Oracle® Fusion Middleware

Administering Oracle Traffic Director

12c (12.2.1.1.0)

E71400-01

June 2016

Oracle Fusion Middleware Administering Oracle Traffic Director, 12c (12.2.1.1.0)

E71400-01

Contributors: Isvaran Krishnamurthy, Amit Gupta, Basant Kukreja, Julien Pierre, Hideaki Hayashi, Murthy Chintalapati, Sandhya Prasad, Zhong Xu, Sriram Natarajan, Meena Vyas, Prem Kumar Venkatasalapathy, Prashant Sharma, Raghunandan Seshadri, Sarath Babu, Megha Shivaprasad, Nanda Kishore

Contents

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:

- Working in a terminal window
- Executing operating system commands on UNIX-like platforms
- Oracle WebLogic Server administration tasks

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:

- *Release Notes for Oracle Traffic Director*
- *Installing Oracle Traffic Director*
- *WebLogic Scripting Tool Command Reference for Oracle Traffic Director*
- *Configuration File Reference for Oracle Traffic Director*

- *Using WebLogic Server MT*
- *Oracle Exalogic Elastic Cloud Documentation*

Conventions

The following text conventions are used in this document:

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

Part I

Getting Started

Part I contains the following chapters:

- Chapter 1, "Getting Started with Oracle Traffic Director" provides an overview of Oracle Traffic Director and its features, explains the related terminology, describes the administrative framework of the product.

1 Getting Started with Oracle Traffic Director

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

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. For more information about Exalogic, see the Oracle Exalogic Elastic Cloud documentation, `http://docs./cd/E18476_01/index.htm`. Oracle Traffic Director is also certified with various Fusion Middleware products.

Oracle Traffic Director is easy to install, configure, and use. It includes a simple, web browser based graphical user interface (using Oracle Enterprise Manager) as well as robust Command Line Interface using Oracle WebLogic WLST.

For high availability, you can set up pairs of Oracle Traffic Director instances for 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.

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.

This chapter provides information to help you understand and get started with Oracle Traffic Director. It contains the following sections:

- New Features in 12c
- Features of Oracle Traffic Director
- Typical Network Topology
- Oracle Traffic Director Terminology
- Oracle Traffic Director Deployment Scenarios
- Overview of Administration Tasks
- Accessing the Administration Interfaces
- Setting Up a Simple Load Balancer Using Oracle Traffic Director

1.1 New Features in 12c

This section describes features that have been added to the current release of Oracle Traffic Director:

- Section 1.1.1, "Weblogic Management Framework"
- Section 1.1.2, "WLST Commands"
- Section 1.1.3, "Multi-tenancy Support"
- Section 1.1.4, "Monitoring Enhancements"
- Section 1.1.5, "Oracle Fusion Middleware T2P Utility for Oracle Traffic Director"
- Section 1.1.6, "External Health Check Executable"
- Section 1.1.7, "Queueing with Request Limiting"
- Section 1.1.8, "Origin Server Traffic Control"
- Section 1.1.9, "Origin Server and Origin Server Pool Maintenance"
- Section 1.1.10, "Prioritized Backend Connection Management"
- Section 1.1.11, "Forward Proxy Support in Origin Server Pools"
- Section 1.1.12, "NZ Security Library"
- Section 1.1.13, "ModSecurity Upgrade"
- Section 1.1.14, "Support for Event Notifications"
- Section 1.1.15, "High availability using active-active failover mode"
- Section 1.1.16, "Status Listeners to monitor Oracle Traffic Director instances"
- Section 1.1.17, "Enabling FTP configuration for TCP proxies"

1.1.1 Weblogic Management Framework

Previous releases of Oracle Traffic Director did not require databases; this release supports database-based installation, or Restricted JRF-based installation. For more information on installation prerequisites, see Installing Oracle Traffic Director. This release of 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. For more information on the WebLogic Management Framework, see *What is the WebLogic Management Framework*.

OTD functionality is the same for both Database and Restricted-JRF mode based installations. The WebLogic Management Framework introduces these enhancements:

- Configuration is a post-installation task that begins with creating a domain, primarily by using Configuration Wizard. For more information, see *Installing and Configuring Oracle Traffic Director*.
- WLST: You can create, configure, and manage WebLogic domains using WLST, command-line utilities, and Fusion Middleware Control interchangeably. The method that you choose depends on whether you prefer using a graphical or command-line interface, and whether you can automate your tasks by using a script.
- Oracle Traffic Director configurations and instances are part of the WebLogic domain.

1.1.2 WLST Commands

The command line interface in Oracle Traffic Director WLST (Weblogic Scripting Tool). A number of WLST commands are used to configure and administer Oracle Traffic Director.

There are two types of WLST commands:

- Online Command
- Offline Command

Most of WLS commands are "only online commands" which requires connection to the WLS server. Some of them are "only offline commands" which does not required any connection to the WLS server. Some of them are both online and offline those can be invoked in both the ways.

All the commands configured through WLST, should be activated.

For more information see *WebLogic Scripting Tool Command Reference for Oracle Traffic Director*.

1.1.3 Multi-tenancy Support

Oracle WebLogic Server introduces a new concept Multi-tenancy, It provides a sharable infrastructure for use by multiple organizations that means, 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 that is dedicated to running application instances and related resources for a tenant.

In a typical deployment scenario WebLogic Server will be front ended by Oracle Traffic Director, so whenever partition related tasks are performed from WebLogic Server side Oracle Traffic Director need to be configured appropriately, so that it can successfully front end the Web Logic Partition.

With the Multi-tenancy support, Oracle Traffic Director will be configured automatically with out any explicit user action. For more information see, *Configuring Oracle Traffic Director* and *End-to-End Life Cycle Management*.

For more information, see *Oracle Fusion Middleware Using WebLogic Server Multitenant*.

1.1.4 Monitoring Enhancements

The monitoring subsystem, it gives the detailed information on the state of runtime components/processes which used to track the performance bottlenecks, to tune the system for optimal performance, to aid capacity planning, to predict failures, to do root cause analysis in case of failures. In some situations it is used to make sure that everything is functioning as per the requirements.

For more information, see Chapter 12, "Monitoring Oracle Traffic Director Instances".

1.1.5 Oracle Fusion Middleware T2P Utility for Oracle Traffic Director

The Oracle Fusion Middleware T2P utility allows you to move a Fusion Middleware environment from test to production with customization specific to the production environment. In 12.2.1, this utility supports Oracle Traffic Director.

For more information, see "Oracle Fusion Middleware T2P Utility for Oracle Traffic Director".

1.1.6 External Health Check Executable

Oracle Traffic Director now supports a generic health check hook-up mechanism, so that customers can write their 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.

For more information, see "Configuring Health-Check Settings to Use an External Executable".

1.1.7 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.

For more information, see Section 15.3, "Tuning the Thread Pool and Connection Queue".

1.1.8 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.

1.1.9 Origin Server and Origin Server Pool Maintenance

By using this feature, an origin server or origin server pool can be marked for maintenance, all the new connections and sessions to such an origin server or origin server pool are drained further.

