profile="profile-0" countCalls="0" countRequests="0"
ticksDispatch="0" ticksFunction="0"/>
 <profile-bucket profile="profile-1" countCalls="0" countRequests="0"
```

```
ticksDispatch="0" ticksFunction="0"/>
 <profile-bucket profile="profile-2" countCalls="0" countRequests="0"
ticksDispatch="0" ticksFunction="0"/>
 </thread><thread mode="idle" type="sync" asyncState="async-none"
timeStarted="1475563593" connectionQueueId="keep-alive">
 <request-bucket countRequests="0" countBytesReceived="0"
countBytesTransmitted="0" count2xx="0" count3xx="0" count4xx="0" count5xx="0"
countOther="0" count200="0" count302="0" count304="0" count400="0" count401="0"
count403="0" count404="0" count503="0"/>
 <profile-bucket profile="profile-0" countCalls="0" countRequests="0"
ticksDispatch="0" ticksFunction="0"/>
 <profile-bucket profile="profile-1" countCalls="0" countRequests="0"
ticksDispatch="0" ticksFunction="0"/>
 <profile-bucket profile="profile-2" countCalls="0" countRequests="0"
ticksDispatch="0" ticksFunction="0"/>
 </thread>
 <thread mode="response" type="sync" asyncState="async-none"
timeStarted="1475563593" function="stats-xml" connectionQueueId="cq1"
virtualServerId="test" timeRequestStarted="1475572257009438">
 <request-bucket method="GET" uri="/stats-xml" countRequests="5"
countBytesReceived="1435" countBytesTransmitted="22730" count2xx="4" count3xx="0"
count4xx="0" count5xx="1" countOther="0" count200="4" count302="0" count304="0"
count400="0" count401="0" count403="0" count404="0" count503="0"/>
 <profile-bucket profile="profile-0" countCalls="48" countRequests="5"
ticksDispatch="1" ticksFunction="7581"/>
 <profile-bucket profile="profile-1" countCalls="48" countRequests="5"
ticksDispatch="1" ticksFunction="7581"/>
 <profile-bucket profile="profile-2" countCalls="0" countRequests="0"
ticksDispatch="0" ticksFunction="0"/>
 </thread>
 <thread mode="idle" type="sync" asyncState="async-none" timeStarted="1475563593"
connectionQueueId="cq1">
 <request-bucket countRequests="4" countBytesReceived="784"
countBytesTransmitted="19544" count2xx="4" count3xx="0" count4xx="0" count5xx="0"
countOther="0" count200="4" count302="0" count304="0" count400="0" count401="0"
count403="0" count404="0" count503="0"/>
 <profile-bucket profile="profile-0" countCalls="36" countRequests="4"
ticksDispatch="1" ticksFunction="1"/>
 <profile-bucket profile="profile-1" countCalls="36" countRequests="4"
ticksDispatch="1" ticksFunction="1"/>
 <profile-bucket profile="profile-2" countCalls="0" countRequests="0"
ticksDispatch="0" ticksFunction="0"/>
 </thread>
</process>
<virtual-server id="test" flagEnabled="1">
 <request-bucket method="GET" uri="/.perf" countRequests="9"
countBytesReceived="2219" countBytesTransmitted="42274" count2xx="8" count3xx="0"
count4xx="0" count5xx="1" countOther="0" count200="8" count302="0" count304="0"
count400="0" count401="0" count403="0" count404="0" count503="0"/>
 <profile-bucket profile="profile-0" countCalls="84" countRequests="9"
ticksDispatch="2" ticksFunction="7582"/>
 <profile-bucket profile="profile-1" countCalls="84" countRequests="9"
ticksDispatch="2" ticksFunction="7582"/>
 <profile-bucket profile="profile-2" countCalls="0" countRequests="0"
ticksDispatch="0" ticksFunction="0"/>
 <webapp-firewall countRequestsIntercepted="0" countRequestsAllowed="0"
countRequestsDenied="0" countRequestsDropped="0" countRequestsRedirected="0"
countRequestsDenyDetected="0" countRequestsDropDetected="0"
countRequestsRedirectDetected="0"/>
 <websocket countUpgradeRequests="0" countUpgradeRequestsFailed="0"
countUpgradeRequestsRejected="0" countActiveConnections="0" countRequestsAborted="0"
```

```

countRequestsTimedout="0" countBytesReceived="0" countBytesTransmitted="0"
millisecondsConnectionActiveAverage="0"/>
 <route id="default-route" condition="default">
 <request-bucket countRequests="2" countBytesReceived="390"
countBytesTransmitted="715" count2xx="1" count3xx="0" count4xx="0" count5xx="1"
countOther="0" count200="1" count302="0" count304="0" count400="0" count401="0"
count403="0" count404="0" count503="0"/>
 </route>
</virtual-server>
<server-pool name="origin-server-pool-1" type="http" countRetries="9">
 <service-queue countQueued="0" countQueuedHighPriority="0"
countQueuedNormalPriority="0" countQueuedLowPriority="0" countQueuedTimedout="0"
countTotalQueued="0" countTotalQueuedHighPriority="0"
countTotalQueuedNormalPriority="0" countTotalQueuedLowPriority="0"
countTotalQueuedSticky="0" countTotalStickyToNonSticky="0"
millisecondsQueuedHighPriorityAverage="0"
millisecondsQueuedNormalPriorityAverage="0"
millisecondsQueuedLowPriorityAverage="0"/>
 <origin-server name="http://example.com:4000" status="online" flagDiscovered="0"
flagRampedUp="1" type="generic (Oracle-iPlanet-Web-Server/7.0)" flagBackup="0"
secondsOnline="6774" countDetectedOffline="1" countConnectAttempts="339"
countConnectFailures="120" countClosedConnections="218"
countConnectionsClosedByOriginServer="0" countActiveConnections="0"
countIdleConnections="1" countActiveStickyConnections="0"
secondsKeepAliveTimeout="0" countRequestsAborted="0" countRequestsTimedout="0"
countStickyRequests="0" countRequests="1" countHealthCheckRequests="218"
countBytesTransmitted="508" countBytesReceived="484" weightResponseTime="1.00"
requests1MinuteAverage="0.000000" requests5MinuteAverage="0.000000"
requests15MinuteAverage="0.000000">
 <websocket countUpgradeRequests="0" countUpgradeRequestsFailed="0"
countUpgradeRequestsRejected="0" countActiveConnections="0" countRequestsAborted="0"
countRequestsTimedout="0" countBytesReceived="0" countBytesTransmitted="0"
millisecondsConnectionActiveAverage="0"/>
 </origin-server>
</server-pool>
<cache flagEnabled="1" countEntries="0" sizeHeapCache="16580" countContentHits="0"
countContentMisses="0" countHits="0" countRevalidationRequests="0"
countRevalidationFailures="0"/>
<cpu-info cpu="1" percentIdle="99.283209" percentUser="0.501040"
percentKernel="0.215751"/>
<cpu-info cpu="2" percentIdle="99.234605" percentUser="0.540530"
percentKernel="0.224865"/>
</server>
</stats>

```

## Sample Plain-Text (perfdump) Report

This section contains a sample perfdump statistics report that you can view by using the `otd_getPerfDump` command or through a URI. For information about viewing the plain-text report, see [Viewing Statistics Using WLST](#) and [Viewing stats-xml and perfdump Reports Through a Browser](#).

Note that the values shown in this sample report might not be meaningful. The sample report is provided here merely to indicate the metrics that the report includes and to give you a general idea about the format of the report.

Oracle Traffic Director 12.2.1.2.0 B160911.182037 (Linux)

Server started Mon Oct 03 23:46:32 2016

Process 19949 started Mon Oct 03 23:46:32 2016

ConnectionQueue

Current/Peak/Limit Queue Length	0/1/1536
Total Connections Queued	8
Average Queue Length (1, 5, 15 minutes)	0.00, 0.00, 0.00
Average Queuing Delay	0.12 milliseconds

HTTP Listener http-listener-1

Address	0.0.0.0:8080
Acceptor Threads	1
Default Virtual Server	test

KeepAliveInfo

KeepAliveCount	0/3072
KeepAliveHits	0
KeepAliveFlushes	0
KeepAliveRefusals	0
KeepAliveTimeouts	0
KeepAliveTimeout	30 seconds

SessionCreationInfo

Active Sessions	1
Keep-Alive Sessions	0
Keep-Alive threads	2
Active Async Sessions	0
HTTP Sessions current/max	4/258
TCP Sessions current/max	2/2

Cache

Cache Enabled	yes
Object Cache Entries	0
Cache lookup (hits/misses)	0/0
Requests served from Cache	0
Revalidation (successful/total)	0/0 ( 0.00%)
Heap space used	16580

Thread Pool NativePool

Idle/Peak/Limit	1/1/128
Work Queue Length/Peak/Limit	0/0/0

DNSSCacheInfo

enabled	yes
CacheEntries	0/1024
HitRatio	0/0 ( 0.00%)

Async DNS disabled

Performance Counters

```
Total number of requests 8
Average Request processing time 0.9479
Total Request processing time 7.5830
```

default-bucket (Default bucket)

Counter Name	Average	Total	Percent
Number of Requests		8	(100.00%)
Number of Invocations		75	(100.00%)
Latency	0.0003	0.0020	( 0.03%)
Function Processing Time	0.9476	7.5810	( 99.97%)
Total Response Time	0.9479	7.5830	(100.00%)

HTTP Origin Servers

Pool-name	Host:Port	Status	ActiveConn				
IdleConn	StickyConn	Timeouts	Aborted	Sticky-Req	Total-Req	BytesTrans	BytesRecvd
origin-server-pool-1	http://example.com:4000	online	0	1			
0	0	0	0	1	508	484	

TCP Origin Servers

No TCP origin servers are configured

Sessions

Process Function	Session	Status	Async-state	Client	Age	VS	Method	URI
		Origin-Server						
19949	sync	response	-	148.87.19.37	1	test	GET	/.perf
service-dump/	-							

TCP Proxy

```
Active Connections 0
Avg Duration 0.00 seconds
Requests (timeout/aborted/total) 0/0/0
```

SSL session cache statistics (global)

```
Total SSL sessions stored since starting: 0
Total SSL sessions expired since starting: 0
Total (pre-expiry) SSL sessions scrolled out of the cache: 0
Total SSL session retrieves since starting: 0 hits, 0 misses
Total SSL session removes since starting: 0 hits, 0 misses
```

# 17

## Tuning Oracle Traffic Director for Performance

You can use statistical data about Oracle Traffic Director instances and virtual servers to identify potential performance bottlenecks. Also, you can configure changes that you can make to improve Oracle Traffic Director performance. This chapter contains the following sections:

- [General Tuning Guidelines](#)
- [Tuning the File Descriptor Limit](#)
- [Tuning the Thread Pool and Connection Queue](#)
- [Tuning HTTP Listener Settings](#)
- [Tuning Keep-Alive Settings](#)
- [Tuning HTTP Request and Response Limits](#)
- [Tuning DNS Caching Settings](#)
- [Tuning SSL/TLS-Related Settings](#)
- [Configuring Access-Log Buffer Settings](#)
- [Enabling and Configuring Content Compression](#)
- [Tuning Connections to Origin Servers](#)
- [Solaris-specific Tuning](#)

### General Tuning Guidelines

The outcome of the tuning suggestions provided in this chapter might vary depending on your specific environment. When deciding the tuning parameters that are suitable for your needs, keep the following guidelines in mind:

- **Adjust one parameter at a time**  
To the extent possible, make one adjustment at a time. Measure the performance before and after each change, and revert any change that does not result in measurable improvement.
- **Establish test cases that you can use to create a performance benchmark**  
Before changing any parameter, set up test cases, and automate them if possible, to test the effect of the changes on performance.
- **Tune gradually**  
When adjusting a quantitative parameter, make changes in small increments. This approach is most likely to help you identify the optimal setting quickly.
- **Start afresh after a hardware or software change**



At each major system change, a hardware or software upgrade, for example, verify whether the previous tuning changes still apply.

## Tuning the File Descriptor Limit

The operating system uses file descriptors to handle file-system files as well as pseudo files, such as connections and listener sockets.

When an Oracle Traffic Director instance starts, the following parameters are taken into consideration when auto-configuring values related to file descriptors:

- HTTP processing threads (<thread-pool>)
- Access log counts for all virtual servers (<access-log>)
- Listeners (<http-listener>, <tcp-listener>)
- Keep-alive connections (<keep-alive>)
- Number of origin server pools (<origin-server-pool>)
- Number of origin servers (<origin-server>)
- Origin server connections (<origin-server>/<max-connections>)
- TCP processing threads (<tcp-thread-pool>)

The key Oracle Traffic Director objects that require file descriptors are keep-alive connections, queued connections, and connections to origin servers. If you do not explicitly specify limits for these objects, then when the Oracle Traffic Director instance starts, it configures the limits—maximum keep-alive connections, connection queue size, and maximum connections for each origin server—automatically based on the total number of available file descriptors in the system.

When the file descriptor limit is set to a very high value, auto-configuration of unspecified parameters can cause Oracle Traffic Director instances to consume excessive amount of memory or can result in sub-optimal configurations. To avoid these issues, specify values for these parameters explicitly on systems that have a high file-descriptor limit.

For instance, `max-threads * 4` should ideally be less than the maximum number of file descriptors available to the process. For example, if the file descriptor limit is set to 65536, then setting `max-threads` to 20000 will cause sub-optimal tuning as 80000 ( $20000*4=80000$ ) will exhaust/reserve file descriptors for the worker threads, which does not leave much for other subsystems. Hence a high value should be set for `max-threads` only after some experimentation.

The number of allocated file descriptors cannot exceed the limit that the system can support. To find out the current system limit for file descriptors, run the following command:

```
$ cat /proc/sys/fs/file-max
2048
```

To find out how many of the available file descriptors are being currently used, run the following command:

```
$ cat /proc/sys/fs/file-nr
```

The command returns an output that resembles the following:

625 52 2048

In this example, 625 is the number of allocated file descriptors, 52 is the number of free allocated file descriptors, and 2048 is the maximum number of file descriptors that the system supports.

 **Note:**

In Solaris, system wide file descriptors in use can be found by using the following command:

```
echo `:kmastat | mdb -k | grep file_cache`
```

This command returns an output that resembles the following:

```
file_cache 56 1154 1305 73728B 659529 0
```

In this example, 1154 is the number of file descriptors in use and 1305 the number of allocated file descriptors. Note that in Solaris, there is no maximum open file descriptors setting. They are allocated on demand as long as there is free RAM available.

When the number of allocated file descriptors reaches the limit for the system, the following error message is displayed in the system console when you try to open a file:

```
Too many open files in system.
```

The following message is written to the server log:

```
[ERROR:16] [OTD-10546] Insufficient file descriptors for optimum configuration.
```

This is a serious problem, indicating that the system is unable to open any more files. To avoid this problem, consider increasing the file descriptor limit to a reasonable number.

To change the number of file descriptors in Linux, do the following as the `root` user:

1. Edit the following line in the `/etc/sysctl.conf` file:

```
fs.file-max = value
```

`value` is the new file descriptor limit that you want to set.

2. Apply the change by running the following command:

```
/sbin/sysctl -p
```

 **Note:**

In Solaris, change the value of `rlim_fd_max` in the `/etc/system` file to specify the “hard” limit on file descriptors that a single process might have open. Overriding this limit requires superuser privilege. Similarly, `rlim_fd_cur` defines the “soft” limit on file descriptors that a single process can have open. A process might adjust its file descriptor limit to any value up to the “hard” limit defined by `rlim_fd_max` by using the `setrlimit()` call or by issuing the `limit` command in whatever shell it is running. You do not require superuser privilege to adjust the limit to any value less than or equal to the hard limit.

For example, to increase the hard limit, add the following command to `/etc/system` and reboot it once:

```
set rlim_fd_max = 65536
```

For more information about Solaris file descriptor settings, see [Files Open in a Single Process \(File Descriptor Limits\)](#).

As a rough rule of thumb, the thread-pool element, `max-threads * 4` should be less than the maximum number of file descriptors available to the process. That is, `max-threads` should be less than 1/5th of the maximum number of file descriptors.

For example, if the file descriptor limit is set to 65536, then setting `max-threads` to 20000 will cause sub-optimal tuning as  $20000 * 4 = 80000$  will exhaust/reserve file descriptors for the worker threads, leaving little else for other subsystems.

High values of `max-threads` should be used only after experimentation. Having tens of thousands of threads in a process may hurt performance.

## Tuning the Thread Pool and Connection Queue

This section contains the following topics:

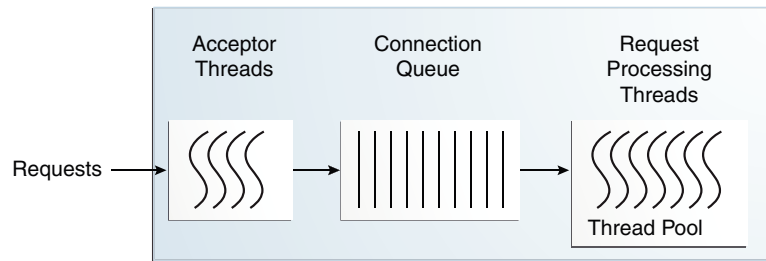
- [About Threads and Connections](#)
- [Reviewing Thread Pool Metrics for an Instance](#)
- [Reviewing Connection Queue Metrics for an Instance](#)
- [Tuning the Thread Pool and Connection Queue Settings](#)

### About Threads and Connections

When a client sends a request to an HTTP listener in an Oracle Traffic Director instance, the connection is first accepted by an *acceptor thread* that is associated with the HTTP listener. The acceptor thread puts the connection in a *connection queue* and then waits for the next client request. A *request processing thread* from a *thread pool* takes the connection from the connection queue and processes the request. Note that if the thread pool is disabled, acceptor threads themselves process every request. The connection queue and request-processing threads do not exist.

[Connection Handling in Oracle Traffic Director](#) depicts the connection handling process.

**Figure 17-1 Connection Handling in Oracle Traffic Director**



When an Oracle Traffic Director instance starts, it creates the specified number of acceptor threads for each listener and a thread pool that contains a specified, minimum number of request-processing threads.

- If the number of acceptor threads for a listener is not specified, Oracle Traffic Director creates one acceptor thread per CPU on the host.
- If the minimum size of the thread pool is not specified, Oracle Traffic Director creates one request-processing thread per processor on the host on which the instance is running.

As the request load increases, Oracle Traffic Director compares the number of requests in the connection queue with the number of request-processing threads. If the number of requests in the queue is more than the number of request-processing threads, Oracle Traffic Director creates additional threads, up to the specified maximum size for the thread pool.

The default value of the maximum number of request-processing threads will never be more than quarter of the maximum number of file descriptors available to the process. If there are 1, 2 CPUs, then the default is 256 and if there are 3, 4 CPUs, the default is 512. If there are more than 4 CPUs, the default is 1024.

The maximum number of threads is a hard limit for the number of sessions that can run simultaneously. Note that the maximum threads limit applies across all the virtual servers in the instance.

## Reviewing Thread Pool Metrics for an Instance

You can review the thread-pool information for an instance in the `SessionCreationInfo` section of the plain-text `perfdump` report, as shown in the following example.