A single pool can have a multiple partitions, but only one of those partitions may be pushed to a new pool. All the old requests must directed to the old pool until a time-out expires or the admin take some action.

All the Scaling up and scaling down process of origin servers need not require any functionality change on the Oracle Traffic Director. So Oracle Traffic Director admin will update the origin server as “disabled”. All on-going requests will still be processed by the disabled origin server. This is not just the runtime change but also the configuration will be updated to reflect the new state of the origin server.

1.1.10 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.

1.1.11 Forward Proxy Support in Origin Server Pools

Oracle Traffic Director always in contact to it's configured origin servers in a direct manner. The HTTP forward proxy server can be optionally associated with an origin server pool so that all member origin servers (of said pool) will be communicated with via the configured HTTP forward proxy server.

For more information, see Chapter 5, "Managing Origin-Server Pools".

1.1.12 NZ Security Library

The security library is NZ and the cert/key store is Oracle Wallet. NSS is no longer supported.

1.1.13 ModSecurity Upgrade

In Oracle Traffic Director 12c, we are upgrading the ModSecurity code used in WAF from version 2.6.7 to 2.8.0.

1.1.14 Support for Event Notifications

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

- Origin Server Status Change event
- Request limit exceeded event

For more information, see Chapter 13, "Event Notifications."

1.1.15 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.

For more information, see Chapter 14, "Configuring Oracle Traffic Director for High Availability".

1.1.16 Status Listeners to monitor Oracle Traffic Director instances

Oracle Traffic Director supports configuring dedicated Status Listeners to monitor Oracle Traffic Director instances.

For more information, see Section 9.6, "Configuring Status Listener".

1.1.17 Enabling FTP configuration for TCP proxies

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.

For more information, see Chapter 8, "Managing TCP Proxies"

1.2 Features of Oracle Traffic Director

Oracle Traffic Director provides the following features:

- **Advanced methods for load distribution**

  You can configure Oracle Traffic Director to distribute client requests to servers in the back end by using one of the following algorithms:

  – Round robin

  – Least connection count

  – Least response time

  – IP Hash

  - Weighted round robin
  - Weighted least connection count

- **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 server in the back end. 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. For more information, see the section Section 7.4, "Configuring Routes" and the *WebLogic Scripting Tool Command Reference for Oracle Traffic Director*.
- **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.

- **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 SSL 3.0, TLS 1.0, TLS 1.1, and TLS 1.2**

    To secure data during transmission and to ensure that only authorized users access the servers in the back end, you can configure SSL/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 Fusion Middleware Control or the WLST.

  - **Web Application Firewalls**

    Web application Firewalls enable 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.

  - **Single Sign-On with WebGate**

    WebGate enables single sign-on (SSO) for Oracle Traffic Director. WebGate examines incoming requests and determines whether the requested resource is protected, and if so, retrieves the session information for the user. Through WebGate, Oracle Traffic Director becomes an SSO partner application enabled to use SSO to authenticate users, obtain their identity by using Oracle Single Sign-On, and to make user identities available to web applications accessed through Oracle Traffic Director.

- **High availability**

  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 is available. This feature ensures continued availability even when some servers in the back end fail.

- **Failover for load balancing**

  Two 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: Fusion Middleware Control, 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.

■ **Virtualization-enabled solution**

Oracle Traffic Director can be deployed as a virtual appliance on cloud and virtual platforms.

After deploying Oracle Traffic Director as a physical application, you can create a virtual appliance from an Oracle Traffic Director instance or create an assembly containing multiple such appliances. You can then deploy the appliance or assembly on the Oracle Virtual Machine hypervisor. To enable such a deployment, Oracle provides an Oracle Traffic Director plug-in as part of Oracle Virtual Assembly Builder, a tool that you can use to build virtual appliances and assemblies from physical applications.

For more information about creating and deploying virtual assemblies containing Oracle Traffic Director instances, see the *Oracle Virtual Assembly Builder User's Guide*.

- **TCP load balancing**

  With TCP load balancing, Oracle Traffic Director accepts client connections and routes the requests to a pool of servers running TCP-based protocols.

1.3 Typical Network Topology

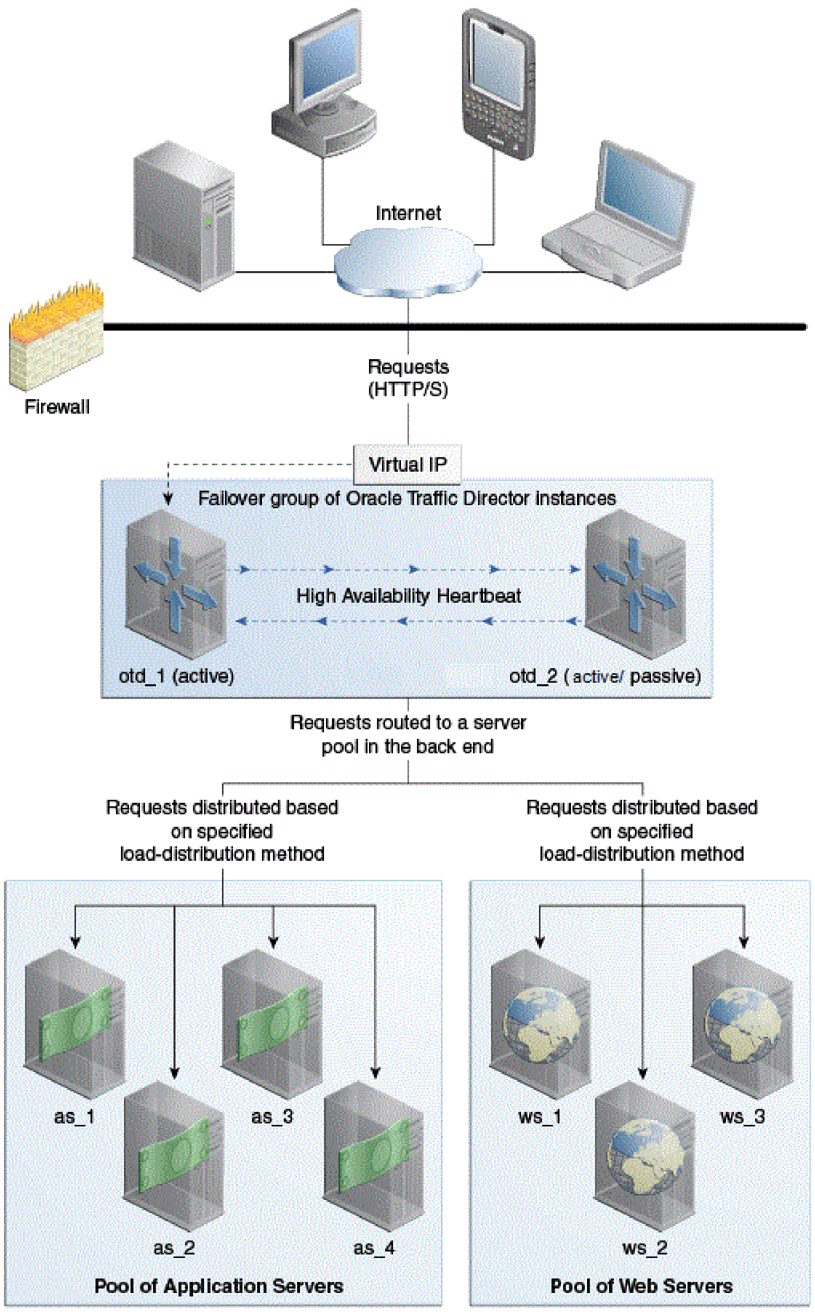
The network topology that you create for Oracle Traffic Director varies depending on your business requirements such as the number of back-end applications for which you want to use Oracle Traffic Director to balance requests, IT requirements such as security, and the features of Oracle Traffic Director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

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.

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 two homogenous Oracle Traffic Director instances running on different nodes forming an active-passive failover pair.

Figure 1–1 shows a typical Oracle Traffic Director network topology for a high-availability use case.

Figure 1–1 Oracle Traffic Director Network Topology



The topology shown in Figure 1–1 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 Section 14.1.2, "Failover configuration modes".

Note that Figure 1–1 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.

For more information about configuring Oracle Traffic Director instances in failover groups, see Section 14.2, "Creating and Managing Failover Groups."

1.4 Oracle Traffic Director Terminology

An Oracle Traffic Director configuration is a collection of elements that define the run-time behavior of an Oracle Traffic Director instance. An Oracle Traffic Director configuration contains information about various elements of an Oracle Traffic Director instance such as listeners, origin servers, failover groups, and logs.

The following table describes the terms used in this document when describing administrative tasks for Oracle Traffic Director.

Term	Description
Configuration	A collection of configurable elements (metadata) that determine the run-time behavior of an Oracle Traffic Director instance. 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.
Instance	An Oracle Traffic Director server that is instantiated from a configuration and deployed on an administration node.
Failover group	Two Oracle Traffic Director instances grouped by a virtual IP address (VIP), to provide high availability in 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. 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.
ORACLE_HOME	A directory of your choice in which you install the Oracle Traffic Director binaries.
DOMAIN_HOME	A path to the directory which contains Oracle Traffic Director domain
Fusion Middleware Control	A web-based graphical interface on the administration server that you can use to create, deploy, and manage Oracle Traffic Director instances.
Client	Any agent—a browser or an application, for example—that sends HTTP, HTTPS and TCP requests to Oracle Traffic Director instances.
Origin server	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. Origin servers can be application servers like Oracle WebLogic Server managed servers, web servers, and so on.
Origin-server pool	A collection of origin servers that host the same application or service that you can load-balance by using Oracle Traffic Director. 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.

Term	Description
Virtual server	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. 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.

1.5 Oracle Traffic Director Deployment Scenarios

Oracle Traffic Director can be used either as a physical application or as a virtual appliance.

- **Physical application**

  You can install Oracle Traffic Director on an Oracle Linux 6.5 system and run one or more instances of the product to distribute client requests to servers in the back end.

  For more information, see *Installing Oracle Traffic Director* .

- **Appliance running on a virtual platform**

  After deploying Oracle Traffic Director as a physical application, you can create a virtual appliance from an Oracle Traffic Director instance or create an assembly containing multiple such appliances. You can then deploy the appliance or assembly on the Oracle Virtual Machine hypervisor. To enable such a deployment, Oracle provides an Oracle Traffic Director plug-in as part of Oracle Virtual Assembly Builder, a tool that you can use to build virtual appliances and assemblies from physical applications.

  For more information about creating and deploying virtual assemblies containing Oracle Traffic Director instances, see the *Oracle Virtual Assembly Builder User's Guide*.

1.6 Overview of Administration Tasks

- 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 Admin Server, but uses the Admin Server in Oracle WebLogic Server.

  For more information, see *Installing Oracle Traffic Director* .

- Create a WebLogic domain for Oracle Traffic Director. For more information, see Section 2, "Configuring the WebLogic Server Domain for Oracle Traffic Director."

- 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 Section 1.7, "Accessing the Administration Interfaces."

- 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.

  For information, see Chapter 3, "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.

  For information, see Chapter 4, "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.

  For more information, see Chapter 5, "Managing Origin-Server Pools" and Chapter 6, "Managing Origin Servers."

- 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.

  For more information, see Chapter 7, "Managing Virtual Servers" and Chapter 9, "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.

  For more information, see Chapter 10, "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.

  For more information, see Chapter 11, "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.

For more information, see Chapter 12, "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.

  For more information, see Chapter 14, "Configuring Oracle Traffic Director for High Availability."

- 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.

  For more information, see Chapter 15, "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.

  For more information, see Chapter 16, "Diagnosing and Troubleshooting Problems."

1.7 Accessing the Administration Interfaces

This section contains the following topics:

- Section 1.7.1, "Accessing WebLogic Scripting Tool"
- Section 1.7.2, "Displaying Fusion Middleware Control"

1.7.1 Accessing WebLogic Scripting Tool

The command line interface in Oracle Traffic Director 12c is WLST (Weblogic Scripting Tool). 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. For more information about using WLST, see *WebLogic Scripting Tool Command Reference for Oracle Traffic Director*. For a complete list of Oracle Traffic Director WLST command, see *WebLogic Scripting Tool Command Reference for Oracle Traffic Director*. Oracle Traffic Director ships with 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`.

For more information about using WLST, see the *WebLogic Scripting Tool Command Reference for Oracle Traffic Director*.

1.7.1.1 Usage Modes

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

- Interactive Mode
- Script Mode
- Embedded Mode

For more information on using WLST in these modes, see *WebLogic Scripting Tool Command Reference for Oracle Traffic Director*.

1.7.2 Displaying 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 Oracle Fusion Middleware administrator user name and password and click **Login.**

You can now create Oracle Traffic Director configurations and deploy them as instances on administration nodes. For more information, see Chapter 3, "Managing Configurations."

1.8 Setting Up a Simple Load Balancer Using Oracle Traffic Director

This section describes how you can set up a load-balanced service using Oracle Traffic Director with the minimum necessary configuration. The purpose of this section is to reinforce and illustrate the concepts discussed earlier in this chapter and to prepare you for the configuration tasks described in the remaining chapters.