```
SessionCreationInfo:

Active Sessions 2187
Keep-Alive Sessions 0
Total Sessions Created 4016/4016
```

- `Active Sessions` is the number of request-processing threads that are currently servicing requests.
- `Keep-Alive Sessions` shows the number of HTTP request processing threads serving keep-alive sessions.
- `Total Sessions Created`

- The first number is the number of request-processing threads created.
- The second number is the maximum threads allowed in the thread pool; that is, the sum of the maximum threads configured in the thread-pool and the number of keep alive threads.

If you observe that the total number of request-processing threads created is consistently near the maximum number of threads, consider increasing the thread limit. Otherwise, requests might have to wait longer in the connection queue; and, if the connection queue becomes full, further requests are not accepted. If the average queueing delay (see [Reviewing Connection Queue Metrics for an Instance](#)) is significantly high in proportion to the average response time, that too is an indication that the thread limit needs to be increased.

## Reviewing Connection Queue Metrics for an Instance

If the maximum size of the connection queue is not large enough, client requests might be rejected during peak load periods. You can detect this situation by examining the connection queue section in the `perfdump` plain-text report, as shown in the following example.

```
ConnectionQueue:

Current/Peak/Limit Queue Length 0/1853/160032
Total Connections Queued 11222922
Average Queue Length (1, 5, 15 minutes) 90.35, 89.64, 54.02
Average Queueing Delay 4.80 milliseconds
```

- The `Current/Peak/Limit Queue Length` line indicates the following:
  - **Current:** The number of connections currently in the queue.
  - **Peak:** The largest number of connections that have been in the queue simultaneously.  
  
If the peak queue length is close to the limit, it is an indication that the connection queue might not be large enough for the given load.
  - **Limit:** The maximum size of the connection queue, which is equal to the size of the thread-pool queue + maximum threads + the size of the keep-alive queue.
- `Total Connections Queued` is the total number of times a connection has been queued. This number includes newly-accepted connections and connections from the keep-alive system.
- `Average Queue Length` is the average number of connections in the queue during the most recent 1-minute, 5-minute, and 15-minute intervals.
- `Average Queueing Delay` is the average amount of time a connection spends in the connection queue. It represents the delay between when a request is accepted by the server and when a request-processing thread begins processing the request. If the average queueing delay is relatively high in proportion to the average response time, consider increasing the number of threads in the thread pool.

## Tuning the Thread Pool and Connection Queue Settings

You can change the thread pool and connection queue settings by using either Fusion Middleware Control or the WLST, as described in the following sections:

### Changing the Thread Pool and Connection Queue Settings Using Fusion Middleware Control

To change the thread-pool settings by using the Fusion Middleware Control, do the following:

1. Log in to Fusion Middleware Control, as described in [Graphical User Interface-Fusion Middleware Control](#).
2. Click the WebLogic Domain button at the upper left corner of the page.
3. Select Administration > OTD Configurations.  
A list of the available configurations is displayed.
4. Select the configuration for which you want to modify.
5. Click the Traffic Director Configuration In the Common Tasks pane.
6. Select Advanced Configuration > Settings.
7. Go to the **Thread Pool** section on the page.
8. Specify the parameters that you want to change.

On-screen help and prompts are provided for all of the parameters.

When you change the value in a field or tab out of a text field that you changed, the **Apply** button near the upper right corner of the page is enabled.

At any time, you can discard the changes by clicking the **Revert** button.

9. After making the required changes, click **Apply**.
  - A message, confirming that the updated configuration was saved, is displayed in the Console Messages pane.

### Changing the Thread Pool and Connection Queue Settings Using WLST

- To view the current thread-pool settings, run the `otd_getHttpThreadPoolProperties` Or `otd_getTcpThreadPoolProperties` commands, as shown in the following example:

```
props = {}
props['configuration'] = 'foo'
otd_getHttpThreadPoolProperties(props)

enabled=true
queue-size=2000
min-threads=20480
max-threads=20480
stack-size=262145
```

- To change the thread-pool settings, run the `otd_setHttpThreadPoolProperties` Or `otd_setTcpThreadPoolProperties` commands,.

For example, to change the stack size for HTTP processing threads, run the following command:

```
props = {}
props['configuration'] = 'foo'
props['stack-size'] = '8192'
otd_setHttpThreadPoolProperties(props)
```

## Tuning HTTP Listener Settings

The following are the key HTTP listener parameters that affect performance:

- **Listener address**

The listener address consists of an IP address and a port number. The host on which an Oracle Traffic Director instance is running can have multiple network interfaces and multiple IP addresses.

A listener that is configured to listen for client requests on all network interfaces on the host machine would have 0.0.0.0 as its IP address. While specifying 0.0.0.0 as the IP address for a listener is convenient, it results in one additional system call for each connection. For better performance, consider specifying an actual IP address for the listener.

- **Number of acceptor threads**

Acceptor threads receive client requests and put them in the connection queue. When an Oracle Traffic Director instance starts, it creates the specified number of acceptor threads for each listener. If the number of acceptor threads for a listener is not specified, Oracle Traffic Director creates one acceptor thread per CPU on the host

Too many idle acceptor threads place an unnecessary burden on the system, while having too few acceptor threads might result in client requests not being accepted. One acceptor thread per CPU, which is the default setting, is an acceptable trade-off in most situations.

For HTTP 1.0 workloads, which necessitate opening and closing a relatively large number of connections, the default number of acceptor threads—1 per listener—would be suboptimal. Consider increasing the number of acceptor threads.

- **Listen queue size**

As explained earlier, acceptor threads receive client requests and put them in the connection queue. If the operating system has not yet scheduled the acceptor thread, the operating system kernel maintains TCP connections on behalf of Oracle Traffic Director process. The kernel can accept connections up to the limit specified by the listen queue size.

HTTP 1.0-style workloads can have many connections established and terminated. So if clients experience connection timeouts when an Oracle Traffic Director instance is heavily loaded, you can increase the size of the HTTP listener backlog queue by setting the listen queue size to a larger value.

The plain-text `perfdump` report shows the IP address and the number of acceptor threads for each HTTP listener in the configuration, as shown in the following example:

```
ListenSocket ls1:

Address https://0.0.0.0:1904
Acceptor Threads 1
Default Virtual Server net-soa
```

You can change the HTTP listener settings by using either Fusion Middleware Control or WLST , as described in [Modifying a Listener](#).

## Tuning Keep-Alive Settings

This section contains the following topics:

- [About Keep-Alive Connections](#)
- [Reviewing Keep-Alive Connection Settings and Metrics](#)
- [Tuning Keep-Alive Settings](#)

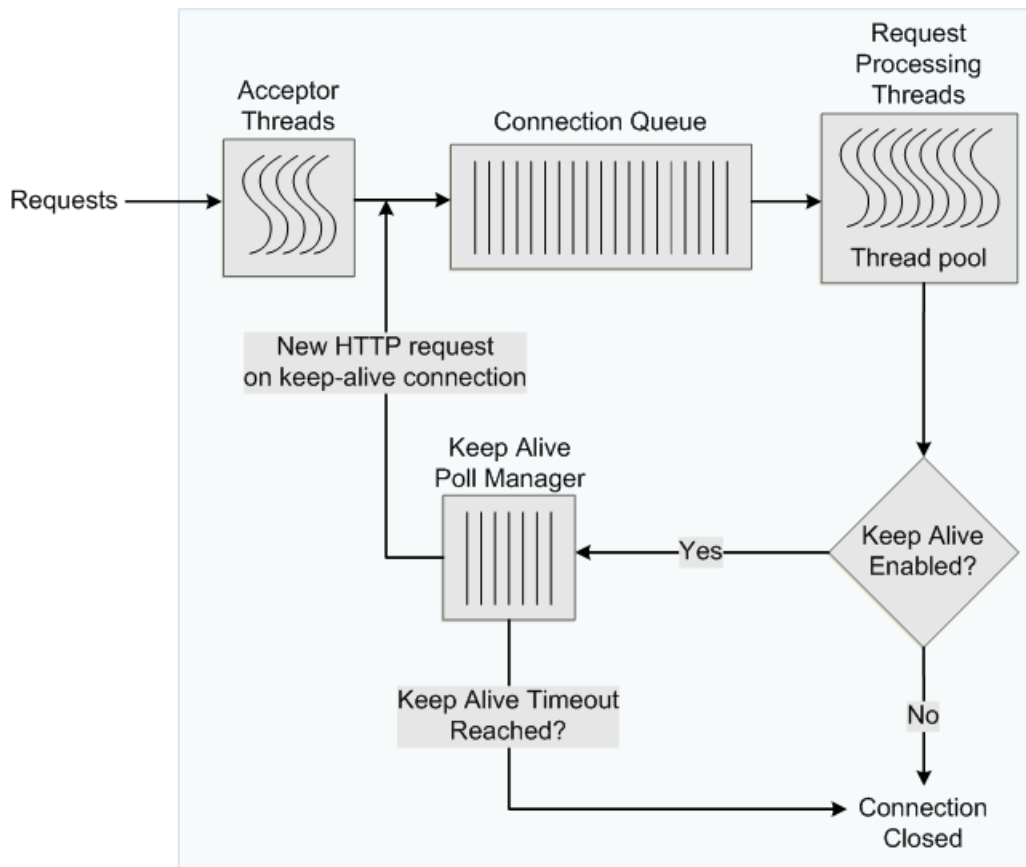
### About Keep-Alive Connections

HTTP 1.0 and HTTP 1.1 support sending multiple requests over a single HTTP connection. This capability, which was called *keep alive* in HTTP 1.0, is called *persistent connections* in HTTP 1.1 and is enabled by default in Oracle Traffic Director.

Keeping a connection active even after processing the original request reduces the time and overhead associated with creating and closing TCP connections for future similar requests. However, keep-alive connections over which few or no requests are received are an unnecessary burden on the system.

[Connection Handling in Oracle Traffic Director with Keep Alive Enabled](#) depicts the connection handling process when keep-alive is enabled.



**Figure 17-2 Connection Handling in Oracle Traffic Director with Keep Alive Enabled**

To avoid this problem, you can specify the maximum number of waiting keep-alive connections. If there are more open connections waiting for requests than the specified maximum number, the oldest connection is closed when a keep-alive request is received. In addition, you can specify the period after which inactive keep-alive connections must be closed.

## Reviewing Keep-Alive Connection Settings and Metrics

The plain-text `perfdump` report shows the current keep-alive settings and metrics, as shown in the following example:

```

KeepAliveInfo:

KeepAliveCount 26/60000
KeepAliveHits 154574634
KeepAliveFlushes 0
KeepAliveRefusals 0
KeepAliveTimeouts 5921
KeepAliveTimeout 120 seconds

```

The `KeepAliveInfo` section of the `perfdump` report shows the following:

- `KeepAliveCount`:

- The first number is the number of connections in keep-alive mode.
- The second number is the maximum number of keep-alive connections allowed.
- `KeepAliveHits` is the number of times a request was successfully received over a connection that was kept alive.

If `KeepAliveHits` is high when compared with `KeepAliveFlushes`, it indicates that the keep-alive connections are being utilized well.

If `KeepAliveHits` is low, it indicates that a large number of keep-alive connections remain idle, unnecessarily consuming system resources. To address this situation, you can do the following:

- Decrease the maximum number of keep-alive connections so that fewer connections are kept alive.

Note that the number of connections specified by the maximum connections setting is divided equally among the keep-alive threads. If the maximum connections setting is not equally divisible by the keep-alive threads setting, the server might allow slightly more than the maximum number of keep-alive connections.

- Decrease the `KeepAliveTimeout` so that keep-alive connections do not remain idle for long. Note that if the `KeepAliveTimeout` is very low, the overhead of setting up new TCP connections increases.
- `KeepAliveFlushes` is the number of times the server closed connections that the client requested to be kept alive.

To reduce keep-alive flushes, increase the keep-alive maximum connections.

#### **Caution:**

On UNIX/Linux systems, if the keep-alive maximum connections setting is too high, the server can run out of open file descriptors. Typically, 1024 is the limit for open files on UNIX/Linux; so increasing the keep-alive maximum connections above 500 is not recommended. Alternatively, you can increase the file descriptor limit, as described in [Tuning the File Descriptor Limit](#).

- `KeepAliveRefusals` is the number of times the server could not hand off a connection to a keep-alive thread, possibly because the `KeepAliveCount` exceeded the keep-alive maximum connections. If this value is high, consider increasing the maximum number of keep-alive connections.
- `KeepAliveTimeouts` is the number of times idle keep-alive connections were closed because no requests were received over them during the last `KeepAliveTimeout` period.
- `KeepAliveTimeout` is the duration, in seconds, after which idle keep-alive connections are closed.

Another parameter that is configurable and affects performance, but is not shown in the `perfdump` report is the keep-alive poll interval, which, together with `KeepAliveTimeout`, controls latency and throughput. Decreasing the poll interval and the timeout period reduces latency on lightly loaded systems. Increasing the values of these settings raises the aggregate throughput on heavily loaded

systems. However, if there is too much latency and too few clients, the aggregate throughput suffers, because the server remains idle unnecessarily. Therefore, at a given load, if there is idle CPU time, decrease the poll interval; if there is no idle CPU time, increase the poll interval.

## Tuning Keep-Alive Settings

You can tune the keep-alive settings by using either Fusion Middleware Control or the WLST.

### Changing Keep-Alive Settings Using Fusion Middleware Control

To change the keep-alive settings by using the Fusion Middleware Control, do the following:

1. Log in to Fusion Middleware Control, as described in [Graphical User Interface-Fusion Middleware Control](#).
2. Click the WebLogic Domain button at the upper left corner of the page.
3. Select Administration > OTD Configurations.

A list of the available configurations is displayed.

4. Select the configuration for which you want to modify.
5. Click the Traffic Director Configuration In the Common Tasks pane.
6. Select Advanced Configuration > HTTP.
7. Go to the **Keep Alive** section on the page.
8. Specify the parameters that you want to change.

On-screen help and prompts are provided for all of the parameters.

When you change the value in a field or tab out of a text field that you changed, the **Apply** button near the upper right corner of the page is enabled.

At any time, you can discard the changes by clicking the **Revert** button.

9. After making the required changes, click **Apply**.
  - A message, confirming that the updated configuration was saved, is displayed in the Console Messages pane.

### Changing Keep-Alive Settings Using WLST

- To view the current the keep-alive settings, run the `otd_getKeepaliveProperties` command, as shown in the following example:

```
props = {}
props['configuration'] = 'foo'
otd_getKeepaliveProperties(props)
```

```
enabled=true
threads=20
max-connections=2000
timeout=30
poll-interval=0.001
```

- To change the keep-alive settings, run the `otd_setKeepaliveProperties` command.

For example to change the maximum number of keep-alive subsystem threads, run the following command:

```
props = {}
props['configuration'] = 'foo'
props['threads'] = '128'
otd_setKeepaliveProperties(props)
```

## Tuning HTTP Request and Response Limits

To optimize the time that an Oracle Traffic Director instance spends in processing requests and responses, you can configure parameters such as the size of request and response headers, the number of allowed header fields in a request, and the time that Oracle Traffic Director waits to receive an HTTP request body and header.

You can view the change the HTTP request and response limits by using either Fusion Middleware Control or the WLST.

### Viewing and Changing HTTP Request/Response Limits Using Fusion Middleware Control

1. Log in to Fusion Middleware Control, as described in [Graphical User Interface-Fusion Middleware Control](#).
2. Click the WebLogic Domain button at the upper left corner of the page.
3. Select Administration > OTD Configurations.

A list of the available configurations is displayed.

4. Select the configuration for which you want to modify.
5. Click the Traffic Director Configuration In the Common Tasks pane.
6. Select Advanced Configuration > HTTP.
7. Go to the **HTTP** section on the page.
8. Specify the parameters that you want to change.

On-screen help and prompts are provided for all of the parameters.

When you change the value in a field or tab out of a text field that you changed, the **Apply** button near the upper right corner of the page is enabled.

At any time, you can discard the changes by clicking the **Revert** button.

9. After making the required changes, click **Apply**.
  - A message, confirming that the updated configuration was saved, is displayed in the Console Messages pane.

### Viewing and Changing HTTP Request/Response Limits Using WLST

- To view the current settings, run the `otd_getHttpProperties` command, as shown in the following example:

```
props = {}
props['configuration'] = 'foo'
otd_getHttpProperties(props)

server-header=Oracle Traffic Director/12.2.1
etag=true
request-header-buffer-size=8192
strict-request-headers=false
```

```
websocket-strict-upgrade=false
discard-misquoted-cookies=true
max-request-headers=64
body-buffer-size=1024
output-buffer-size=8192
max-unchunk-size=8192
unchunk-timeout=60
io-timeout=30
request-body-timeout=-1
request-header-timeout=30
ecid=true
favicon=true
```

- To change the request and response limits, run the `otd_setHttpProperties` command.

For example to change the un-chunk timeout, run the following command:

```
props = {}
props['configuration'] = 'foo'
props['unchunk-timeout'] = '120'
otd_setHttpProperties(props)
```

## Tuning DNS Caching Settings

DNS caching helps reduce the number of DNS lookups that Oracle Traffic Director needs to perform to resolve client host names to IP addresses. The DNS cache is enabled by default in Oracle Traffic Director and stores IP address-to-DNS name mappings. Each entry in the DNS cache represents a single IP address or DNS name lookup. The DNS cache is used only when DNS lookup is enabled and when Oracle Traffic Director performs operations that require DNS lookup, such as recording client IP addresses and host names in the access log.

For the DNS cache hit rate to be high, the cache should be large enough to store the IP address-to-DNS name mappings for the maximum number of clients that you expect to access Oracle Traffic Director concurrently. You can tune the maximum number of entries allowed in the DNS cache and the cache expiry time. Note that setting the cache size too high might result in wasted memory.

This section contains the following topics:

- [Viewing DNS Cache Settings and Metrics](#)
- [Configuring DNS Cache Settings](#)

## Viewing DNS Cache Settings and Metrics

### Viewing DNS Cache Settings

To view the current DNS cache settings for a configuration, run the `otd_getDnsCacheProperties` command, as shown in the following example:

```
props = {}
props['configuration'] = 'foo'
otd_getDnsCacheProperties(props)

enabled=true
max-age=120
max-entries=1024
```

## Viewing DNS Cache Metrics

You can view the current DNS cache utilization and hit rate in the plain-text `perfdump` report, as shown in the following example:

```
DNSCacheInfo:

enabled yes
CacheEntries 0/1024
HitRatio 0/0 (0.00%)

Async DNS disabled
```

- The first line indicates whether the DNS cache is enabled.
- `CacheEntries` shows the number of entries currently in the DNS cache and the maximum number of entries allowed.
- `HitRatio` is the number of cache hits compared to the number of DNS cache lookups.
- The last line indicates whether asynchronous DNS lookup is enabled.

You can configure Oracle Traffic Director to perform DNS lookups by using either its own asynchronous resolver or the operating system's synchronous resolver. DNS lookups performed by using the operating system's resolver are faster.

## Configuring DNS Cache Settings

You configure the DNS cache settings for a configuration by using either Fusion Middleware Control or the WLST.

### Configuring DNS Cache Settings Using Fusion Middleware Control

To configure DNS cache settings by using the Fusion Middleware Control, do the following:

1. Log in to Fusion Middleware Control, as described in [Graphical User Interface-Fusion Middleware Control](#).
2. Click the WebLogic Domain button at the upper left corner of the page.
3. Select Administration > OTD Configurations.  
A list of the available configurations is displayed.
4. Select the configuration for which you want to modify.
5. Click the Traffic Director Configuration In the Common Tasks pane.
6. Select Advanced Configuration > Settings.
7. Go to the **DNS** section on the page.
8. Specify the parameters that you want to change.

On-screen help and prompts are provided for all of the parameters.

When you change the value in a field or tab out of a text field that you changed, the **Apply** button near the upper right corner of the page is enabled.

At any time, you can discard the changes by clicking the **Revert** button.

9. After making the required changes, click **Apply**.

- A message, confirming that the updated configuration was saved, is displayed in the Console Messages pane.

### Configuring DNS Cache Settings Using WLST

To change the DNS cache settings for a configuration, run the `otd_setDnsCacheProperties` command.

For example, the following command changes the maximum amount of time to cache a DNS lookup result to 240 seconds:

```
props = {}
props['configuration'] = 'foo'
props['max-age'] = '240'
otd_setDnsCacheProperties(props)
```

## Tuning SSL/TLS-Related Settings

This section contains the following topics:

- [SSL/TLS Session Caching](#)
- [Ciphers and Certificate Keys](#)

### SSL/TLS Session Caching

During the initial SSL/TLS handshake process for an HTTPS connection, the client and server negotiate the cipher suites to be used, and the encryption/decryption and MAC keys (see [SSL/TLS Concepts](#)). This activity requires significant CPU time, depending on whether RSA or ECC private keys are used, and the size of the keys.

The initial SSL/TLS handshake results in the generation of a unique SSL/TLS session ID. If the SSL/TLS session ID is cached, then the next time that same HTTPS client opens a new socket connection, the server can reduce the time taken to establish the connection by retrieving the SSL/TLS session ID from the cache and performing an abbreviated SSL/TLS handshake, which is less CPU-intensive than the initial handshake.

SSL/TLS session caching is enabled by default in Oracle Traffic Director. When a new connection is established on an SSL/TLS-enabled listener, Oracle Traffic Director checks whether the SSL/TLS session cache contains a session ID for the client. If the session ID for the client exists in the cache and is valid, Oracle Traffic Director allows the client to reuse the session.

You can configure the maximum number of entries in the SSL/TLS session cache and the duration for which SSL/TLS session IDs should be stored in the cache.

You can configure the SSL/TLS session cache settings for a configuration by using either Fusion Middleware Control or the WLST.

### Configuring SSL/TLS Session Cache Settings Using Fusion Middleware Control

To configure SSL/TLS session cache settings by using the Fusion Middleware Control, do the following:

1. Log in to Fusion Middleware Control, as described in [Displaying Fusion Middleware Control](#).
2. Click the WebLogic Domain button at the upper left corner of the page.
3. Select Administration > OTD Configurations.  
A list of the available configurations is displayed.
4. Select the configuration for which you want to modify.
5. Click the Traffic Director Configuration In the Common Tasks pane.
6. Select Advanced Configuration > Settings.
7. Go to the **SSL/TLS Cache** section on the page.
8. Specify the parameters that you want to change.

On-screen help and prompts are provided for all of the parameters.

When you change the value in a field or tab out of a text field that you changed, the **Apply** button near the upper right corner of the page is enabled.

At any time, you can discard the changes by clicking the **Revert** button.

9. After making the required changes, click **Apply**.
  - A message, confirming that the updated configuration was saved, is displayed in the Console Messages pane.

## Configuring SSL/TLS Session Caching Settings Using WLST

- To view the current SSL/TLS caching settings for a configuration, run the `otd_getSslSessionCacheProperties` command, as shown in the following example:

```
props = {}
props['configuration'] = 'foo'
otd_getSslSessionCacheProperties(props)

enabled=true
max-entries=10000
```

- To change the SSL/TLS session caching settings, run the `otd_setSslSessionCacheProperties` command.

For example, the following command changes the maximum number of entries allowed in the SSL/TLS session cache to 20000.

```
props = {}
props['configuration'] = 'foo'
props['max-entries'] = '20000'
otd_setSslSessionCacheProperties(props)
```

## Ciphers and Certificate Keys

Strong ciphers and large private keys provide better security for SSL/TLS connections, but they affect performance.

- In SSL/TLS connections, certain ciphers—such as AES, require less computing resources for the data transfer than stronger ciphers such as 3DES. Consider this



factor when you select SSL/TLS ciphers for listeners for which Strict SNI Host Matching is enabled.

For information about configuring ciphers for listeners, see [Configuring SSL on a HTTP/TCP Listener](#).

- The initial SSL/TLS handshake process takes less time for RSA certificates with small key sizes— 2048 bits—than for certificates with large key sizes—3072 and 4096 bits.

For information about creating self-signed certificates and certificate-signing requests, see [Managing Certificates](#).

## Configuring Access-Log Buffer Settings

The access log contains information about client requests to, and responses from, the server. When the rate at which an Oracle Traffic Director instance receives client requests is very high, which is usually the case in a production environment, the frequency of writing entries to the log file on the disk increases. Writing frequently to the disk is an I/O-intensive activity that can affect the performance of the server.

To reduce the frequency at which Oracle Traffic Director writes entries to the access log on the disk, access log updates can be buffered. Access-log buffering is enabled by default in Oracle Traffic Director.

You can specify limits for the access-log buffer size, the number of access-log buffers per server, and the maximum duration for which entries should be held in the buffer. When the buffer size, the number of buffers, or the age of an entry in the buffer reaches the specified limit, Oracle Traffic Director writes the buffered data to the access log on the disk.

You can configure the access-log buffer settings by using either Fusion Middleware Control or the WLST.

### Configuring Access-Log Buffer Settings Using Fusion Middleware Control

To configure access-log buffer settings by using the Fusion Middleware Control, do the following:

1. Log in to Fusion Middleware Control, as described in [Graphical User Interface-Fusion Middleware Control](#).
2. Click the WebLogic Domain button at the upper left corner of the page.
3. Select Administration > OTD Configurations.

A list of the available configurations is displayed.

4. Select the configuration for which you want to configure access-log buffer preferences.
  5. Click the Traffic Director Configuration In the Common Tasks pane.
  6. Select Administration > Logging.
- The Log Preferences page is displayed.
7. Go to the Advanced Settings section on the page, and scroll down to the Access Log Buffer subsection.
  8. Specify the parameters that you want to change.

On-screen help and prompts are provided for all of the parameters.

When you change the value in a field or tab out of a text field that you changed, the **Apply** button near the upper right corner of the page is enabled.

At any time, you can discard the changes by clicking the **Revert** button.

9. After making the required changes, click **Apply**.
  - A message, confirming that the updated configuration was saved, is displayed in the Console Messages pane.

### Configuring Access-Log Buffer Settings Using WLST

- To view the current access-log buffer properties, run the `otd_getAccessLogBufferProperties` command, as shown in the following example:

```
props = {}
props['configuration'] = 'foo'
otd_getAccessLogBufferProperties(props)

enabled=true
buffer-size=8192
direct-io=false
max-buffers=1000
max-buffers-per-file=default
max-age=1
```

- To change the access-log buffer properties, run the `otd_setAccessLogBufferProperties` command, as shown in the following example:

```
props = {}
props['configuration'] = 'foo'
props['max-buffers'] = '2000'
otd_setAccessLogBufferProperties(props)
```

For information about viewing logs, configuring log preferences, rotating logs, and so on, see [Managing Logs](#).

## Enabling and Configuring Content Compression

Compressed objects are delivered faster to clients, with fewer round-trips, reducing the overall latency without increasing the investment in expensive hardware.

You can create one or more compression rules specific to each Oracle Traffic Director virtual server, and configure the rules to be applicable either to all requests or to only those requests that match a specified condition.

### Note:

Certain files—such as GIF, JPEG, and PNG images; and zipped files—are either already compressed or cannot be compressed any further. Requiring Oracle Traffic Director to compress such files causes additional overhead without providing any compression benefit. Therefore, when creating compression rules for a virtual server, exclude such files.

For each compression rule, you can also specify the following parameters:

- Compression level, on the scale 1–9. At level 1, the compression time is the least; at level 9, the compression ratio is the best.

At the higher compression levels, more CPU resources are consumed during the compression process, but relatively less network bandwidth is required to transmit the compressed content. On the other hand, compression at the lower levels is relatively less CPU-intensive, but more bandwidth is required to transmit the resulting content. So when choosing the compression level, consider which resource is more expensive in your environment—CPU resources or network bandwidth.

- If CPU usage is more expensive, select a lower compression level.
  - If network bandwidth is the primary constraint, select a higher compression level.
- Number of bytes (fragment size) that should be compressed at a time.
  - Whether the `Vary: Accept-Encoding` header should be included in the response.

The `Vary: Accept-Encoding` header instructs proxies situated between the client and Oracle Traffic Director that the compressed content should not be served to clients that cannot decompress the content.

### Configuring Compression Rules Using Fusion Middleware Control

To create compression rules by using the Fusion Middleware Control, do the following:

1. Log in to Fusion Middleware Control, as described in [Graphical User Interface-Fusion Middleware Control](#)
2. Click the WebLogic Domain button at the upper left corner of the page.
3. Select Administration > OTD Configurations.  
A list of the available configurations is displayed.
4. Select the configuration for which you want to create compression rules.
5. Click the Traffic Director Configuration In the Common Tasks pane.
6. Select Administration > Virtual Servers.  
The Virtual Servers page is displayed.
7. In the navigation pane, expand **Virtual Servers**, expand the name of the virtual server for which you want to create compression rules, and select **Compression**.

The Compression Rules page is displayed. It lists the compression rules that are currently defined for the virtual server, and indicates whether the rules are enabled.

#### Creating a Compression Rule

- a. Click **New Compression Rule**.

The New Compression Rule dialog box is displayed.

In the **Name** field, enter a name for the new compression rule.

#### Note:

Only small letters are available. If the value contains capital letters, it will be changed to small letters without any notifications.

**b. Click Next.**

If you wish to apply the condition, select **Edit Expression**. In the New Expression pane, select **Create** button a new page displays, Select Variable/Functions and an Operator from the respective drop-down lists and provide a value in the **Value** field.

Select the `and/or operator` from the drop-down list when configuring multiple expressions. Similarly, use the `Not` operator when you want the route to be applied only when the given expression is not true.

To enter a condition manually, click **Edit Manually** on the right top corner of the page. In the **Condition** field, specify the condition under which the rule should be applied. For information about building condition expressions, click the help button near the Condition field or see *Using Variables, Expressions, and String Interpolation* in the *Configuration File Reference for Oracle Traffic Director*.

**c. Click OK and then click Create Compression Rule.**

The caching rule that you just created is displayed on the Compression Rules page.

**Editing a Compression Rule**

To enable or disable a compression rule, or to change the settings of a rule, do the following:

**a. Click the Name of the compression rule that you want to change.**

The Edit Compression Rule dialog box is displayed.

 **Note:**

To access the condition builder to edit conditions, select **Requests satisfying the condition** and click **Edit**. The condition builder enables you to delete old expressions and add new ones.

**b. Specify the parameters that you want to change.**

On-screen help and prompts are provided for all of the parameters.

When you change the value in a field or tab out of a text field that you changed, the **OK** button near the upper right corner of the page is enabled.

At any time, you can discard the changes by clicking the **Cancel** button.

**c. After making the required changes, click OK.**

A message, confirming that the updated configuration was saved, is displayed in the Console Messages pane.

**Deleting a Compression Rule**

To delete a compression rule, click the **Delete** button. At the confirmation prompt, click **Yes**.

**Configuring Compression Rules Using WLST**

- To create a compression rule for a virtual server, run the `otd_createCompressionRule` command.

For example, the following command creates a rule named `compress-docs` for the virtual server `bar` in the configuration `foo`, to cache the requests for which the expression `$uri='^/docs'` evaluates to true.

```
props = {}
props['configuration'] = 'foo'
props['virtual-server'] = 'bar'
props['compression-rule'] = 'compress-docs'
props['condition'] = '$uri='^/docs''
otd_createCompressionRule(props)
```

Note that the value of the `condition` property should be a regular expression. For information about building condition expressions, see *Using Variables, Expressions, and String Interpolation* in the *Configuration File Reference for Oracle Traffic Director*.

- To view a list of the compression rules defined for a virtual server, run the `otd_listCompressionRules` command, as shown in the following example:

```
props = {}
props['configuration'] = 'foo'
props['virtual-server'] = 'bar'
otd_listCompressionRules(props)
```

```
compress-docs
compress-all
```

- To view the current settings of a compression rule, run the `otd_getCompressionRuleProperties` command, as shown in the following example:

```
props = {}
props['configuration'] = 'foo'
props['virtual-server'] = 'bar'
props['compression-rule'] = 'compression-rule-1'
otd_getCompressionRuleProperties(props)
```

```
name=compression-rule-1
condition="$uri = '^/doc'"
insert-vary-header=true
compression-level=6
fragment-size=8192
```

- To change a compression rule, run the `otd_setCompressionRuleProperties` command.

For example, the following command changes the compression level for the rule `compression-rule-1` to level 8.

```
props = {}
props['configuration'] = 'foo'
props['virtual-server'] = 'bar'
props['compression-rule'] = 'compression-rule-1'
props['compression-level'] = '8'
otd_setCompressionRuleProperties(props)
```

- To delete a compression rule, run the `otd_deleteCompressionRule` command, as shown in the following example.