This section contains the following topics:

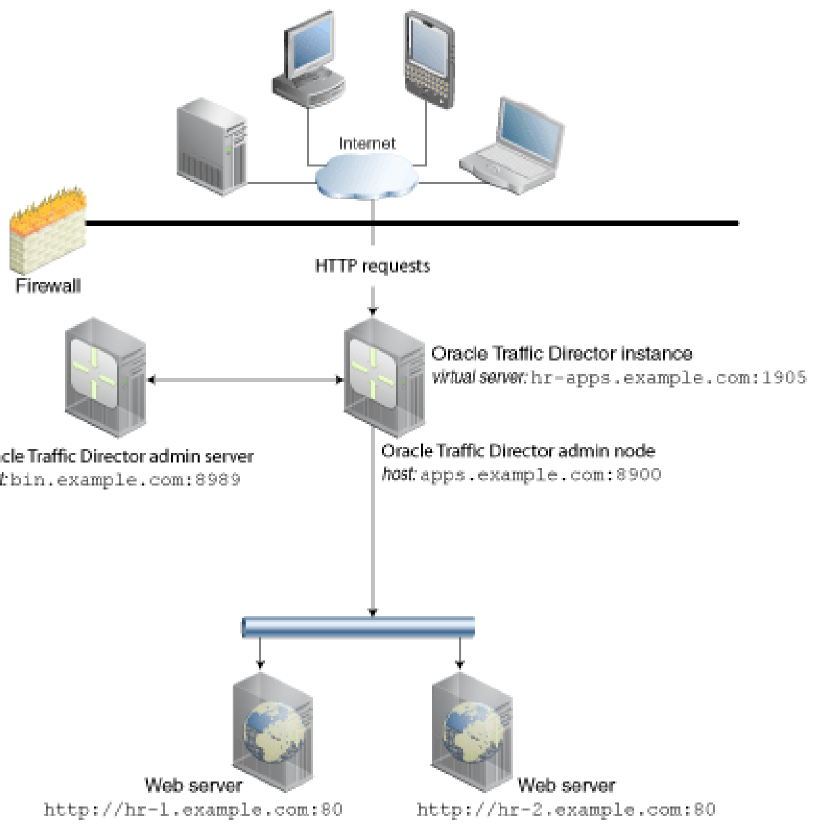
- Section 1.8.1, "Example Topology"
- Section 1.8.2, "Creating the Load Balancer for the Example Topology"
- Section 1.8.3, "Verifying the Load-Balancing Behavior of the Oracle Traffic Director Instance"

1.8.1 Example Topology

In this example, we will create a single instance of Oracle Traffic Director that will receive HTTP requests and distribute them to two origin servers in the back end, both serving identical content.

Figure 1–2 shows the example topology.

Figure 1–2 Oracle Traffic Director Deployment Example


Internet
Firewall
HTTP requests
Oracle Traffic Director instance
virtual server: hr-apps.example.com:1905
Oracle Traffic Director admin server
host: bin.example.com:8989
Oracle Traffic Director admin node
host: apps.example.com:8900
Web server
http://hr-1.example.com:80
Web server
http://hr-2.example.com:80

The example topology is based on the following configuration:

- Administration server host and port: `bin.example.com:8989`
- Administration node host and port: `apps.example.com:8900`
- Virtual server host and port 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 which the `DocumentRoot` directive of the web servers points, 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: Round robin

1.8.2 Creating the Load Balancer for the Example Topology

This section describes how to set up the topology described in Section 1.8.1, "Example Topology."

1. Install Oracle Traffic Director as described in *Installing Oracle Traffic Director* .
2. Create a configuration `hr-config` using the `otd_createConfiguration` WLST command.

```
props = {}
props['name'] = '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)
```

3. 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 OTD instance is running.

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

4. Start the Oracle Traffic Director instance that you just created by running the `start` WLST command.

```
start('otd_foo_machine1’)
```

Note: The steps in this procedure use only WLST, but you can use Fusion Middleware Control as well.

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

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

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`.

This section describes how you can verify the load-balancing behavior of the Oracle Traffic Director instance by using your browser.

Note:

- Make sure 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` administration node 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`.

Part II

Basic Administration

Part II contains the following chapters:

- Chapter 2, "Configuring the WebLogic Server Domain for Oracle Traffic Director" describes how to create and manage WebLogic domain for Oracle Traffic Director, so that Configurations and Instances can be created and managed.
- Chapter 3, "Managing Configurations" describes how to create and manage configurations, which are collections of metadata that determine the runtime behavior of Oracle Traffic Director instances.
- Chapter 4, "Managing Instances" describes how to create and manage Oracle Traffic Director instances.
- Chapter 5, "Managing Origin-Server Pools" describes how to create and manage pools of servers in the back end, to which Oracle Traffic Director instances can route client requests.
- Chapter 6, "Managing Origin Servers" describes how to add and manage servers in origin-server pools.
- Chapter 7, "Managing Virtual Servers" describes how to create and manage virtual servers to process client request, and how to create and manage route rules.
- Chapter 8, "Managing TCP Proxies" describes how to create and manage TCP proxies to handle TCP requests.
- Chapter 9, "Managing Listeners" describes how to create and manage HTTP listeners for virtual servers and TCP listeners for TCP proxies.

2 Configuring the WebLogic Server Domain for Oracle Traffic Director

This chapter provides instructions for creating a WebLogic domain with Oracle Traffic Director to be able to create and configure Oracle Traffic Director Configurations and Instances using the Oracle Fusion Middleware Configuration Wizard.

- Creating a Domain
- Understanding Oracle Traffic Director Domain types
- Creating a Collocated Oracle Traffic Director Domain
- Creating a Standalone Oracle Traffic Director Domain

2.1 Creating a Domain

After installation of Oracle Traffic Director, you should create a WebLogic domain with Oracle Traffic Director.

The following sections detail the steps necessary to create the domain.

2.2 Understanding Oracle Traffic Director Domain types

Oracle Traffic Director supports the following domain types:

- "Oracle Traffic Director with WebLogic Server (Collocated)"
- "Oracle Traffic Director without WebLogic Server (Standalone)"

2.2.1 Oracle Traffic Director with WebLogic Server (Collocated)

When a WebLogic Server domain is extended to Oracle Traffic Director, the Oracle Traffic Director instances and configurations can be managed like any other elements of the WebLogic Server domain. Specifically, the instances can be managed from Enterprise Manager Fusion Middleware Control, the WLST Command line interface, and WebLogic Server Node Manager.

To configure Oracle Traffic Director in a WebLogic Server domain, Oracle Traffic Director must be installed in an existing Oracle Fusion Middleware Infrastructure Oracle home.

Using Oracle Fusion Middleware Configuration Wizard from an Oracle home, you can configure or extend a WebLogic Server domain so that it contains one or more Oracle Traffic Director instances, in addition to the Administration Server, Managed Servers, and other elements of the domain.

Note: Remember that Oracle Traffic Director instances cannot be created using the Configuration Wizard; you must use either Enterprise Manager or WLST to do that.

2.2.2 Oracle Traffic Director without WebLogic Server (Standalone)

The Standalone Domain supports only one type of system component and has only one Node Manager. The custom WLST commands for administering Oracle Traffic Director cannot be used in a Standalone Domain.