```
props = {}
props['configuration'] = 'foo'
props['virtual-server'] = 'bar'
```

```
props['compression-rule'] = 'compression-rule-1'
otd_deleteCompressionRule(props)
```

## Tuning Connections to Origin Servers

Each Oracle Traffic Director virtual server acts as a reverse proxy through which clients outside the network can access critical data and applications hosted on multiple origin servers in the back end. This section describes the parameters that you can tune to improve the performance of Oracle Traffic Director as a reverse-proxy server.

- **Enable keep-alive:** This parameter indicates whether the Oracle Traffic Director virtual server should attempt to use persistent connections to the origin server or create a new connection for each request. It is enabled by default.
- **Keep-alive timeout:** This parameter specifies the maximum duration, in seconds, for which a persistent connection can be kept open. The default timeout duration is 29 seconds.
- **Idle timeout:** This parameter specifies the maximum duration, in seconds, for which a connection to the origin server can remain idle. The default duration is 300 seconds.
- **Always use keep-alive:** This parameter indicates whether the Oracle Traffic Director virtual server can reuse existing persistent connections to origin servers for all types of requests. If this parameter is not enabled (default), the Oracle Traffic Director virtual server attempts to use persistent connections to the origin server only for the GET, HEAD, and OPTIONS request methods.
- **Proxy buffer size:** This parameter specifies the size of the buffer in which Oracle Traffic Director stores data received from the origin server, before sending the data to the client. Larger the buffer, lower is the number of `write` system calls. The default size of the proxy buffer is 16 kilobytes.

The reverse-proxy settings for connections between an Oracle Traffic Director virtual server and an origin server pool are defined in routes. To change the reverse-proxy settings, you should edit the routes by using either Fusion Middleware Control or the WLST.

 **Note:**

In the current release, you cannot configure the proxy buffer size by using Fusion Middleware Control or WLST.

To configure the proxy buffer size for a route, do the following:

1. Add the `proxy-buffer-size` parameter to the `http-client-config` server application function (SAF) in the `vs_name-obj.conf` configuration file of the virtual server that contains the route that you want to edit.

The `vs_name-obj.conf` file is located in the following directory:

```
INSTANCE_HOME/net-config_name/config
```

The following is an example of a route (`route1`) for which the `proxy-buffer-size` and other reverse-proxy parameters have been configured.

```
<Object name="route1">
ObjectType fn="http-client-config" keep-alive-timeout="31" always-use-
keep-alive="true" keep-alive="false" timeout="360" proxy-buffer-
size="32768"
Route fn="set-origin-server" origin-server-pool="origin-server-pool-1"
</Object>
```

2. Save and close the `vs_name-obj.conf` file.
3. Run the `pullComponentChanges` command to update the configuration store on the administration server and to give effect to this change in all the instances of the configuration.

```
pullComponentChanges('otd_example.com')
```

`otd_example.com` is the name of the node on which you configured the proxy buffer size.

For more information about the `http-client-config` server application function (SAF), see the *Configuration File Reference for Oracle Traffic Director*.

## Editing Routes Using Fusion Middleware Control

To edit routes by using the Fusion Middleware Control, do the following:

1. Log in to Fusion Middleware Control, as described in [Displaying Fusion Middleware Control](#).
2. Click the WebLogic Domain button at the upper left corner of the page.
3. Select Administration > OTD Configurations.

A list of the available configurations is displayed.

4. Select the configuration for which you want to edit routes.
5. Click the Traffic Director Configuration In the Common Tasks pane.
6. Select Administration > Virtual Servers.

The Virtual Servers page is displayed.

7. In the navigation pane, expand **Virtual Servers**, expand the name of the virtual server for which you want to edit routes, and select **Routes**.

The Routes page is displayed. It lists the routes that are currently defined for the virtual server.

8. Click the **Name** of the route that you want to edit.

The Route Settings page is displayed.

9. Specify the reverse-proxy parameters in the following fields in the **Advanced Settings: Client Configuration for Connections with Origin Servers** section on the Route Settings page:

- Keep Alive
- Keep Alive Timeout
- Always Use Keep Alive
- Idle Timeout

On-screen help and prompts are provided for all of the parameters.

When you change the value in a field or tab out of a text field that you changed, the **Ok** button near the upper right corner of the page is enabled.

At any time, you can discard the changes by clicking the **Cancel** button.

10. After making the required changes, click **Ok**.

- A message, confirming that the updated configuration was saved, is displayed in the Console Messages pane.

### Configuring Routes Using WLST

To change the properties of a route, run the `otd_setRouteProperties` command. The following are the names of the reverse-proxy parameters described earlier:

```
keep-alive-timeout
always-use-keep-alive
use-keep-alive
timeout
```

For example, the following command changes the keep-alive timeout duration for the route `route1` in the virtual server `bar` of the configuration `foo` to 30 seconds.

```
props = {}
props['configuration'] = 'foo'
props['virtual-server'] = 'bar'
props['route'] = 'route-1'
props['keep-alive-timeout'] = '30'
otd_setRouteProperties(props)
```

For more information about the WLST commands mentioned in this section, see the *WebLogic Scripting Tool Command Reference for Oracle Traffic Director*.

## Solaris-specific Tuning

This section provides tuning information that is specific to Solaris. Note that these are platform-specific tuning tips and any changes that you make could affect other process on the system.



## Files Open in a Single Process (File Descriptor Limits)

Different platforms have different limits on the number of files that can be open in a single process at one time. For busy sites, increase that number. On Solaris systems, control this limit by setting `rlim_fd_max` and `rlim_fd_cur` in the `/etc/system` file. For Solaris 11, the default for `rlim_fd_max` is 65536 and the default value for `rlim_fd_cur` is 256.

After making this or any change in the `/etc/system` file, reboot Solaris for the new settings to take effect. In addition, if you upgrade to a new version of Solaris, remove any line added to `/etc/system` and add it again only after verifying that it is still valid.

An alternative way to make this change is by using the `ulimit -n <value>` command. Using this command does not require a system restart. However, this command only changes the login shell, whereas editing the `etc/system` file affects all shells.

## Failure to Connect to HTTP Server

If clients experience connection timeouts when an Oracle Traffic Director instance is heavily loaded, you can increase the size of the HTTP listener backlog queue. To increase this setting, edit the HTTP listener's `listen` queue value.

In addition to this, you must also increase the limits within the Solaris TCP/IP networking code. There are two parameters that are changed by executing the following commands:

```
ipadm set-prop -p _conn_req_max_q=4096 tcp
```

```
ipadm set-prop -p _conn_req_max_q0=4096 tcp
```

These two settings increase the maximum number of two Solaris listen queues that can fill up with waiting connections. The setting `_conn_req_max_q` increases the number of completed connections waiting to return from an `accept()` call. The setting `_conn_req_max_q0` increases the maximum number of connections with the handshake incomplete. The default values for `_conn_req_max_q` and `_conn_req_max_q0` are 128 and 1024, respectively.

You can monitor the effect of these changes by using the `netstat -s` command and looking at the `tcpListenDrop`, `tcpListenDropQ0`, and `tcpHalfOpenDrop` values. Review them before adjusting these values. If the counters are not zero, adjust the value to 2048 initially, and continue monitoring the `netstat` output.

Do not accept more connections than Oracle Traffic Director is able to process. The value of 2048 for the parameters `tcpListenDrop`, `tcpListenDropQ0`, and `tcpHalfOpenDrop` typically reduces connection request failures, and improvement has been seen with values as high as 4096.

The HTTP listener's `listen` queue setting and the related Solaris `_conn_req_max_q` and `_conn_req_max_q0` settings are meant to match the throughput of Oracle Traffic Director. These queues act as a buffer to manage the irregular rate of connections coming from web users. These queues allow Solaris to accept the connections and hold them until they are processed by Oracle Traffic Director.

## Tuning TCP Buffering

TCP buffering can be tuned by using the `send_buf` and `recv_buf` parameters. For more information about these parameters, see [Table 17-1](#).

## Reduce File System Maintenance

UNIX file system (UFS) volumes maintain the time that each file was accessed. If the file access time updates are not important in your environment, you can turn them off by adding the `noatime` parameter to the data volume's mount point in `/etc/vfstab`. For example:

```
/dev/dsk/c0t5d0s6 /dev/rdisk/c0t5d0s6 /data0 ufs 1 yes noatime
```

### Note:

The `noatime` parameter does not turn off the access time updates when the file is modified, but only when the file is accessed.

For ZFS, you can use the `zfs set` command to modify any settable dataset property. The following example sets the `atime` property to `off` for `tank/home`.

```
zfs set atime=off tank/home
```

## Long Service Times on Busy Volumes or Disks

An Oracle Traffic Director instance's responsiveness depends greatly on the performance of the disk subsystem. The `iostat` utility can be used to monitor how busy the disks are and how rapidly they complete I/O requests (the `%b` and `svct` columns, respectively). Service times are not important for disks that are less than 30% busy. However, for busier disks, service times should not exceed about 20 milliseconds. If busy disks have slower service times, improving disk performance can help performance substantially. If some disks are busy while others are lightly loaded, balance the load by moving some files from the busy disks to the idle disks.

## Short-Term System Monitoring

Solaris offers several tools for keeping track of system behavior. Although you can capture their output in files for later analysis, the tools listed below are primarily meant for monitoring system behavior in real time:

- The `iostat -x 60` command reports disk performance statistics at 60-second intervals.

To see how busy each disk is, take a look at the `%b` column. For any disk that is busy more than 20% of the time, pay attention to the service time as reported in the `svct` column. Other columns provide information about I/O operation rates, amount of data transferred, and so on.

- The `vmstat 60` command summarizes virtual memory activity and some CPU statistics at 60-second intervals.

Take a look at the `sr` column to keep track of the page scan rate and take action if it is too high. In addition, monitor the `us`, `sy`, and `id` columns to see how heavily the CPUs are being used. Note that you need to keep plenty of CPU power in reserve to handle sudden bursts of activity. Also keep track of the `r` column to see how many threads are competing for CPU time. If this remains higher than about four times the number of CPUs, reduce the server's concurrency.

- The `mpstat 60` command provides detailed view of the CPU statistics, while the `dlstat show-link -i 60` command summarizes network activity.

## Long-Term System Monitoring

While it is important to monitor system performance with the tools mentioned above, collecting longer-term performance histories is equally important, as it can help you detect trends. For example, a baseline record of a system will help you find out what has changed if the system starts behaving poorly. Enable the system activity reporting package by doing the following:

- Run the following command:

```
svcadm enable system/sar
```

- Run the command `crontab -e sys` and remove the `#` comment characters from the lines with the `sa1` and `sa2` commands. You can adjust how often the commands run and the time depending on your site's activity profile. For an explanation of the format of this file see the `crontab` man page.

This command causes the system to store performance data in files in the `/var/adm/sa` directory, where they are retained for one month by default. You can then use the `sar` command to examine the statistics for time periods of interest.

## Tuning for Performance Benchmarking

The following table shows the operating system tuning for Solaris used when benchmarking for performance and scalability. These values are an example of how you can tune your system to achieve the desired result.

**Table 17-1 Tuning Solaris for Performance Benchmarking**

Parameter	Scope	Default Value	Tuned Value	Comments
<code>rlim_fd_cur</code>	<code>/etc/system</code>	256	65536	Soft limit
<code>rlim_fd_max</code>	<code>/etc/system</code>	65536	65536	Process open file descriptors limit; accounts for the expected load (for the associated sockets, files, and pipes if any).
<code>_time_wait_interval</code>	<code>ipadm set-prop</code>	60000	600000	Set on clients as well.
<code>_conn_req_max_q</code>	<code>ipadm set-prop</code>	128	1024	
<code>_conn_req_max_q0</code>	<code>ipadm set-prop</code>	1024	4096	

**Table 17-1 (Cont.) Tuning Solaris for Performance Benchmarking**

Parameter	Scope	Default Value	Tuned Value	Comments
_ip_abort_interval	ipadm set-prop	300000	600000	
_keepalive_interval	ipadm set-prop	7200000	9000000	For high traffic web sites, lower this value.
_rexmit_interval_initial	ipadm set-prop	1000	3000	If re-transmission is greater than 30-40%, increase this value.
_rexmit_interval_max	ipadm set-prop	60000	100000	
_rexmit_interval_min	ipadm set-prop	200	3000	
smallest_open_port	ipadm set-prop	32768	65535	Set on clients as well.
send_buf	ipadm set-prop	49152	128000	To increase the transmit buffer.
recv_buf	ipadm set-prop	128000	1048576	To increase the receive buffer.

# Diagnosing and Troubleshooting Problems

Learn about the methods and information sources you can use for diagnosing and solving problems that you might encounter while using Oracle Traffic Director. This chapter contains the following sections:

- [Roadmap for Troubleshooting Oracle Traffic Director](#)
- [Solutions to Common Errors](#)
- [Frequently Asked Questions](#)
- [Contacting Oracle for Support](#)

## Roadmap for Troubleshooting Oracle Traffic Director

This section provides the sequence of tasks you can perform to diagnose and solve problems with Oracle Traffic Director.

1. Verify whether the system configuration is correct.

For information about the supported platforms and operating systems, see the Oracle Fusion Middleware Supported System Configurations at:

<http://www.oracle.com/technetwork/middleware/ias/downloads/fusion-certification-100350.html>

2. Look for a solution to the problem in [Solutions to Common Errors](#).
3. Check whether the information in [Frequently Asked Questions](#) helps you understand or solve the problem.
4. Try to diagnose the problem.

- a. Review the messages logged in the server log. Look for messages of type `WARNING`, `ERROR`, and `INCIDENT_ERROR`.

For messages of type `WARNING` and `ERROR`, try to solve the problem by following the directions, if any, in the error message.

An `INCIDENT_ERROR` message indicates a serious problem caused by unknown reasons. You should contact Oracle for support.

- b. Increase the verbosity of the server log, and try to reproduce the problem.

Oracle Traffic Director supports several log levels for the server log, as described in [Server Log](#). The default log level is `NOTIFICATION:1`. The least verbose log level is `INCIDENT_ERROR`, at which only serious error messages are logged. At the `TRACE:1`, `TRACE:16`, or `TRACE:32` levels, the logs are increasingly verbose, but provide more detailed information, which can be useful for diagnosing problems.

Increase the log verbosity and then try to reproduce the problem. When the problem occurs again, review the messages logs for pointers to the cause of the problem.

For information about changing the server log level, see [Configuring Log Preferences](#).

5. Contact Oracle for support, as described in [Contacting Oracle for Support](#).

## Troubleshooting High Availability Configuration Issues

This section provides information about the tasks you can perform to diagnose and solve problems with an Oracle Traffic Director high availability configuration.

- The Oracle Traffic Director configuration must be deployed on two nodes. See [Managing Failover Groups](#).
- The router ID for each failover group has to be unique.
- Make sure that KeepAlived is installed. In most cases KeepAlived software is installed by default on both the Exalogic compute nodes (or VMs) where Oracle Traffic Director instances are running. To check if KeepAlived is installed, run the following command:

```
rpm -qa | grep keepalived
```

If KeepAlived is correctly installed, an output similar to the following is displayed:

```
keepalived-1.2.2-1.el5
```

Note that if KeepAlived is not installed, the RPM can be found in the software repository.

- For KeepAlived specific information, check the logs in the `/var/log/messages`.
- Make sure to provide the correct VIP address and the appropriate subnet mask (netmask) bit-size for successfully completing the high availability configuration. In addition, ensure that both the nodes that host the instances must be in the same subnet. See [Creating Failover Groups](#).

## Solutions to Common Errors

This section provides solutions to the following problems:

- [Startup failure: could not bind to port](#)
- [Unable to start server with HTTP listener port 80](#)
- [Oracle Traffic Director consumes excessive memory at startup](#)
- [Operating system error: Too many open files in system](#)
- [Oracle Traffic Director does not maintain session stickiness](#)

### Startup failure: could not bind to port

This error occurs when one or more HTTP listeners in the configuration are assigned to a TCP port number that is already in use by another process.