Note: The custom WLST commands for administering OTD cannot be used in a standalone domain. For information about the standalone WLST commands, see *WebLogic Scripting Tool Command Reference for Oracle Traffic Director*

2.2.3 Collocated Domain Vs Standalone Domain

Use the following guidelines to decide on an Oracle Traffic Director domain option that will best suit your needs:

Table 2–1 Selecting Oracle Traffic Director Domain Configuration

Use Collocated Domain configuration if:	Use Standalone Domain configuration if:
You plan to use the Oracle Traffic Director instance as a front-end Web tier, which will be routing requests to an Oracle Fusion Middleware Infrastructure domain. This includes using Oracle Traffic Director in a domain configured for Oracle WebLogic Server MT.	You plan to use the Oracle Traffic Director instance as a front-end Web tier, which will route requests to a domain where Oracle Fusion Middleware Infrastructure is not available. For example, to front-end an Oracle WebLogic Server and Coherence domain.
You want to take advantage of advanced management capabilities of Fusion Middleware Control to manage your Oracle Traffic Director instances. You can also use WLST custom commands for the administration of Oracle Traffic Director. See *WebLogic Scripting Tool Command Reference for Oracle Traffic Director* for more information.	You do not wish to manage the Oracle Traffic Director instances with Fusion Middleware Control. Instead, you can use the WLST command-line and other features available in a standalone domain. Please note that not all offline wlst commands are available in the standalone domain. Standalone domains are not managed, and there are no management capabilities available (including FMWControl and wlst custom commands).
You want to scale out your domain to multiple hosts and still manage the Oracle Traffic Director instances from a single FMW Control.	

2.3 Creating a Collocated Oracle Traffic Director Domain

2.3.1 Creating a Domain using Restricted JRF Template (Recommended)

This is the recommended mode of creating an OTD domain. The restricted JRF mode, basically, creates a WLS runtime without a datasource connection. Hence, one does not need a running Oracle database to create the domain.

Follow these steps to create the OTD domain using Restricted JRF Template:

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

   The first screen of the Config Wizard appears (Create Domain).

2. Choose *Create a new domain*, and enter the desired domain home path.

3. Click **Next**. The *Templates* screen appears. In the Available Templates section, choose *Oracle Traffic Director - Restricted JRF 12.2.1 [otd]. Oracle Enterprise Manager - Restricted JRF*, *Oracle Restricted JRF* and *Weblogic Coherence Cluster Extension* will be automatically selected. Other templates can be selected. Refer to those individual component documents for details.

4. 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 will be needed to access WebLogic Server Control and Fusion Middleware Control.

6. Click **Next**. Choose the Domain Mode *Development*.

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

8. 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)`*

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

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

   Click **Next**.

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

11. Click **Next**. The *Clusters* screen appears. Do not add any clusters.

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

13. Click **Next**. The *Machines* screen appears. Click **Add**. Enter the following information:

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

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

15. Click **Create**. The Configuration Progress screen appears.
16. 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.

2.3.2 Creating a Domain using Full JRF Template

The full JRF template for domain configuration is used in these cases:

- Oracle Traffic Director and SOA are in the same domain
- The Domain is expected to support various components/scenarios such as CCWS, OPSS support for partitions, OPSS support for up-stack components of WebLogic, and so on.

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

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

   The first screen of the Config Wizard appears (Create Domain).
2. Choose *Create a new domain*, and enter the desired domain home path.
3. Click **Next**. The *TemplatesThe dependent templates will be automatically selected. The dependent templates are Oracle Enterprise Manager Plug-in for OTD - 12.1.4.0 [em]* and *Oracle JRF* screen appears. In the Available Templates section, choose *Oracle Traffic Director - 12.2.1 [otd].*
4. 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 will be 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: *`$WLS_ADMIN_PORT`*
    - Enable SSL: Use default `(unchecked)`
    - SSL Port: Use default `(disabled)`
    - Server Groups: Use default `(unspecified)`
13. Click **Next**. The *Node Manager* screen appears. Do not add any nodes. Use defaults (default: `Per Domain`). For Node Manager Credentials, enter the following:
    - User Name: *`$NM_USER`*
    - Password: *`$NM_PASSWORD`*
    - Confirm password: *`$NM_PASSWORD`*

    Click **Next**.
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**. 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 config wizard, the system component screen in the config wizard should be ignored while configuring Oracle Traffic Director. For creating Oracle Traffic Director configurations/instances, please use either Oracle Traffic Director custom wlst commands or FWMControl.

2.3.3 Creating a Repository using Repository Creation Utility

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.

2.3.4 Creating a Repository (Configurations without Restricted JRF Only)

When you install Oracle Traffic Director in a Collocated 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 Collocated domain, you will require a database and a repository with schema space for the domain. To create a repository:

1. 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:

```
welcome1
welcome1
welcome1
welcome1
welcome1
welcome1
```

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
```

2.3.5 Login 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.

2.4 Creating a Standalone Oracle Traffic Director Domain

You can create a Standalone Domain using either Configuration Wizard or the WLST.

2.4.1 Creating a Standalone Domain using the offline WLST commands

1. Launch the WLST command shell.

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

2. Run the following command to create an Oracle Traffic Director standalone domain. For more information, see the `otd_createStandaloneDomain` command in *WebLogic Scripting Tool Command Reference for Oracle Traffic Director*.

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

2.4.2 Creating a Standalone Domain using the Configuration Wizard

1. Invoke the installer

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

2. The *Create Domain* page appears. Use the default values. Click **Next**.
3. The *Templates* page appears. In the Available Templates section, choose *Oracle Traffic Director - Standalone 12.2.1 [otd]*. Click **Next**.

   Note: The **Basic Standalone System Component Domain - 12.2.1 [wlserver]** is selected by default.

   The other OTD templates *Oracle Traffic Director - 12.2.1 [otd]* and '*Oracle Traffic Director - Restricted JRF - 12.2.1 [otd]* are not applicable for standalone domain.

   ---

4. The *JDK Selection* page appears. Use the default JDK. Click **Next**.
5. The *System Components* page appears. Click **Next**.

   Note: You need not create system component, as OTD has commands for configuring system components.

   ---

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

2.4.3 Instance Management

- Start an Oracle Traffic Director instance.

  ```
  $DOMAIN_HOME/config/fmwconfig/components/OTD/instances/INSTANCE_
  NAME/bin/startserv
  ```
- Stop an Oracle Traffic Director instance.

  ```
  $DOMAIN_HOME/config/fmwconfig/components/OTD/instances/INSTANCE_NAME/bin/stop
  ```
- Restart an Oracle Traffic Director instance.

  ```
  $DOMAIN_HOME/config/fmwconfig/components/OTD/instances/INSTANCE_
  NAME/bin/restart
  ```
- Delete an Oracle Traffic Director instance

```
$DOMAIN_HOME/config/fmwconfig/components/OTD/instances/INSTANCE_NAME/bin/delete
```

- Reconfigure an Oracle Traffic Director instance.

  Reconfigure will dynamically apply configuration changes on instances without a server restart. Only dynamically reconfigurable changes in the configuration take effect. If there are any changes in the configuration that need a restart, restart required message will be shown on std out.

  ```
  $DOMAIN_HOME/config/fmwconfig/components/OTD/instances/$INSTANCE_
  NAME/bin/reconfig
  ```

- Rotate log files.

  The server saves the old log files and marks the saved files with a name that includes the date and time when they were rotated.

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

2.4.4 Monitoring Oracle Traffic Director Instance

The following commands can be used for monitoring the statistics pertaining to the OTD instance by executing the commands directly on the host where the OTD instance resides.

2.4.4.1 Using statistics

1. Launch WLST command shell

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

2. Execute the below command to view the statistics pertaining to the instance. Refer to otd_getStatsXml for more details.

   ```
   > print otd_getStatsXml({'domain-home': '$DOMAIN_HOME', 'instance': 'test'})

   <stats versionMajor="1" versionMinor="3" flagEnabled="1">
       <server id="test" versionServer="Oracle Traffic Director 12.2.1.0.0
   B20141215.181149 (Linux)" timeStarted="1418713548" secondsRunning="3"
   ticksPerSecond="1000" maxProcs="1" maxThreads="512" flagProfilingEnabled="1"
   load1MinuteAverage="1.020000" load5MinuteAverage="0.630000"
   load15MinuteAverage="0.350000" rateBytesTransmitted="13155"
   rateBytesReceived="62630" requests1MinuteAverage="0.000000"
   requests5MinuteAverage="0.000000" requests15MinuteAverage="0.000000"
   errors1MinuteAverage="0.000000" errors5MinuteAverage="0.000000"
   errors15MinuteAverage="0.000000" responseTime1MinuteAverage="0.000000"
   responseTime5MinuteAverage="0.000000" responseTime15MinuteAverage="0.000000">
           <virtual-server id="test" flagEnabled="1" listeners="*:30007">
               <request-bucket countRequests="0" countBytesReceived="0"
   countBytesTransmitted="0" rateBytesTransmitted="0" countOpenConnections="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"/>
               <websocket countUpgradeRequests="0" countUpgradeRequestsFailed="0"
   countUpgradeRequestsRejected="0" countActiveConnections="0"
   countRequestsAborted="0" countRequestsTimedout="0" countBytesReceived="0"
   countBytesTransmitted="0" millisecondsConnectionActiveAverage="0"/>
               <route id="default-route">
   ```

```
                <request-bucket countRequests="0" countBytesReceived="0"
countBytesTransmitted="0" rateBytesTransmitted="0" countOpenConnections="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"/>
            </route>
        </virtual-server>
...
     </server>
</stats>
```

3. Execute the below command to view the perfdump statistics pertaining to the instance. Refer to otd_getPerfDump for more details.

```
> print otd_getPerfDump({'domain-home': '$DOMAIN_HOME', 'instance': 'test'})
Oracle Traffic Director 12.2.1.0.0 B20141215.181149 (Linux)

Server started Mon Dec 15 23:05:47 2014
Process 24883 started Mon Dec 15 23:05:48 2014

ListenSocket http-listener-1:
------------------------
Address                 0.0.0.0:30007
Acceptor Threads        4
Default Virtual Server  test
```

2.4.4.2 Using SNMP

The following commands are used to stop/start SNMP agent.

- The below command starts OTD SNMP sub-agent on the host where OTD standalone domain resides. Refer to otd_startSnmpSubAgent for more details.

```
otd_startSnmpSubAgent({'domain-home': 'DOMAIN_HOME'})
```

- The below command stops OTD SNMP sub-agent on the host where OTD standalone domain resides. Refer to otd_stopSnmpSubAgent for more details.

```
otd_stopSnmpSubAgent({'domain-home': 'DOMAIN_HOME'})
```

Statistics collected by the SNMP subagent can also be viewed by using the snmpwalk command-line utility. snmpwalk is part of the net-snmp-utils RPM on OEL.

Run the snmpwalk command by explicitly specifying the required MIB object name.

```
> snmpwalk -m $ORACLE_HOME/otd/lib/snmp/ORACLE-TRAFFICDIRECTOR-MIB.txt -v 2c -c
public localhost:11161 ORACLE-TRAFFICDIRECTOR-MIB::cacheTable

ORACLE-TRAFFICDIRECTOR-MIB::cacheEnabled.1 = INTEGER: true(1)
ORACLE-TRAFFICDIRECTOR-MIB::cacheCountEntries.1 = Counter64: 0
ORACLE-TRAFFICDIRECTOR-MIB::cacheSizeHeap.1 = Counter64: 16558
ORACLE-TRAFFICDIRECTOR-MIB::cacheCountContentHits.1 = Counter64: 0
ORACLE-TRAFFICDIRECTOR-MIB::cacheCountContentMisses.1 = Counter64: 0
ORACLE-TRAFFICDIRECTOR-MIB::cacheCountHits.1 = Counter64: 0
ORACLE-TRAFFICDIRECTOR-MIB::cacheCountRevalidationRequests.1 = Counter64: 0
ORACLE-TRAFFICDIRECTOR-MIB::cacheCountRevalidationFailures.1 = Counter64: 0
ORACLE-TRAFFICDIRECTOR-MIB::cacheInstanceName.1 = STRING: otd_test_
mymachine.oracle.com
```

3 Managing Configurations

The first step toward creating a load-balanced service with Oracle Traffic Director is to create a configuration, which 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.

Note: For the definitions of the Oracle Traffic Director terminology—*configuration*, *administration node*, and *instance*, see Section 1.4, "Oracle Traffic Director Terminology." For information about the relationship between configurations, administration nodes, and instances, see Chapter 1, "Getting Started with Oracle Traffic Director."

This chapter contains the following topics:

- Creating a Configuration
- Viewing a List of Configurations
- Activate Configuration Changes
- Modifying a Configuration
- Copying a Configuration
- Deleting a Configuration

3.1 Creating a Configuration

You can create configurations by using either Fusion Middleware Control or the WLST.

Note: For information about using WLST, see Section 1.7.1, "Accessing WebLogic Scripting Tool."

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 its name.

Note: The server user that you specify for a configuration must meet the following requirements:

- When the administration server is running as `root`, the server user of a configuration must either be `root` or belong to the same group as the user that installed Oracle Traffic Director.
- When the administration server is running as a non-`root` user, the server user of a configuration must be the same as the administration server's server user.

Note that the nodes to which a configuration is deployed must be homogenous in terms of the user accounts and groups configured on those systems.

- A unique listener `host:port` combination for the default virtual server that you will create as part of the configuration.
- `host:port` addresses of the servers in the origin-server pool that you will create as part of the configuration.
- (optional) Host names of the administration nodes on which you want to create instances of the configuration.

Note: While creating a configuration by using the New Configuration wizard, you can choose to also instantiate the configuration on one or more administration nodes. The wizard enables you to do this by displaying the host names of the administration nodes that are registered with the administration server.

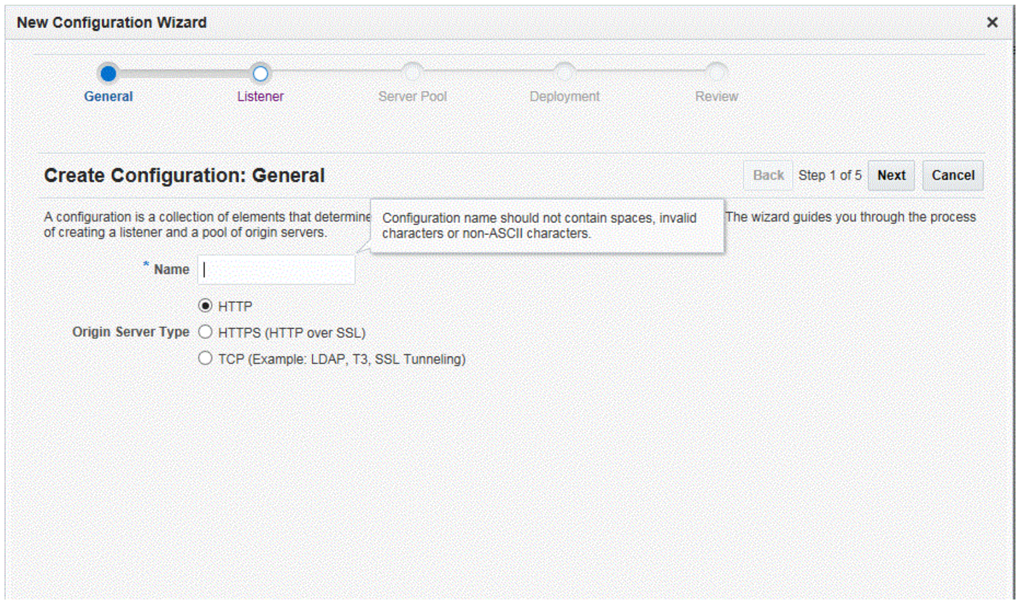
Creating a Configuration Using Fusion Middleware Control

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

1. Log in to Fusion Middleware Control for Traffic Director, as described in Section 1.7.2, "Displaying 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.

Figure 3–1 New Configuration Wizard



Create Configuration: General

A configuration is a collection of elements that determine [...] The wizard guides you through the process of creating a listener and a pool of origin servers.

Configuration name should not contain spaces, invalid characters or non-ASCII characters.

* Name

Origin Server Type: (●) HTTP ( ) HTTPS (HTTP over SSL) ( ) TCP (Example: LDAP, T3, SSL Tunneling)

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.

For example, the following command 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)

# Offline
readDomain('/export/2110_12c/iplanet/ias/server/work/TD_Linux2.6_
DBG.OBJ/domains/otd_domain')
props = {}
props['configuration'] = 'foo'
props['listener-port'] = '12345'
props['server-name'] = 'foo'
props['origin-server'] = 'www.mycompany.com:80'
otd_createConfiguration(props)
updateDomain()
```

```
closeDomain()
```