```
[ERROR:32] startup failure: could not bind to port port (Address already in use)
[ERROR:32] [OTD-10380] http-listener-1: http://host:port: Error creating socket
(Address already in use)
[ERROR:32] [OTD-10376] 1 listen sockets could not be created
[ERROR:32] server initialization failed
```

You can find out the process that is listening on a given port by running the following command:

```
> netstat -npl | grep :port | grep LISTEN
```

If the configured HTTP listener port is being used by another process, then either free the port or change it as described in [Modifying a Listener](#).

## Unable to start server with HTTP listener port 80

This error occurs if you configure an HTTP listener port up to 1024 (say 80) and attempt to start the Oracle Traffic Director instance as a non-root user.

The following messages are written to the server log:

```
[ERROR:32] [OTD-10376] 1 listen sockets could not be created
[ERROR:32] [OTD-10380] http-listener-1: http://soa.example.com:80:
 Error creating socket (No access rights)
```

Port numbers up to 1024 are assigned by the Internet Assigned Numbers Authority (IANA) to various services. These port numbers are accessible only by the root user.

To solve this problem, you can do one of the following:

- Configure the Oracle Traffic Director listener with a port number higher than 1024 (say, 8080), and create an IP packet-filtering rule to internally redirect requests received at port 80 to the configured Oracle Traffic Director port, as shown in the following examples:

```
/sbin/iptables -t nat -A PREROUTING -p tcp -m tcp --dport 80 -j REDIRECT --to-ports 8080
/sbin/iptables -t nat -A PREROUTING -p udp -m udp --dport 80 -j REDIRECT --to-ports 8080
```

Make sure that the `iptables` service is started by default when the server restarts by running the `chkconfig` command, as shown in the following example:

```
chkconfig --level 35 iptables on
```

- If `xinetd` is installed in the system, create a file (named `otd`, for example) in the `/etc/xinetd.d/` directory with the following entry:

```
service otd
{
 type = UNLISTED
 disable = no
 socket_type = stream
 protocol = tcp
 user = root
 wait = no
 port = 80
 redirect = 127.0.0.1 8080
}
```

This entry redirects all incoming TCP traffic received on port 80 to port 8080 on the local machine.

See the Linux `xinetd` documentation.

## Oracle Traffic Director consumes excessive memory at startup

When you start an Oracle Traffic Director instance, the values for certain parameters—maximum number of keep-alive connections, size of the connection queue, and maximum number of connections to origin servers—are assigned automatically based on the system's file descriptor limit.

If the file descriptor limit is very high, the auto-assigned values for undefined parameters can be needlessly high, causing Oracle Traffic Director to consume an excessive amount of memory. To avoid this problem, explicitly configure the maximum number of keep-alive connections ([Tuning Keep-Alive Settings](#)), the size of the connection queue ([Tuning the Thread Pool and Connection Queue Settings](#)), and the maximum number of connections to individual origin servers ([Modifying an Origin Server](#)).

## Operating system error: Too many open files in system

This operating system error occurs in Linux when the number of allocated file descriptors reaches the limit for the system.

The following message is written to the server log:

```
[ERROR:16] [OTD-10546] Insufficient file descriptors for optimum configuration.
```

To avoid this error, increase the file descriptor limit on Linux from the default of 1024 to a reasonable number. See [Tuning the File Descriptor Limit](#).

## Oracle Traffic Director does not maintain session stickiness

Oracle Traffic Director can maintain session stickiness as follows:

### Cookie Based Session Persistence

This is a common scenario where clients accept cookies from web or application servers. In this scenario, Oracle Traffic Director, while load balancing HTTP traffic, ensures session persistence using its own cookie. This ensures that sticky requests, requests containing HTTP Session cookie, are routed to the same back-end application server where this session cookie originated.

### URI Based Session Persistence

This is not a very common scenario. In this case, cookies are disabled on clients and back-end web or application servers maintain session persistence by appending HTTP session information to the URI.

In this scenario, Oracle Traffic Director can honor session persistence if the back-end application server appends Oracle Traffic Director's `JRoute` cookie to the URI. Origin servers like WebLogic Server 10.3.6.2 and higher, 12.1 and higher, and GlassFish 2.0 and higher have the ability to append this `JRoute` cookie to the URI. Hence, Oracle Traffic Director is able to maintain URI based session persistence only with these origin servers.



## Frequently Asked Questions

This section contains the following subsections:

- [What is a "configuration"?](#)
- [How do I access Fusion Middleware Control?](#)
- [Why do I see a certificate warning when I access Fusion Middleware Control for the first time?](#)
- [Can I manually edit configuration files?](#)
- [In Fusion Middleware Control, what is the difference between saving a configuration and deploying it?](#)
- [Why is the "Deployment Pending" message displayed in Fusion Middleware Control?](#)
- [Why is the "Instance Configuration Deployed" message is displayed in Fusion Middleware Control?](#)
- [Why does Fusion Middleware Control session end abruptly?](#)
- [How do I access the WLST?](#)
- [Why is a certificate warning message displayed when I tried to access the WLST for the first time?](#)
- [How do I find out the short names for the options of a WLST command?](#)
- [Why am I unable to select TCP as the health-check protocol when dynamic discovery is enabled?](#)
- [After I changed the origin servers in a pool to Oracle WebLogic Servers, they are not discovered automatically, though dynamic discovery is enabled. Why?](#)
- [How do I view the request and response headers sent and received by Oracle Traffic Director?](#)
- [How do I enable SSL/TLS for an Oracle Traffic Director instance?](#)
- [How do I find out which SSL/TLS cipher suites are supported and enabled?](#)
- [How do I view a list of installed certificates?](#)
- [How do I issue test requests to an SSL/TLS-enabled Oracle Traffic Director instance?](#)
- [How do I analyze SSL/TLS connections?](#)
- [How do I view details of SSL/TLS communication between Oracle Traffic Director instances and Oracle WebLogic Server origin servers?](#)
- [Why are certain SSL/TLS-enabled origin servers marked offline after health checks, even though the servers are up?](#)
- [Does Oracle Traffic Director rewrite the source IP address of clients before forwarding requests to the origin servers?](#)
- [Why does Oracle Traffic Director return a 405 status code?](#)

## What is a "configuration"?

A configuration, in Oracle Traffic Director terminology, is a collection of configurable elements (metadata) that determine the run-time behavior of an Oracle Traffic Director instance.

For more information, see [Oracle Traffic Director Terminology](#).

## How do I access Fusion Middleware Control?

See [Displaying Fusion Middleware Control](#).

## Why do I see a certificate warning when I access Fusion Middleware Control for the first time?

The browser displays a warning because the administration server has a self-signed certificate. To proceed, you should choose to trust the certificate.

## Can I manually edit configuration files?

The files in the configuration store are updated automatically when you edit a configuration by using either Fusion Middleware Control or the WLST. Unless otherwise instructed in the Oracle Traffic Director documentation, DO NOT edit the files in the configuration store manually.

For the configuration changes to take effect, you should activate the configuration to the instances as described in [Activating Configuration Changes](#).

## In Fusion Middleware Control, what is the difference between saving a configuration and deploying it?

When you save a configuration, the changes you made are saved in the configuration store on the administration server. For the changes to take effect in the instances of the configuration, you must activate the configuration as described in [Activating Configuration Changes](#).

## Why is the "Deployment Pending" message displayed in Fusion Middleware Control?

The **Deployment Pending** message is displayed in Fusion Middleware Control when you change a configuration and save it on the administration server. It indicates that the changes are yet to be copied over to the instances of the configuration.

If you have finished making the required configuration changes, you can deploy the changes to all of the instances by clicking **Deploy Changes** in Fusion Middleware Control or by running the `activate WLST` command, as described in [Activating Configuration Changes](#).

## Why is the "Instance Configuration Deployed" message is displayed in Fusion Middleware Control?

The **Instance Configuration Deployed** message is displayed in Fusion Middleware Control when you manually edit the configuration files of an instance. It indicates that the configuration files of one or more instances are different from the corresponding configuration files stored in the configuration store on the administration server.

## Why does Fusion Middleware Control session end abruptly?

If an Fusion Middleware Control session remains inactive for 30 minutes, it ends automatically. You should log in again.

## How do I access the WLST?

See [Accessing WebLogic Scripting Tool](#).

## Why is a certificate warning message displayed when I tried to access the WLST for the first time?

The WLST connects to the SSL port of the administration server. The administration server has a self-signed certificate. The message that you see when you connect to the administration server for the first time is a prompt to choose whether you trust the certificate. Make sure that you are connecting to the correct server and port, and enter `y` to trust the certificate. For subsequent invocations of the WLST, the warning message is not displayed.

## How do I find out the short names for the options of a WLST command?

See help for the command, by running the command with the `--help` option.

## Why am I unable to select TCP as the health-check protocol when dynamic discovery is enabled?

When dynamic discovery is enabled, Oracle Traffic Director needs to send, at a specified interval, an HTTP request containing specific headers to determine whether the origin servers specified in the pool are Oracle WebLogic Server instances and whether the servers belong to a cluster. The response to a TCP health-check request would not provide the necessary information to determine the presence of Oracle WebLogic Server instances. So when dynamic discovery is enabled, the health-check protocol must be set to HTTP.

## After I changed the origin servers in a pool to Oracle WebLogic Servers, they are not discovered automatically, though dynamic discovery is enabled. Why?

If dynamic discovery is enabled, when the Oracle Traffic Director instance starts, it determines whether or not the configured origin server is an Oracle WebLogic Server instance.

So if you initially configured, say, an Oracle GlassFish Server instance as the origin server, then at startup, Oracle Traffic Director determines that the origin server is not an Oracle WebLogic Server instance. Subsequently, if you replace the origin server with an Oracle WebLogic Server instance, then for Oracle Traffic Director to determine afresh that the origin server is now an Oracle WebLogic Server instance, you must either restart the Oracle Traffic Director instances or reconfigure them.

If you want to change the origin servers from Oracle WebLogic Server instances to other servers, or vice versa, without restarting the instances, do the following:

1. Create a new origin-server pool with the required origin servers, and delete the old pool. For more information, see [Managing Origin-Server Pools](#).
2. Update the appropriate routes to point to the new pool, as described in [Configuring Routes for a Virtual Server](#).
3. Reconfigure the Oracle Traffic Director instances by using the `softRestart` WLST command, as described in [Updating Oracle Traffic Director Instances Without Restarting..](#)

## How do I view the request and response headers sent and received by Oracle Traffic Director?

You can enable logging of the request and response headers in the server log by modifying the appropriate route, using either Fusion Middleware Control or the WLST.

- **Using Fusion Middleware Control**
  1. Log in to Fusion Middleware Control, as described in [Graphical User Interface-Fusion Middleware Control](#).
  2. Click the WebLogic Domain button at the upper left corner of the page.
  3. Select Administration > OTD Configurations.  
A list of the available configurations is displayed.
  4. Select the configuration for which you want to configure routes.
  5. Click the Traffic Director Configuration In the Common Tasks pane.
  6. Select Administration > Virtual Servers.  
The Virtual Servers page is displayed.
  7. In the navigation pane, expand **Virtual Servers**, expand the name of the virtual server for which you want to edit routes, and select **Routes**.  
The Routes page is displayed. It lists the routes that are currently defined for the selected virtual server.

8. Click the **Name** of the route that you want to configure.

The Route Settings page is displayed.

9. Go to the **Advanced Settings** section of the Route Settings page, and scroll down to the **Client Configuration for Connections with Origin Servers** subsection.

10. Select the **Log Headers** check box.

11. Click **OK**.

- **Using WLST**

Run the `otd_setRouteProperties` command, as shown in the following example:

```
props = {}
props['configuration'] = 'foo'
props['virtual-server'] = 'bar'
props['route'] = 'route-1'
props['log-headers'] = 'true'
otd_setRouteProperties(props)
```

This command enables logging of the headers that Oracle Traffic Director sends to, and receives from, the origin servers associated with the route named `route-1` in the virtual server `bar` of the configuration `foo`.

The headers are logged in the server log as shown in the following example:

```
[2011-11-11T03:45:00.000-08:00] [net-test] [NOTIFICATION] [OTD-11008] []
 [pid: 8184] for host 10.177.243.152 trying to OPTIONS / while trying to GET
 /favicon.ico, service-http reports: request headers sent to origin
server(soa.example.com:1900) :[[
OPTIONS / HTTP/1.1
Proxy-agent: Oracle-Traffic-Director/12.2.1.0
Surrogate-capability: otd="Surrogate/1.0"
Host: dadvma0178:1900
Proxy-ping: true
X-weblogic-force-jvmid: unset
Via: 1.1 net-test
Connection: keep-aliv e
]]
[2011-11-11T03:45:00.000-08:00] [net-test] [NOTIFICATION] [OTD-11009] []
 [pid: 8184] for host 10.177.243.152 trying to OPTIONS / while trying to GET
 /favicon.ico, service-http reports: response headers received from origin
server(soa.example.com:1900) :[[
HTTP/1.1 200 OK
date: Fri, 11 Nov 2011 11:45:00 GMT
server: Apache/2.2.17 (Unix)
allow: GET,HEAD,POST,OPTIONS,TRACE
content-length: 0
keep-alive: timeout=5, max=100
connection: Keep-Alive
content-type: text/html]
```

## How do I enable SSL/TLS for an Oracle Traffic Director instance?

See [Configuring SSL/TLS Between Oracle Traffic Director and Clients](#).

## How do I find out which SSL/TLS cipher suites are supported and enabled?

See [Cipher Suites Supported by Oracle Traffic Director](#).

## How do I view a list of installed certificates?

See [Viewing a List of Certificates](#).

## How do I issue test requests to an SSL/TLS-enabled Oracle Traffic Director instance?

Run the following command:

```
$ openssl s_client -host hostname -port portnumber -quiet
```

- If you omit the `-quiet` option, information about the SSL/TLS connection—such as the server DN, certificate name, and the negotiated cipher suite—is displayed.
- For testing with a specific cipher, specify the `-cipher` option.

After the SSL/TLS connection is established, enter an HTTP request, as shown in the following example.

```
GET /
```

For more information, see the `s_client` man page.

## How do I analyze SSL/TLS connections?

Several tools are available to observe request and response data over SSL/TLS connections. One such tool is `ssltap`, which serves as a simple proxy between the client and the Oracle Traffic Director and displays information about the connections that it forwards.

Run the following command:

```
$ ssltap -l -s otd_host:otd_port
```

For example, to observe the communication between clients and the SSL/TLS-enabled Oracle Traffic Director listener `soa.example.com:1905`, run the following command:

```
$ ssltap -l -s soa.example.com:8080
```

The following messages are displayed:

```
Looking up "localhost" ...
Proxy socket ready and listening
```

By default, `ssltap` listens on port 1924. Connect to `https://localhost:1924` by using your browser.

You will see an output similar to the following:

```

Connection #1 [Tue Oct 25 04:29:46 2011]
Connected to localhost:8080
--> [
(177 bytes of 172)
SSLRecord { [Tue Oct 25 04:29:46 2011]
 type = 22 (handshake)
 version = { 3,1 }
 length = 172 (0xac)
 handshake {
 type = 1 (client_hello)
 length = 168 (0x0000a8)
 ClientHelloV3 {
 client_version = {3, 1}
 random = {...}
 session ID = {
 length = 0
 contents = {...}
 }
 }
 cipher_suites[29] = {
 (0x00ff) TLS_EMPTY_RENEGOTIATION_INFO_SCSV
 (0xc00a) TLS/ECDHE-ECDSA/AES256-CBC/SHA
 (0xc014) TLS/ECDHE-RSA/AES256-CBC/SHA
 (0x0039) TLS/DHE-RSA/AES256-CBC/SHA
 (0x0038) TLS/DHE-DSS/AES256-CBC/SHA
 (0xc00f) TLS/ECDH-RSA/AES256-CBC/SHA
 (0xc005) TLS/ECDH-ECDSA/AES256-CBC/SHA
 (0x0035) TLS/RSA/AES256-CBC/SHA
 (0xc007) TLS/ECDHE-ECDSA/RC4-128/SHA
 (0xc009) TLS/ECDHE-ECDSA/AES128-CBC/SHA
 (0xc011) TLS/ECDHE-RSA/RC4-128/SHA
 (0xc013) TLS/ECDHE-RSA/AES128-CBC/SHA
 (0x0033) TLS/DHE-RSA/AES128-CBC/SHA
 (0x0032) TLS/DHE-DSS/AES128-CBC/SHA
 (0xc00c) TLS/ECDH-RSA/RC4-128/SHA
 (0xc00e) TLS/ECDH-RSA/AES128-CBC/SHA
 (0xc002) TLS/ECDH-ECDSA/RC4-128/SHA
 (0xc004) TLS/ECDH-ECDSA/AES128-CBC/SHA
 (0x0004) SSL3/RSA/RC4-128/MD5
 (0x0005) SSL3/RSA/RC4-128/SHA
 (0x002f) TLS/RSA/AES128-CBC/SHA
 (0xc008) TLS/ECDHE-ECDSA/3DES-EDE-CBC/SHA
 (0xc012) TLS/ECDHE-RSA/3DES-EDE-CBC/SHA
 (0x0016) SSL3/DHE-RSA/3DES192EDE-CBC/SHA
 (0x0013) SSL3/DHE-DSS/DES192EDE3CBC/SHA
 (0xc00d) TLS/ECDH-RSA/3DES-EDE-CBC/SHA
 (0xc003) TLS/ECDH-ECDSA/3DES-EDE-CBC/SHA
 (0xfeff) SSL3/RSA-FIPS/3DESEDE-CBC/SHA
 (0x000a) SSL3/RSA/3DES192EDE-CBC/SHA
 }
 compression[1] = {
 (00) NULL
 }
 extensions[55] = {
 extension type server_name, length [29] = {
0: 00 1b 00 00 18 64 61 64 76 6d 61 30 31 37 38 2e |soa.
10: 75 73 2e 6f 72 61 63 6c 65 2e 63 6f 6d | example.com
 }
 extension type elliptic_curves, length [8] = {
0: 00 06 00 17 00 18 00 19 |
 }
 extension type ec_point_formats, length [2] = {

```

```

0: 01 00 | ..
 }
 extension type session_ticket, length [0]
 }
}

```

This is the SSL/TLS *client hello* sent from the browser to the Oracle Traffic Director instance. Note the list of cipher suites sent by the browser. These are the cipher suites that the browser is configured to handle, sorted in order of preference. The server selects one of the cipher suites for the handshake. If the server is not configured any of the cipher suites indicated by the client, the connection fails. In the above example, the session ID is empty, indicating that the browser does not have any cached SSL/TLS session with the specified server.

The Oracle Traffic Director instance's response would be similar to the following output:

```

<-- [
(823 bytes of 818)
SSLRecord { [Tue Oct 25 04:29:46 2011]
 type = 22 (handshake)
 version = { 3,1 }
 length = 818 (0x332)
 handshake {
 type = 2 (server_hello)
 length = 77 (0x00004d)
 ServerHello {
 server_version = {3, 1}
 random = {...}
 session ID = {
 length = 32
 contents = {...}
 }
 }
 cipher_suite = (0x0035) TLS/RSA/AES256-CBC/SHA
 compression method = (00) NULL
 extensions[5] = {
0: 00 | .
 }
 }
 }
 type = 11 (certificate)
 length = 729 (0x0002d9)
 CertificateChain {
 chainlength = 726 (0x02d6)
 Certificate {
 size = 723 (0x02d3)
 data = { saved in file 'cert.001' }
 }
 }
 type = 14 (server_hello_done)
 length = 0 (0x000000)
}
]
--> [

```



The server selected the cipher suite, TLS/RSA/AES256-CBC/SHA and a session ID, which the client will include in subsequent requests.

The server also sent its certificate chain for the browser to verify. `ssltap` saved the certificates in the file `cert.001`. You can examine the certificates with any tool that can parse X.509 certificates. For example, run the following command:

```
$ openssl x509 -in cert.001 -text -inform DER
```

 **Note:**

`ssltap` is a single threaded proxy server. So if you issue multiple requests through it, the requests will get serialized. If you need to analyze a specific problem with your application that only occurs on concurrent requests through SSL/TLS, try running multiple `ssltap` instances.

## How do I view details of SSL/TLS communication between Oracle Traffic Director instances and Oracle WebLogic Server origin servers?

Configure SSL debugging for the Oracle WebLogic Server instance by adding the `-Dssl.debug=true` system property in the start script of the server. For more information, see *SSL Debugging* in the *Oracle Fusion Middleware Securing Oracle WebLogic Server*.

Increase the verbosity of the Oracle Traffic Director server log by setting the log level to `TRACE:32`, as described in [Configuring Log Preferences](#).

## Why are certain SSL/TLS-enabled origin servers marked offline after health checks, even though the servers are up?

This error can occur for the following origin servers:

- SSL/TLS-enabled origin servers that are configured in the origin-server pool by using IP addresses instead of host names.
- Dynamically discovered, SSL/TLS-enabled Oracle WebLogic Server origin servers. Oracle Traffic Director refers to them using their IP addresses rather than the host names.

While Oracle Traffic Director refers to such origin servers by using their IP addresses, the certificates of the origin servers contain the servers' host names. So, in response to health-check requests, when the origin servers present certificates, Oracle Traffic Director attempts, unsuccessfully, to validate them. The SSL/TLS handshake fails. As a result, the health checks show such origin servers to be offline. Note that server-certificate validation is enabled by default.

If you set the server-log level to `TRACE:32`, you can view the message logged for this failure, as shown in the following example:

```
[2011-11-21T09:50:54-08:00] [net-soa] [TRACE:1] [OTD-10969] [] [pid: 22466]
 trying to OPTIONS /, service-http reports: error sending request
 (SSL_ERROR_BAD_CERT_DOMAIN: Requested domain name does not match the server's
 certificate.)
```

To solve this problem, disable validation of the origin-server certificates for the origin server, by running the `otd_setOriginServerPoolSslProperties` WLST command, as shown in the following example:

```
props = {}
props['configuration'] = 'foo'
props['origin-server-pool'] = 'origin-server-pool-1'
props['validate-server-cert'] = 'false'
otd_setOriginServerPoolSslProperties(props)
```

For more information, see `otd_setOriginServerPoolSslProperties` command in the *WebLogic Scripting Tool Command Reference for Oracle Traffic Director*.

## Does Oracle Traffic Director rewrite the source IP address of clients before forwarding requests to the origin servers?

The default behavior of Oracle Traffic Director is to rewrite the source IP address. However, Oracle Traffic Director does send the client IP address in an additional request header `Proxy-client-ip`. You can set up Oracle Traffic Director to block or forward `Proxy-client-ip` and other request headers by configuring the appropriate route as described in [Configuring Routes for a Virtual Server](#).

Note that Oracle Traffic Director cannot maintain case sensitivity of the HTTP request headers while forwarding them to origin servers.

## Why does Oracle Traffic Director return a 405 status code?

If an HTTP request does not meet the conditions specified in any of the defined routes and there is no default (=unconditional) route in the configuration, then Oracle Traffic Director returns the 405 status code. This error indicates that Oracle Traffic Director did not find any valid route for the request. This situation can occur only if the default route, which is used when the request does not meet the conditions specified in any of the other routes, is deleted manually in the `obj.conf` configuration file. To solve this issue the administrator must create a valid route.



### Note:

The default (=unconditional) route cannot be deleted through Fusion Middleware Control and the WLST, and should not be deleted manually.

## Contacting Oracle for Support

If you have a service agreement with Oracle, you can contact Oracle Support (<http://support.oracle.com>) for help with Oracle Traffic Director problems.

### Before Contacting Oracle Support

Before contacting Oracle Support, do the following:

- Try all the appropriate diagnostics and troubleshooting guidelines described in this document (*Oracle Traffic Director Administrator's Guide*).

- Check whether the problem you are facing, or a similar problem, has been discussed in the OTN Discussion Forums at <http://forums.oracle.com/>.

If the information available on the forum is not sufficient to help you solve the problem, post a question on the forum. Other Oracle Traffic Director users on the forum might respond to your question.

- To the extent possible, document the sequence of actions you performed just before the problem occurred.
- Where possible, try to restore the original state of the system, and reproduce the problem using the documented steps. This helps to determine whether the problem is reproducible or an intermittent issue.
- If the issue can be reproduced, try to narrow down the steps for reproducing the problem. Problems that can be reproduced by small test cases are typically easier to diagnose when compared with large test cases.

Narrowing down the steps for reproducing problems enables Oracle Support to provide solutions for potential problems faster.

### Information You Should Provide to Oracle Support

When you contact Oracle for support, provide the following information.

- The release number of Oracle Traffic Director.
- A brief description of the problem, including the actions you performed just before the problem occurred.
- If you need support with using the administration interfaces, the name of the command-line subcommand or the title of the administration-console screen for which you require help.
- Zip file containing the configuration files for the configuration in which you encountered the error.
- The latest server and access log files.

#### Note:

When you send files to Oracle Support, remember to provide the MD5 checksum value for each file, so that Oracle Support personnel can verify the integrity of the files before using them for troubleshooting the problem.

# A

## Metrics Tracked by Oracle Traffic Director

This appendix lists the metrics for which Oracle Traffic Director tracks and maintains statistics.

- [Instance Metrics](#)
- [Process Metrics](#)
- [Connection Queue Metrics](#)
- [Thread Pool Metrics](#)
- [DNS Cache Metrics](#)
- [Keep-Alive Metrics](#)
- [Thread Metrics](#)
- [Compression and Decompression Metrics](#)
- [Virtual Server Metrics](#)
- [CPU Metrics](#)
- [Origin Server Metrics](#)
- [Failover Instance Metrics](#)
- [Cache Metrics](#)
- [DMS Metrics Tables](#)

### Instance Metrics

This section lists the metrics that Oracle Traffic Director tracks for individual instances. For each metric, the object name in the SNMP MIB and the names of the corresponding element and attribute in the stats-xml report are provided. Metrics that are not available through SNMP or in the stats-xml report are marked NA.

**Table A-1 Instance Metrics**

Metric	Object Name in the SNMP MIB	stats-xml Element: Attribute
Number of seconds the instance has been running	instanceUptime	server: secondsRunning
Number of requests processed	instanceRequests	request-bucket: countRequests
Number of octets received	instanceInOctets	request-bucket: countBytesReceived
Number of octets transmitted	instanceOutOctets	request-bucket: countBytesTransmitted
Number of 2xx (Successful) responses issued	instanceCount2xx	request-bucket: count2xx

**Table A-1 (Cont.) Instance Metrics**

<b>Metric</b>	<b>Object Name in the SNMP MIB</b>	<b>stats-xml Element: Attribute</b>
Number of 3xx (Redirection) responses issued	instanceCount3xx	request-bucket: count3xx
Number of 4xx (Client Error) responses issued	instanceCount4xx	request-bucket: count4xx
Number of 5xx (Server Error) responses issued	instanceCount5xx	request-bucket: count5xx
Number of other (neither 2xx, 3xx, 4xx, nor 5xx) responses issued	instanceCountOther	request-bucket: countOther
Number of 200 (OK) responses issued	instanceCount200	request-bucket: count200
Number of 302 (Moved Temporarily) responses issued	instanceCount302	request-bucket: count302
Number of 304 (Not Modified) responses issued	instanceCount304	request-bucket: count304
Number of 400 (Bad Request) responses issued	instanceCount400	request-bucket: count400
Number of 401 (Unauthorized) responses issued	instanceCount401	request-bucket: count401
Number of 403 (Forbidden) responses issued	instanceCount403	request-bucket: count403
Number of 404 (Not Found) responses issued	instanceCount404	request-bucket: count404
Number of 503 (Unavailable) responses issued	instanceCount503	request-bucket: count503
Average load in the last 1 minute	instanceLoad1MinuteAverage	server: load1MinuteAverage
Average load in the last 5 minutes	instanceLoad5MinuteAverage	server: load5MinuteAverage
Average load for in the last 15 minutes	instanceLoad15MinuteAverage	server: load15MinuteAverage
Number of octets transmitted on the network per second	instanceNetworkInOctets	server: rateBytesReceived
Number of octets received on the network per second	instanceNetworkOutOctets	server: rateBytesTransmitted
Average number of requests served in the last 1 minute	instanceRequests1MinuteAverage	server: requests1MinuteAverage
Average number of requests served in the last 5 minutes	instanceRequests5MinuteAverage	server: requests5MinuteAverage
Average number of requests served in the last 15 minutes	instanceRequests15MinuteAverage	server: requests15MinuteAverage
Average number of error responses in the last 1 minute	instanceErrors1MinuteAverage	server: errors1MinuteAverage

**Table A-1 (Cont.) Instance Metrics**

<b>Metric</b>	<b>Object Name in the SNMP MIB</b>	<b>stats-xml Element: Attribute</b>
Average number of error responses in the last 5 minutes	instanceErrors5MinuteAverage	server: errors5MinuteAverage
Average number of error responses in the last 15 minutes	instanceErrors15MinuteAverage	server: errors15MinuteAverage
Average response time for the requests in the last 1 minute	instanceResponseTime1MinuteAverage	server: responseTime1MinuteAverage
Average response time for the requests in the last 5 minutes	instanceResponseTime5MinuteAverage	server: responseTime5MinuteAverage
Average response time for the requests in the last 15 minutes	instanceResponseTime15MinuteAverage	server: responseTime15MinuteAverage
Number of open connections at the time when statistics were gathered	NA	request-bucket: countOpenConnections
Name of the TCP proxy for which this element holds statistics	tcpProxyName	tcp-proxy:name
State of the TCP proxy at the time of gathering the statistics	tcpProxyEnabled	tcp-proxy:flagEnabled
The IP address and port on which this TCP proxy listens for requests	tcpProxyListeners	tcp-proxy:listeners
Number of active TCP proxy connections	tcpProxyCountActiveConnections	tcp-proxy:countActiveConnections
Total number of requests processed	tcpProxyCountRequests	tcp-proxy:countRequests
Total number of requests that were aborted	tcpProxyCountRequestsAborted	tcp-proxy:countRequestsAborted
Total number of requests that were closed because of timeout	tcpProxyCountRequestsTimedout	tcp-proxy:countRequestsTimedout
Number of bytes received from the clients	tcpProxyCountBytesReceived	tcp-proxy:countBytesReceived
Number of bytes transmitted to the clients	tcpProxyCountBytesTransmitted	tcp-proxy:countBytesTransmitted
Average duration of active connections in milliseconds	tcpProxyMillisecondsConnectionActiveAverage	tcp-proxy:millisecondsConnectionActiveAverage

## Process Metrics

This section lists the metrics that Oracle Traffic Director tracks at the process level. For each metric, the object name in the SNMP MIB and the names of the corresponding element and attribute in the stats-xml report are provided. Metrics that are not available through SNMP or in the stats-xml report are marked NA.

**Table A-2 Process Metrics**

<b>Metric</b>	<b>Object Name in the SNMP MIB</b>	<b>stats-xml Element: Attribute</b>
Number of request processing threads currently available	processThreadCount	thread-pool: countThreads
Number of request processing threads currently idle	processThreadIdle	thread-pool: countIdleThreads
Number of connections currently in keepalive queue	processKeepaliveCount	keepalive: countConnections
Maximum number of connections allowed in keepalive queue	processKeepaliveMax	keepalive: maxConnections
Number of requests that were processed on connections in the Keep Alive subsystem	NA	keepalive: countHits
Number of connections in the Keep Alive subsystem that were flushed	NA	keepalive: countFlushes
Number of times the server could not hand off the connection to a keep-alive thread.	NA	keepalive: countRefusals
Number of connections that were closed due to Keep Alive subsystem being idle beyond the specified timeout period	NA	keepalive: countTimeouts
Idle period after which the Keep Alive subsystem should time out	NA	keepalive: secondsTimeout
Process size in kbytes	processSizeVirtual	process: sizeVirtual
Process resident size in kbytes	processSizeResident	process: sizeResident
Fraction of process memory in system memory	processFractionSystemMemoryUsage	process: fractionSystemMemoryUsage
Total number of active connections for which requests are getting processed	NA	tcp-thread:countActiveConnections

## Connection Queue Metrics

This section lists the connection-queue metrics that Oracle Traffic Director tracks. For each metric, the object name in the SNMP MIB and the names of the corresponding element and attribute in the stats-xml report are provided. Metrics that are not available through SNMP or in the stats-xml report are marked NA.

**Table A-3 Connection Queue Metrics**

<b>Metric</b>	<b>Object Name in the SNMP MIB</b>	<b>stats-xml Element: Attribute</b>
Number of connections currently in connection queue	connectionQueueCount	connection-queue: countQueued
Total number of connections added to this connection queue since start up	NA	connection-queue: countTotalQueued
Average length of the queue in the last one minute	NA	connection-queue: countQueued1MinuteAverage
Average length of the queue in the last one minutes	NA	connection-queue: countQueued5MinuteAverage
Average length of the queue in the last fifteen minutes	NA	connection-queue: countQueued15MinuteAverage
Largest number of connections that have been queued simultaneously	connectionQueuePeak	connection-queue: peakQueued
Maximum number of connections allowed in connection queue	connectionQueueMax	connection-queue: maxQueued
Total number of connections that have been accepted since start up	connectionQueueTotal	connection-queue: countTotalConnections
Number of connections rejected due to connection queue overflow	connectionQueueOverflows	connection-queue: countOverflows

## Thread Pool Metrics

This section lists the metrics that Oracle Traffic Director tracks for server threads. For each metric, the object name in the SNMP MIB and the names of the corresponding element and attribute in the stats-xml report are provided.

**Table A-4 Thread Pool Metrics**

<b>Metric</b>	<b>Object Name in the SNMP MIB</b>	<b>stats-xml Element: Attribute</b>
Number of requests queued for processing by this thread pool.	threadPoolCount	thread-pool: countQueued
Largest number of requests that have been queued simultaneously	threadPoolPeak	thread-pool: peakQueued



**Table A-4 (Cont.) Thread Pool Metrics**

<b>Metric</b>	<b>Object Name in the SNMP MIB</b>	<b>stats-xml Element: Attribute</b>
Maximum number of requests allowed in the queue	threadPoolMax	thread-pool: max-threads

## DNS Cache Metrics

This section lists the DNS cache lookup metrics that Oracle Traffic Director tracks. For each metric, the element and attribute in the stats-xml report are provided.

**Table A-5 DNS Cache Metrics**

<b>Metric</b>	<b>stats-xml Element: Attribute</b>
Total number of entries in the cache	dns: countCacheEntries
Total number of times a cache lookup succeeded	dns: countCacheHits
Total number of times a cache lookup failed	dns: countCacheMisses
Number of asynchronous lookups	dns: countAsyncNameLookups
Total number of asynchronous DNS address lookups performed	dns: countAsyncAddrLookups
Number of asynchronous DNS lookups currently in progress	dns: countAsyncLookupsInProgress

## Keep-Alive Metrics

This section lists the metrics that Oracle Traffic Director tracks related to the keep-alive subsystem within the process.

For each metric, the object name in the SNMP MIB and the names of the corresponding element and attribute in the stats-xml report are provided. Metrics that are not available through SNMP or in the stats-xml report are marked NA.

**Table A-6 Keep-Alive Metrics**

<b>Metric</b>	<b>Object Name in the SNMP MIB</b>	<b>stats-xml Element: Attribute</b>
Total number of connections that were added to the keep-alive subsystem	keepaliveCountConnections	keepalive: countConnections
Maximum number of connections that can be maintained in the keep-alive subsystem	keepaliveMaxConnections	keepalive: maxConnections
Number of requests that were processed on connections in the keep-alive subsystem	keepaliveCountHits	keepalive: countHits

**Table A-6 (Cont.) Keep-Alive Metrics**

<b>Metric</b>	<b>Object Name in the SNMP MIB</b>	<b>stats-xml Element: Attribute</b>
Number of connections in the keep-alive subsystem that were flushed	keepaliveCountFlushes	keepalive: countFlushes
Number of times a connection was not able to enter the keep-alive subsystem because the max-connection limit was reached	keepaliveCountRefusals	keepalive: countRefusals
Number of connections that were closed due to idle timeout expiring	keepaliveCountTimeouts	keepalive: countTimeouts
Idle timeout value for the keep-alive subsystem	keepaliveSecondsTimeout	keepalive: secondsTimeout

## Thread Metrics

This section lists the metrics that Oracle Traffic Director tracks for a particular worker thread in the server process.

For each metric, the object name in the SNMP MIB and the names of the corresponding element and attribute in the stats-xml report are provided. Metrics that are not available through SNMP or in the stats-xml report are marked NA.

**Table A-7 Thread Metrics**

<b>Metric</b>	<b>Object Name in the SNMP MIB</b>	<b>stats-xml Element: Attribute</b>
The activity mode of the thread at the time of gathering the statistics	NA	thread: mode
The time when this thread started executing.	NA	thread: timeStarted
The SAF which thread was running at the time the statistics was gathered.	NA	thread: function
The ID of the connection queue from which this worker thread is picking up requests.	NA	thread: connectionQueueId
The virtual server for which the thread was serving request at the time the statistics was gathered.	NA	thread: virtualServerId
IP address of the client for which thread is processing the request at the time the statistics was gathered.	NA	thread: clientAddress
The time when thread started executing the current request.	NA	thread: timeRequestStarted
Current state of the thread within proxy-retrieve.	NA	thread: proxyMode

**Table A-7 (Cont.) Thread Metrics**

Metric	Object Name in the SNMP MIB	stats-xml Element: Attribute
Origin server which is processing the current request.	NA	thread: originServer

## Compression and Decompression Metrics

This section lists the metrics for response data that Oracle Traffic Director compresses and decompresses. For each metric, the object name in the SNMP MIB and the names of the corresponding element and attribute in the stats-xml report are provided.

**Table A-8 Compression and Decompression Metrics**

Metric	Object Name in the SNMP MIB	stats-xml Element: Attribute
Total number of requests compressed	countRequestsCompressed	compression: countRequests
Total number of input bytes for compression	countBytesForCompression	compression: bytesInput
Total number of output bytes after compression	countBytesCompressed	compression: bytesOutput
Average compression per page	pageCompressionAverage	compression: pageCompressionAverage
Overall compression ratio	compressionRatio	compression: compressionRatio
Total number of requests decompressed	countRequestsDecompressed	decompression: countRequests
Total number of input bytes for decompression	countBytesForDecompression	decompression: bytesInput
Total number of output bytes after decompression	countBytesDecompressed	decompression: bytesOutput

## Virtual Server Metrics

This section lists the metrics that Oracle Traffic Director tracks for individual virtual servers. For each metric, the object name in the SNMP MIB and the names of the corresponding element and attribute in the stats-xml report are provided.

**Table A-9 Virtual Server Metrics**

Metric	Object Name in the SNMP MIB	stats-xml Element: Attribute
Number of requests processed	vsRequests	request-bucket: countRequests
Number of octets received	vsInOctets	request-bucket: countBytesReceived

**Table A-9 (Cont.) Virtual Server Metrics**

<b>Metric</b>	<b>Object Name in the SNMP MIB</b>	<b>stats-xml Element: Attribute</b>
Number of octets transmitted	vsOutOctets	request-bucket: countBytesTransmitted
Number of 2xx (Successful) responses issued	vsCount2xx	request-bucket: count2xx
Number of 3xx (Redirection) responses issued	vsCount3xx	request-bucket: count3xx
Number of 4xx (Client Error) responses issued	vsCount4xx	request-bucket: count4xx
Number of 5xx (Server Error) responses issued	vsCount5xx	request-bucket: count5xx
Number of other (neither 2xx, 3xx, 4xx, nor 5xx) responses issued	vsCountOther	request-bucket: countOther
Number of 200 (OK) responses issued	vsCount200	request-bucket: count200
Number of 302 (Moved Temporarily) responses issued	vsCount302	request-bucket: count302
Number of 304 (Not Modified) responses issued	vsCount304	request-bucket: count304
Number of 400 (Bad Request) responses issued	vsCount400	request-bucket: count400
Number of 401 (Unauthorized) responses issued	vsCount401	request-bucket: count401
Number of 403 (Forbidden) responses issued	vsCount403	request-bucket: count403
Number of 404 (Not Found) responses issued	vsCount404	request-bucket: count404
Number of 503 (Unavailable) responses issued	vsCount503	request-bucket: count503
The total number of upgrade requests processed	websocketCountUpgradedRequests	websocket:countUpgradeRequests
Number of WebSocket requests that were denied upgrade by origin server	websocketCountUpgradeRejectedRequests	websocket:countUpgradeRequestsRejected
Number of WebSocket requests that were denied upgrade by server	websocketCountFailedStrictRequests	websocket:countUpgradeRequestsFailed
Number of active WebSocket connections	websocketCountActiveConnections	websocket:countActiveConnections
Total number of requests that were aborted	websocketCountAbortedRequests	websocket:countRequestsAborted
Total number of requests that were closed because of timeout	websocketCountTimeoutRequests	websocket:countRequestsTimedout

**Table A-9 (Cont.) Virtual Server Metrics**

<b>Metric</b>	<b>Object Name in the SNMP MIB</b>	<b>stats-xml Element: Attribute</b>
Number of bytes received from the clients	websocketCountBytesReceived	websocket:countBytesReceived
Number of bytes transmitted to the clients	websocketCountBytesTransmitted	websocket:countBytesTransmitted
Average duration of active time in millisecond	websocketMillisecondsConnectionActiveAverage	websocket:millisecondsConnectionActiveAverage
Total number of requests intercepted by webapp firewall	wafCountInterceptedRequests	webapp-firewall:countRequestsIntercepted
Total number of requests allowed by webapp firewall (allow action)	wafCountAllowedRequests	webapp-firewall:countRequestsAllowed
Total number of denied requests (deny action)	wafCountDeniedRequests	webapp-firewall:countRequestsDenied
Total number of dropped requests (drop action)	wafCountDroppedRequests	webapp-firewall:countRequestsDropped
Total number of redirected requests (redirect action)	wafCountRedirectedRequests	webapp-firewall:countRequestsRedirected
Total number of detected denied requests (deny action)	wafCountDenyDetectedRequests	webapp-firewall:countRequestsDenyDetected
Total number of detected dropped requests (drop action)	wafCountDropDetectedRequests	webapp-firewall:countRequestsDropDetected
Total number of detected redirected requests (redirect action)	wafCountRedirectDetectedRequests	webapp-firewall:countRequestsRedirectDetected

## CPU Metrics

This section lists the CPU-related metrics that Oracle Traffic Director tracks. For each metric, the object name in the SNMP MIB and the names of the corresponding element and attribute in the stats-xml report are provided.

**Table A-10 CPU Metrics**

<b>Metric</b>	<b>Object Name in the SNMP MIB</b>	<b>stats-xml Element: Attribute</b>
Percentage of the time that the CPU is idle	cpuidleTime	cpu-info: percentIdle
Percentage of the time the CPU is spending in user space	cpuUserTime	cpu-info: percentUser
Percentage of the time the CPU is spending in kernel space	cpuKernelTime	cpu-info: percentKernel

## Origin Server Metrics

This section lists the metrics that Oracle Traffic Director tracks for origin server pools and origin servers. For each metric, the object name in the SNMP MIB and the names of the corresponding element and attribute in the stats-xml report are provided.

**Table A-11 Origin Server Metrics**

Metric	Object Name in the SNMP MIB	stats-xml Element: Attribute
Number of times a request was retried (to same or different origin server)	originServerPoolCountRetries	server-pool: countRetries
Type of origin server pool (tcp or http)	originServerPoolType	server-pool:type
Status indicating whether the origin server is currently marked offline,online, or unavailable.	originServerStatus	origin-server: status
Flag indicating whether the node was dynamically discovered	originServerDiscoveryStatus	origin-server: flagDiscovered
Flag indicating whether the node is fully ramped up	originServerRampedupStatus	origin-server: flagRampedUp
Flag indicating whether the origin server is a backup node	originServerBackupStatus	origin-server: flagBackup
Total time, in seconds, since the origin server was marked online	originServerRunningTime	origin-server: secondsOnline
Total number of times the origin server was marked offline	originServerCountOffline	origin-server: countDetectedOffline
Total number of bytes transmitted to the origin server	originServerCountBytesTransmitted	origin-server: countBytesTransmitted
Total number of bytes received from the origin server	originServerCountBytesReceived	origin-server: countBytesReceived
Total number of open connections to the origin server for which requests are getting processed	originServerCountActiveConnections	origin-server: countActiveConnections
Total number of idle connections to the origin server	originServerCountIdleConnections	origin-server: countIdleConnections
Total number of active connections belonging to sticky requests when time statistics were collected	originServerCountActiveStickyConnections	origin-server:countActiveStickyConnections
Total number of times a connection to the origin server was attempted	originServerCountConnectAttempts	origin-server: countConnectAttempts
Total number of times an attempt to connect to the origin server failed	originServerCountConnectFailures	origin-server: countConnectFailures

**Table A-11 (Cont.) Origin Server Metrics**

<b>Metric</b>	<b>Object Name in the SNMP MIB</b>	<b>stats-xml Element: Attribute</b>
Total number of requests that were aborted when proxying requests with this origin server	originServerCountRequestsAborted	origin-server: countRequestsAborted
Total number of times the request timed out when sending or receiving data from the origin server	originServerCountRequestsTimedout	origin-server: countRequestsTimedout
Total number of requests served by the origin server	originServerCountRequests	origin-server: countRequests
Total number of health check requests	originServerCountHealthCheckRequests	origin-server: countHealthCheckRequests
Total number of connections closed	originServerCountConnectionsClosed	origin-server: countConnectionsClosed
Total number of keep-alive connections closed by the origin server	originServerCountConnectionsClosedByOriginServer	origin-server: countConnectionsClosedByOriginServer
Dynamically calculated keep-alive timeout value for the origin server	originServerSecondsKeepAliveTimeout	origin-server: secondsKeepAliveTimeout
Total number of sticky requests	originServerCountStickyRequests	origin-server: countStickyRequests
Dynamic weight detected based on response time (applicable when algorithm is least-response-time)	originServerWeightResponseTime	origin-server: weightResponseTime
Type of origin-server (generic/weblogic/undetected)	originServerType	origin-server: type
Average duration of active time in milliseconds	originServerMillisecondsConnectionActiveAverage	origin-server: millisecondsConnectionActiveAverage

## Failover Instance Metrics

This section lists the metrics for each VIP in the server instance.

These metrics show the current state of the failover instance, as well as which nodes are configured as primary and backup for a failover group.

**Table A-12 Failover Metrics**

Metric	Object Name in the SNMP MIB	stats-xml Element: Attribute
Actual current state of this failover instance. An integer (1 if active, 0 for not active).	failoverFlagActive	failover: flagActive
Name of the node which is configured as backup	failoverBackupInstance	failover: backupInstance
Name of the node which is configured as primary	failoverPrimaryInstance	failover: primaryInstance
Virtual IP address of the failover group	failoverVirtualIp	failover: virtualIp

## Cache Metrics

This section lists the reverse proxy caching metrics that Oracle Traffic Director tracks.

For each metric, the object name in the SNMP MIB and the names of the corresponding element and attribute in the stats-xml report are provided.

**Table A-13 Cache Metrics**

Metric	Object Name in the SNMP MIB	stats-xml Element: Attribute
Flag to indicate if cache is enabled	cacheEnabled	cache: flagEnabled
Total number of entries in the cache	cacheCountEntries	cache: countEntries
Amount of heap space used by cache content	cacheSizeHeap	cache: sizeHeapCache
Total number of times a cache lookup succeeded	cacheCountContentHits	cache: countContentHits
Total number of times a cache lookup failed	cacheCountContentMisses	cache: countContentMisses
Total number of times an entry was served from cache	cacheCountHits	cache: countHits
Total number of requests that were revalidated from the origin server	cacheCountRevalidationRequests	cache: countRevalidationRequests
Total number of times the revalidation requests failed	cacheCountRevalidationFailures	cache: countRevalidationFailures

## DMS Metrics Tables

This section lists the Oracle Traffic Director metric tables that are exposed to Oracle Dynamic Monitoring Service (DMS).

This section lists the Oracle Traffic Director metric tables that are exposed to Oracle Dynamic Monitoring Service (DMS).

[Table A-14](#) shows DMS metrics for the OTD\_Instance metric table.



The key columns are: `instanceName`.

**Table A-14 DMS Metrics: OTD\_Instance Metric Table**

Column Name	Type
<code>instanceName</code>	String
<code>configName</code>	String
<code>instanceVersion</code>	String
<code>timeStarted</code>	Long
<code>secondsRunning</code>	Long
<code>ticksPerSecond</code>	Long
<code>maxProcs</code>	Long
<code>maxThreads</code>	Long
<code>flagProfilingEnabled</code>	Boolean
<code>countRequests</code>	Long
<code>countBytesReceived</code>	Long
<code>countBytesTransmitted</code>	Long
<code>countOpenConnections</code>	Long
<code>count2xx</code>	Long
<code>count3xx</code>	Long
<code>count4xx</code>	Long
<code>count5xx</code>	Long
<code>countOther</code>	Long
<code>count200</code>	Long
<code>count302</code>	Long
<code>count304</code>	Long
<code>count400</code>	Long
<code>count401</code>	Long
<code>count403</code>	Long
<code>count404</code>	Long
<code>count503</code>	Long
<code>load1MinuteAverage</code>	Double
<code>load5MinuteAverage</code>	Double
<code>load15MinuteAverage</code>	Double
<code>inOctets</code>	Long
<code>outOctets</code>	Long
<code>req1MinuteAverage</code>	Double
<code>req5MinuteAverage</code>	Double
<code>req15MinuteAverage</code>	Double
<code>err1MinuteAverage</code>	Double
<code>err5MinuteAverage</code>	Double

**Table A-14 (Cont.) DMS Metrics: OTD\_Instance Metric Table**

Column Name	Type
err15MinuteAverage	Double
responseTime1Minute	Double
responseTime5Minute	Double
responseTime15Minute	Double

shows DMS metrics for the `OTD_Process` metric table.

The key columns are: (`instanceName`, `processId`).

**Table A-15 DMS Metrics: OTD\_Process Metric Table**

Column Name	Type
instanceName	String
processId	Long
configName	String
countThreads	Long
idleThreads	Long
sizeVirtual	Long
sizeResident	Long
fractionSystemMemoryUsage	Double
compression.countRequests	Long
compression.bytesInput	Long
compression.bytesOutput	Long
compression.ratioAverage	Double
compression.ratio	Double
decompression.countRequests	Long
decompression.bytesInput	Long
decompression.bytesOutput	Long
keepalive.countConnections	Long
keepalive.maxConnections	Long
keepalive.countHits	Long
keepalive.countFlushes	Long
keepalive.countRefusals	Long
keepalive.countTimeouts	Long
keepalive.secondsTimeout	Long
connectionQueue.connectionQueueId	String
connectionQueue.connectionQueueCount	Long
connectionQueue.connectionQueuePeak	Long

**Table A-15 (Cont.) DMS Metrics: OTD\_Process Metric Table**

Column Name	Type
connectionQueue.connectionQueueMax	Long
connectionQueue.connectionQueueTotal	Long
connectionQueue.connectionQueueTicksTotal	Long
connectionQueue.connectionQueueOverflows	Long
connectionQueue.connectionQueue1MinuteAverage	Double
connectionQueue.connectionQueue5MinuteAverage	Double
connectionQueue.connectionQueue15MinuteAverage	Double
dns.flagCacheEnabled	Boolean
dns.countCacheEntries	Long
dns.maxCacheEntries	Long
dns.countCacheHits	Long
dns.countCacheMisses	Boolean
dns.flagAsyncEnabled	Long
dns.countAsyncNameLookups	Long
dns.countAsyncAddrLookups	Long
dns.countAsyncLookupsInProgress	Long
threadPool.threadPoolId	String
threadPool.countIdleThreads	Long
threadPool.countThreads	Long
threadPool.countQueued	Long
threadPool.maxQueued	Long
threadPool.maxThreads	Long
threadPool.peakQueued	Long

shows DMS metrics for the OTD\_Cache metric table.

The key columns are: (instanceName).

**Table A-16 DMS Metrics: OTD\_Cache Metric Table**

Column Name	Type
instanceName	String
configName	String
flagEnabled	Boolean
countEntries	Long
sizeHeapCache	Long
countContentHits	Long
countContentMisses	Long

**Table A-16 (Cont.) DMS Metrics: OTD\_Cache Metric Table**

Column Name	Type
countHits	Long
countRevalidationRequests	Long
countRevalidationFailures	Long

shows DMS metrics for the `OTD_VirtualServer` metric table.

The key columns are: (`instanceName`, `vsName`).

**Table A-17 DMS Metrics: OTD\_VirtualServer Metric Table**

Column Name	Type
instanceName	String
vsName	String
configName	String
flagEnabled	Boolean
listeners	String
hosts	String
countRequests	Long
countBytesReceived	Long
countBytesTransmitted	Long
countOpenConnections	Long
count2xx	Long
count3xx	Long
count4xx	Long
count5xx	Long
countOther	Long
count200	Long
count302	Long
count304	Long
count400	Long
count401	Long
count403	Long
count404	Long
count503	Long
waf.countRequestsIntercepted	Long
waf.countAllowedRequests	Long
waf.countRequestsDenied	Long
waf.countRequestsDropped	Long

**Table A-17 (Cont.) DMS Metrics: OTD\_VirtualServer Metric Table**

Column Name	Type
waf.countRequestsRedirected	Long
waf.countRequestsDenyDetected	Long
waf.countRequestsDropDetected	Long
waf.countRequestsRedirectDetected	Long
websocket.countUpgradeRequests	Long
websocket.countUpgradeRequestsRejected	Long
websocket.countUpgradeRequestsFailed	Long
websocket.countRequestsAborted	Long
websocket.countRequestsTimedout	Long
websocket.countBytesReceived	Long
websocket.countBytesTransmitted	Long
websocket.countActiveConnections	Long
websocket.millisecondsConnectionActiveAverage	Double

shows DMS metrics for the OTD\_Route metric table.

The key columns are: (instanceName, vsName, routeName)

**Table A-18 DMS Metrics: OTD\_Route Metric Table**

Column Name	Type
instanceName	String
vsName	String
routeName	String
condition	String
configName	String
countRequests	Long
countBytesReceived	Long
countBytesTransmitted	Long
countOpenConnections	Long
count2xx	Long
count3xx	Long
count4xx	Long
count5xx	Long
countOther	Long
count200	Long
count302	Long
count304	Long

**Table A-18 (Cont.) DMS Metrics: OTD\_Route Metric Table**

Column Name	Type
count400	Long
count401	Long
count403	Long
count404	Long
count503	Long

shows DMS metrics for the `OTD_OriginServerPool` metric table.

The key columns are: (`instanceName`, `serverPoolName`)

**Table A-19 DMS Metrics: OTD\_OriginServerPool Metric Table**

Column Name	Type
<code>instanceName</code>	String
<code>serverPoolName</code>	String
<code>configName</code>	String
<code>serverPoolType</code>	String
<code>countRetries</code>	String
<code>serviceQueue.countQueued</code>	Long
<code>serviceQueue.countQueuedHighPriority</code>	Long
<code>serviceQueue.countQueuedLowPriority</code>	Long
<code>serviceQueue.countQueuedNormalPriority</code>	Long
<code>serviceQueue.countQueuedTimedout</code>	Long
<code>serviceQueue.countTotalQueued</code>	Long
<code>serviceQueue.countTotalQueuedHighPriority</code>	Long
<code>serviceQueue.countTotalQueuedLowPriority</code>	Long
<code>serviceQueue.countTotalQueuedNormalPriority</code>	Long
<code>serviceQueue.countTotalQueuedSticky</code>	Long
<code>serviceQueue.countTotalStickyToNonSticky</code>	Long
<code>serviceQueue.millisecondsQueuedHighPriorityAverage</code>	Double
<code>serviceQueue.millisecondsQueuedLowPriorityAverage</code>	Double
<code>serviceQueue.millisecondsQueuedNormalPriorityAverage</code>	Double

shows DMS metrics for the `OTD_OriginServer` metric table.

The key columns are: (`instanceName`, `serverPoolName`, `originServerName`)

**Table A-20 DMS Metrics: OTD\_OriginServer Metric Table**

Column Name	Type
instanceName	String
serverPoolName	String
originServerName	String
configName	String
type	String
status	String
discovered	Boolean
rampedup	Boolean
backup	Boolean
secondsOnline	Long
countDetectedOffline	Long
countBytesTransmitted	Long
countBytesReceived	Long
countActiveConnections	Long
countIdleConnections	Long
countActiveStickyConnections	Long
countConnectionsClosed	Long
countConnectionsClosedByOriginServer	Long
countConnectAttempts	Long
countConnectFailures	Long
countRequestsAborted	Long
countRequestsTimedout	Long
countRequests	Long
countHealthCheckRequests	Long
countStickyRequests	Long
weightResponseTime	Double
secondsKeepAliveTimeout	Long
websocket.countUpgradeRequests	Long
websocket.countUpgradeRequestsRejected	Long
websocket.countUpgradeRequestsFailed	Long
websocket.countRequestsAborted	Long
websocket.countRequestsTimedout	Long
websocket.countBytesReceived	Long
websocket.countBytesTransmitted	Long
websocket.countActiveConnections	Long
websocket.millisecondsConnectionActiveAverage	Double

shows DMS metrics for the OTD\_TcpOriginServer metric table.

The key columns are: (instanceName, serverPoolName, originServerName)

**Table A-21 DMS Metrics: OTD\_TcpOriginServer Metric Table**

Column Name	Type
instanceName	String
serverPoolName	String
originServerName	String
configName	String
status	String
backup	Boolean
secondsOnline	Long
countDetectedOffline	Long
countBytesTransmitted	Long
countBytesReceived	Long
countActiveConnections	Long
countClosedConnections	Long
countConnectAttempts	Long
countConnectFailures	Long
countRequestsAborted	Long
countRequestsTimedout	Long
countRequests	Long
countHealthCheckRequests	Long
millisecondsConnectionActiveAverage	Double

shows DMS metrics for the OTD\_TcpProxy metric table.

The key columns are: (instanceName, tcpProxyName)

**Table A-22 DMS Metrics: OTD\_TcpProxy Metric Table**

Column Name	Type
instanceName	String
tcpProxyName	String
configName	String
flagEnabled	Boolean
listeners	String
countActiveConnections	Long
countRequests	Long
countRequestsAborted	Long
countRequestsTimedout	Long



**Table A-22 (Cont.) DMS Metrics: OTD\_TcpProxy Metric Table**

Column Name	Type
countBytesReceived	Long
countBytesTransmitted	Long
millisecondsConnectionActiveAverage	String

shows DMS metrics for the `OTD_Listener` metric table.

The key columns are: (`instanceName`, `listenerName`)

**Table A-23 DMS Metrics: OTD\_Listener Metric Table**

Column Name	Type
instanceName	String
listenerName	String
configName	String
type	String(tcp/http)
addressType	String(inet/inet6/inet-sdp)
address	String
port	Long
sslEnabled	Boolean

shows DMS metrics for the `OTD_Failover` metric table.

The key columns are: (`virtualIp`)

**Table A-24 DMS Metrics: OTD\_Failover Metric Table**

Column Name	Type
primaryInstance	String
backupInstance	String
flagActive	Integer

shows DMS metrics for the `OTD_Partition` metric table.

The key columns are: (`instanceName`, `partitionName`)

**Table A-25 DMS Metrics: OTD\_Partition Metric Table**

Column Name	Type
instanceName	String
partitionName	String
configName	String
countRequests	Long

**Table A-25 (Cont.) DMS Metrics: OTD\_Partition Metric Table**

<b>Column Name</b>	<b>Type</b>
countBytesReceived	Long
countBytesTransmitted	Long
countOpenConnections	Long
count2xx	Long
count3xx	Long
count4xx	Long
count5xx	Long
countOther	Long
count200	Long
count302	Long
count304	Long
count400	Long
count401	Long
count403	Long
count404	Long
count503	Long

# B

## Web Application Firewall Examples and Use Cases

The attack prevention feature of web application firewall stands between the client and origin servers. If the web application firewall finds a malicious payload, it will reject the request, performing any one of the built-in actions. This section provides some basic information about how web application firewall works and how some rules are used for preventing attacks.

Some of the features of web application firewall are audit logging, access to any part of the request (including the body) and the response, a flexible rule engine, file-upload interception, real-time validation and buffer-overflow protection.

Web application firewall's functionality is divided into four main areas:

- **Parsing:** Parsers extract bits of each request and/or response, which are stored for use in the rules.
- **Buffering:** In a typical installation, both request and response bodies are buffered so that the module generally sees complete requests (before they are passed to the application for processing), and complete responses (before they are sent to clients). Buffering is the best option for providing reliable blocking.
- **Logging:** Logging is useful for recording complete HTTP traffic, allowing you to log all response/request headers and bodies.
- **Rule engine:** Rule engines work on the information from other components, to evaluate the transaction and take action, as required.

## Basics of Rules

The web application firewall rule engine is where gathered information is checked for any specific or malicious content.

This section provides information about basic rule-writing syntax, and rule directives for securing Web applications from attacks.

The main directive that is used for creating rules is `SecRule`. The syntax for `SecRule` is:

```
SecRule VARIABLES OPERATOR [TRANSFORMATION_FUNCTIONS, ACTIONS]
```

- **VARIABLES:** Specify where to check in an HTTP transaction. Web application firewall pre-processes raw transaction data, which makes it easy for rules to focus on the logic of detection. A rule must specify one or more variables. Multiple rules can be used with a single variable by using the `|` operator.
- **OPERATORS:** Specify how a *transformed* variable is to be analyzed. Operators always begin with an `@` character, and are followed by a space. Only one operator is allowed per rule.
- **TRANSFORMATION\_FUNCTIONS:** Change input in some way before the rule operator is run. A rule can specify one or more transformation functions.

- **ACTIONS:** Specify the required action if the rule evaluates to true, which could be, display an error message, step on to another rule, or some other task.

Here is an example of a rule:

```
SecRule ARGS|REQUEST_HEADERS "@rx <script" msg:'XSSAttack',deny,status:404
```

- `ARGS` and `REQUEST_HEADERS` are variables (request parameters and request headers, respectively).
- `@rx` is the regular expression operator. It is used to match a pattern in the variables.

In the example, the pattern is `<script`.

- `msg`, `deny` and `status` are actions to be performed if a pattern is matched.

The rule in the example is used to avoid XSS attacks, which is done by checking for a `<script` pattern in the request parameters and header, and an XSS Attack log message is generated. Any matching request is denied with a 404 status response.

## Rules Against Major Attacks

This section provides information about some rules that are used for preventing major attacks on Web applications.

### Brute Force Attacks

Brute force attacks involve an attacker repeatedly trying to gain access to a resource by guessing usernames, passwords, e-mail addresses, and similar credentials. Brute force attacks can be very effective if no protection is in place, especially when users choose passwords that are short and easy to remember.

A good way to defend against brute force attacks is to allow a certain number of login attempts, after which the login is either delayed or blocked. Here is an example of how this can be accomplished using Oracle Traffic Director web application firewall.

If your login verification page is situated at `yoursite.com/login` and is served by the virtual server `waf-vs`, then the following rules, in `waf-vs.conf` file configured at the virtual server level, will keep track of the number of login attempts by the users:

```
Block further login attempts after 3 failed attempts

Initialize IP collection with user's IP address
SecAction "initcol:ip=%{REMOTE_ADDR},pass,nolog"

Detect failed login attempts
SecRule RESPONSE_BODY "Unauthorized" "phase:4,pass,setvar:ip.failed_logins=
+1,expirevar:ip.failed_logins=60"

Block subsequent login attempts
SecRule IP:FAILED_LOGINS "@gt 2" deny
```

The rules initialize the IP collection and increment the field `IP:FAILED_LOGINS` after each failed login attempt. When more than three failed logins are detected, further attempts are blocked. The `expirevar` action is used to reset the number of failed login attempts to zero after 60 seconds, so the block will be in effect for a maximum of 60 seconds.

To use the persistent collection, IP, you should specify the path to store the persisted data using the `SecDataDir` directive. Since the scope of this directive is `Main`, it should be specified at the server level. This can be accomplished as follows:

```
The name of the debug log file
SecDebugLog ../logs/brute_force_debug_log

Debug log level
SecDebugLogLevel 3

Enable audit logging
SecAuditEngine On

The name of the audit log file
SecAuditLog ../logs/brute_force_audit_log

Path where persistent data is stored
SecDataDir "/var/run/otd/waf/"
```

If this rules file is called `waf-server.conf`, `<instance-dir>/config/server.xml` would look like this:

```
<server>
...
...
 <webapp-firewall-ruleset>/waf-rules/waf-server.conf</webapp-firewall-ruleset>
...
...
 <virtual-server>
 <name>waf-vs</name>
 <host>yoursite.com</host>
 ...
 <object-file>waf-vs-obj.conf</object-file>
 <webapp-firewall-ruleset>/waf-rules/waf-vs.conf</webapp-firewall-ruleset>
 </virtual-server>
...
...
</server>
```

Web application firewall and response body processing (equivalent of `SecResponseBodyAccess` directive) should be enabled for the `/login` URI in `waf-vs-obj.conf`. `waf-vs-obj.conf` would look like this:

```
<Object name="default">
<If $uri eq "/login">
AuthTrans fn="webapp-firewall" process-response-body="on"
</If>
...
...
</Object>
```

After 3 failed attempts to login, audit log would have the following message:

```
--5c4adf36-A--
[19/Mar/2013:05:06:57 --0700] ygfh3010000000000,0 127.0.0.1 49619 127.0.0.1 5021
--5c4adf36-B--
GET /acl/acl02.html HTTP/1.1
user-agent: curl/7.15.5 (x86_64-redhat-linux-gnu) libcurl/7.15.5 OpenSSL/0.9.8b zlib/
1.2.3 libidn/0.6.5
accept: /*/*
host: yoursite.com
```

```
authorization: Basic YWxwaGE6YmV0YQ==

--5c4adf36-F--
HTTP/1.1 403 Forbidden
status: 403 Forbidden
content-length: 208
content-type: text/html

--5c4adf36-H--
Message: Warning. Unconditional match in SecAction. [file "/waf-rules/waf-vs.conf"]
[line "10"]
Message: Access denied with code 403 (phase 2). Operator GT matched 2 at
IP:failed_logins. [file "/waf-rules/waf-vs.conf"] [line "25"]
Action: Intercepted (phase 2)
Stopwatch: 1363694817000000 898560 (- - -)
Stopwatch2: 1363694817000000 898560; combined=370, p1=14, p2=336, p3=0, p4=0, p5=19,
sr=131, sw=1, l=0, gc=0
Producer: ModSecurity for Apache/2.6.7 (http://www.modsecurity.org/).
Server: Oracle Traffic Director/11.1.1.7

--5c4adf36-Z--
```

## SQL Injection

SQL injection attacks can occur if an attacker is able to supply data to a Web application that is then used in unsanitized form in an SQL query. This can cause the SQL query to do something that is completely different from what was intended by the developers of the Web application. For example, an attacker can try deleting all records from a MySQL table, like this:

```
http://www.example.com/login.php?user=user1';DELETE%20FROM%20users--
```

This can be prevented by using the following directives:

```
SecDefaultAction "phase:2,log,auditlog,deny,status:403"
SecRule ARGS "(select|create|rename|truncate|load|alter|delete|update|insert|desc)
\s*" "t:lowercase,msg:'SQL Injection'"
```

Whenever the web application firewall engine spots such a request, something similar to the following code is logged to `audit_log`:

```
--3923b655-A--
[20/Mar/2013:02:58:35 --0700] Xkjsx601000000000,0 127.0.0.1 35971 127.0.0.1 5021
--3923b655-B--
GET /acl/acl02.html?user=user1';DELETE%20FROM%20users-- HTTP/1.1
host: waf.test.com
connection: close

--3923b655-F--
HTTP/1.1 403 Forbidden
status: 403 Forbidden
content-length: 208
content-type: text/html
connection: close

--3923b655-H--
Message: Access denied with code 403 (phase 2). Pattern match "(select|create|rename|
truncate|load|alter|delete|update|insert|desc)\s*" at ARGS:user. [file "/waf-rules/
sql_injection_attack.conf"] [line "2"] [msg "SQL Injection"]
Action: Intercepted (phase 2)
```

```
Stopwatch: 1363773515000000 668049 (- - -)
Stopwatch2: 1363773515000000 668049; combined=131, p1=8, p2=104, p3=0, p4=0, p5=19,
sr=0, sw=0, l=0, gc=0
Producer: ModSecurity for Apache/2.6.7 (http://www.modsecurity.org/).
Server: Oracle Traffic Director/11.1.1.7
```

```
--3923b655-Z--
```

In response to the attack, `SecDefaultAction` is applied, in which case the request is denied and logged, and the attacker receives a 403 error. If you would like a different action to take place, such as redirect the request to an HTML page with a customized warning content, specify it in the rule, as follows:

```
SecRule ARGS "(select|create|rename|truncate|load|alter|delete|update|insert|desc)
\s*" "t:lowercase,msg:'SQL Injection',redirect:http://yoursite.com/
invalid_request.html
```

## XSS Attacks

Cross-site scripting (XSS) attacks occur when user input is not properly sanitized and ends up in pages sent back to users. This makes it possible for an attacker to include malicious scripts in a page by providing them as input to the page. The scripts will be no different from scripts included in pages by creators of the website, and will thus have all the privileges of an ordinary script within the page, such as the ability to read cookie data and session IDs.

Here is an example of a simple rule to block `<script` in the request parameter:

```
SecDefaultAction phase:2,deny,status:403,log,auditlog
SecRule REQUEST_COOKIES|REQUEST_COOKIES_NAMES|REQUEST_FILENAME|ARGS_NAMES|ARGS|
XML:/* "(?i:<script.*?>)" "phase:
2,capture,t:none,t:htmlEntityDecode,t:compressWhiteSpace,t:lowercase,block,msg:'Cross
-site Scripting (XSS) Attack',id:'101'"
```

# Index

## A

---

activating statistics, [16-2](#), [16-3](#)  
archiving  
    log files, [13-7](#)

## C

---

CA  
    definition (Certificate Authority), [12-4](#)  
Certificate Authority  
    definition, [12-4](#)  
ciphers  
    definition, [12-2](#)  
connection queue information, [17-7](#)  
connections, [17-4](#)  
content compression  
    configuring for content compression, [17-19](#)

## D

---

DNS cache, [17-14](#)

## E

---

Elliptic Curve Cryptography, [12-4](#)  
enabling statistics, [16-2](#), [16-3](#)

## H

---

HTTP 1.1-style workload, [17-13](#)

## I

---

Instance, term, [1-7](#)

## K

---

keep-alive, [17-9](#)

## key

definition, [12-3](#)

## L

---

log files  
    archiving, [13-7](#)

## P

---

persistent connection information, [17-9](#)  
processes, [17-4](#)

## S

---

SNMP  
    basics, [16-8](#)  
    subagent, [16-9](#)  
statistics  
    activating, [16-2](#), [16-3](#)  
subagent  
    SNMP, [16-9](#)

## T

---

threads, [17-4](#)  
tips  
    general, [17-1](#)  
tuning the Web Server  
    threads, processes, and connections, [17-4](#)  
tuning tips  
    general, [17-1](#)

## V

---

Virtual Server, term, [1-7](#)