For more information about `otd_createConfiguration`, see the *WebLogic Scripting Tool Command Reference for Oracle Traffic Director* or run the command with the `--help` option.

For more information on the offline mode, see *Offline Commands* in the *WebLogic Scripting Tool Command Reference for Oracle Traffic Director*.

3.2 Viewing a List of Configurations

At any time, you can view a list of the available configurations by using either Fusion Middleware Control or the WLST.

Note: For information about invoking WLST, see Section 1.7.1, "Accessing WebLogic Scripting Tool."

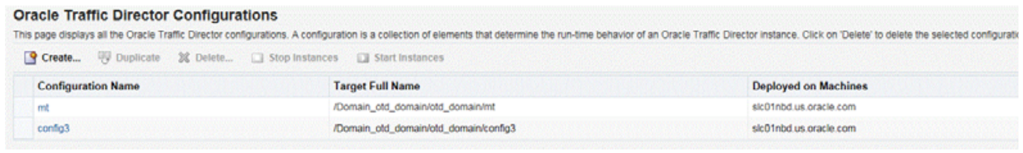
Viewing a List of Configurations Using Fusion Middleware Control

To view a list of the available configurations:

1. Log in to Fusion Middleware Control, as described in Section 1.7.2, "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, as shown in Figure 3–2.

Figure 3–2 List of Configurations



Oracle Traffic Director Configurations

This page displays all the Oracle Traffic Director configurations. A configuration is a collection of elements that determine the run-time behavior of an Oracle Traffic Director instance. Click on 'Delete' to delete the selected configurati

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:

```
# Online
otd_listConfigurations()

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

For more information on the offline mode, see *Offline Commands* in the *WebLogic Scripting Tool Command Reference for Oracle Traffic Director*.

3.3 Activate Configuration Changes

You can activate the configuration changes by using either Fusion Middleware Control or the WLST.

Note: Certain configuration changes cannot be applied dynamically without restarting the instances.

You can activate the configuration changes to the instances. The `activate` command will activate only the changes done after starting an edit session by executing the command `startEdit`. Also, the effect of this command is not just limited to OTD. All the changes done after starting an edit session to the various other components and managed servers will also be activated.

Activate a Configuration Using Fusion Middleware Control

1. Log in to Fusion Middleware Control for Traffic Director, as described in Section 1.7.2, "Displaying Fusion Middleware Control."
2. Click the unlock button just below Weblogic Domain at the upper left corner of the page.
3. In default Auto-Commit Mode is enabled.
   - **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 the commands executed by the WLST should be activated through the `Activate` command to activate changes.

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 is
completed.
Action completed.
wls:/mydomain/edit>
```

For more information about activate, see the *WebLogic Scripting Tool Command Reference for Oracle Traffic Director*.

Note: For some parameters, you can reconfigure an instance without restarting it. For more information, see Section 4.4, "Updating Oracle Traffic Director Instances Without Restarting."

3.4 Modifying a Configuration

After you create a configuration and create instances from it, you might need to change some of the settings—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.

Note: For information about invoking WLST, see Section 1.7.1, "Accessing WebLogic Scripting Tool."

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 Section 1.7.2, "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. For more information, see Section 10.4.2, "Update CRLs Automatically."
     - SSL/TLS caching preferences. For more information, see Section 15.8.1, "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.

     For more information, see Chapter 11, "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.

       For more information, see Section 15.7, "Tuning DNS Caching Settings."

  – Create, enable, disable, view, delete events for the configuration. For more information, see Section 4.6, "Controlling Oracle Traffic Director Instances Through Scheduled Events."

- **HTTP**, under **Advanced Settings**: 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.

  For more information, see Section 15.6, "Tuning HTTP Request and Response Limits."

- **Monitoring**, under **Advanced Settings**
  – Enable and disable statistics collection, profiling, and the SNMP subagent.
  – Specify the statistics-collection interval.

  For more information, see Chapter 12, "Monitoring Oracle Traffic Director Instances."

Note: For information about modifying origin servers, origin-server pools, listeners, and virtual servers, see:

- Section 5.3, "Modifying an Origin-Server Pool"
- Section 6.3, "Modifying an Origin Server"
- Section 7.3, "Modifying a Virtual Server"
- Section 9.3, "Modifying a Listener"

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 Section 3.3, "Activate Configuration Changes."

Caution: 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.

If you deploy the changes without stopping the running instances, an error would occur when you attempt to stop the instances later. For information about solving this problem, see Section 16.2.5, "Unable to stop instance after changing the temporary directory."

Modifying a Configuration Using WLST

WLST provides several commands (see Table 3–1) that you can use to change specific parameters of a configuration.

Note: For information about the WLST commands to change the properties of virtual servers, listeners, origin server pools, and origin servers in a configuration, see the following chapters:

- Chapter 5, "Managing Origin-Server Pools"
- Chapter 6, "Managing Origin Servers"
- Chapter 7, "Managing Virtual Servers"
- Chapter 9, "Managing Listeners"

Table 3–1 WLST Commands for Modifying a Configuration

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

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)
```

For more information about the WLST commands mentioned in this section, see the *WebLogic Scripting Tool Command Reference for Oracle Traffic Director* or run the commands with the `--help` option.

3.5 Copying a 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.

Note: For information about invoking WLST, see Section 1.7.1, "Accessing WebLogic Scripting Tool."

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 Section 1.7.2, "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)
```

For more information about `otd_copyConfiguration`, see the *WebLogic Scripting Tool Command Reference for Oracle Traffic Director*.

3.6 Deleting a Configuration

You can delete a configuration 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. For more information, see Section 14.2.2, "Managing Failover Groups."
- For information about invoking WLST, see Section 1.7.1, "Accessing WebLogic Scripting Tool."

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 Section 1.7.2, "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

Note: You cannot delete a configuration by using WLST if instances of the configuration are deployed to administration nodes, regardless of whether the instances are running or stopped.

To delete such a configuration by using WLST, you must first delete all of its instances.

To delete a configuration, run the `otd_deleteConfiguration` command, as shown in the following example:

```
# 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()
```

For more information about `otd_deleteConfiguration`, see the *WebLogic Scripting Tool Command Reference for Oracle Traffic Director*

For more information on the offline mode, see *Offline Commands* in the *WebLogic Scripting Tool Command Reference for Oracle Traffic Director*.

4 Managing Instances

An instance is an Oracle Traffic Director server running on an administration node, or on the administration server, and listening on one or more ports for requests from clients.

This chapter contains the following sections:

4.1 Creating Oracle Traffic Director Instances

You can create Oracle Traffic Director instances of a configuration by using either Fusion Middleware Control or the WLST.

Note: For information about using WLST, see Section 1.7.1, "Accessing WebLogic Scripting Tool."

Prerequisites for Creating Oracle Traffic Director Instances

To be able to create an instance, you should have done the following:

- Defined a configuration (see Section 3.1, "Creating a Configuration").

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 Section 1.7.2, "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**.
9. A message is displayed confirming the successful creation of the instance.
10. The Instances page is displayed, showing the instance that you just created.

Creating an Oracle Traffic Director Instance Using WLST

To create one or more Oracle Traffic Director instances, run the `otd_createInstance` command.

For example, the following command creates an instance of the configuration named `foo` on the 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()
```

Note: On windows, at any point only one domain with OTD instances is allowed. While there can be multiple domains those will not have OTD instances.

For more information about `otd_createInstance`, see the *WebLogic Scripting Tool Command Reference for Oracle Traffic Director*.

For more information on the offline mode, see *Offline Commands* in the *WebLogic Scripting Tool Command Reference for Oracle Traffic Director*.

4.2 Viewing a List of Oracle Traffic Director Instances

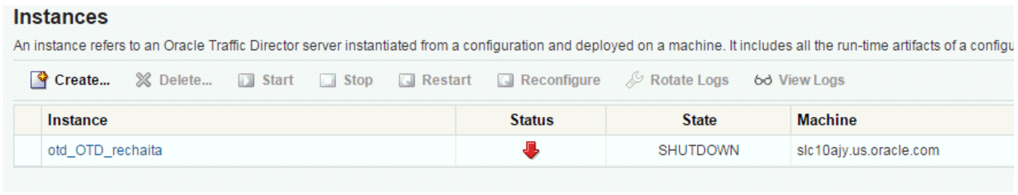
You can view a list of Oracle Traffic Director instances by using either Fusion Middleware Control or the WLST.

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 Section 1.7.2, "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 instance.
5. Click the Traffic Director Configuration In the Common Tasks pane.
6. Select Administration > Instances.
7. The Instances page is displayed, showing the instances of the configuration, as shown in Figure 4–1.

Figure 4–1 List of Instances



An instance refers to an Oracle Traffic Director server instantiated from a configuration and deployed on a machine. It includes all the run-time artifacts of a configu

Instance	Status	State	Machine
otd_OTD_rechaita	↓	SHUTDOWN	slc10ajy.us.oracle.com

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:

```
# 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()
```

For more information on the offline mode, see *Offline Commands* in the *WebLogic Scripting Tool Command Reference for Oracle Traffic Director*.

4.3 Starting, Stopping, and Restarting Oracle Traffic Director Instances

You can start, stop, and restart Oracle Traffic Director instances by using either Fusion Middleware Control or the 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 Section 1.7.2, "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')
```

For more information about the WLST commands mentioned in this section, see *WebLogic Scripting Tool Command Reference for Oracle Traffic Director*.

4.4 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.

Note: For information about invoking WLST, see Section 1.7.1, "Accessing WebLogic Scripting Tool."

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 Section 1.7.2, "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 as follows:

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

For more information see *WebLogic Scripting Tool Command Reference for Oracle Traffic Director*.

4.5 Deleting Oracle Traffic Director Instances

You can delete instances of a configuration by using either Fusion Middleware Control or the WLST.

Deleting an Oracle Traffic Director Instance Using Fusion Middleware Control

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 Section 1.7.2, "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.

Note: To delete an instance that is part of a failover group, you should first remove the instance from the failover group. For more information, see Section 14.2.2, "Managing Failover Groups."

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.

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()
```

For more information about `otd_deleteInstance`, see *WebLogic Scripting Tool Command Reference for Oracle Traffic Director*.

For more information on the offline mode, see *Offline Commands* in the *WebLogic Scripting Tool Command Reference for Oracle Traffic Director*.

4.6 Controlling Oracle Traffic Director Instances Through Scheduled Events

As an administrator, if you have to manage a large number of configurations and their instances, repetitive tasks such as restarting and reconfiguring instances of each configuration individually can become tedious. You can schedule *events* for administrative 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.

Note: For information about invoking WLST, see Section 1.7.1, "Accessing WebLogic Scripting Tool."

Managing Events Using Fusion Middleware Control

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

1. Log in to Fusion Middleware Control, as described in Section 1.7.2, "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 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.

   It lists events that are currently scheduled for the configuration.

   - 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.

     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 Section 3.3, "Activate 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)
  ```

For more information about the WLST commands mentioned in this section, see *WebLogic Scripting Tool Command Reference for Oracle Traffic Director*.

5 Managing Origin-Server Pools

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*. You can define several such 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.

This chapter describes how to create and manage 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

5.1 Creating an Origin-Server Pool

You can create an origin-server pool by using either Fusion Middleware Control or the WLST.

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 Section 3.3, "Activate Configuration Changes."
- For information about invoking WLST, see Section 1.7.1, "Accessing WebLogic Scripting Tool."

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. For more information, see Section 5.5, "Configuring an Oracle WebLogic Server Cluster as an Origin-Server Pool."

- The communication protocol—HTTP/S 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:** For Oracle Traffic Director to communicate with WebLogic Server over SDP, further configuration steps are required on the WebLogic Server. For more information about these configuration steps, see "Enabling Cluster-Level Session Replication Enhancements" in the *Oracle Fusion Middleware Exalogic Enterprise Deployment Guide*.

Creating an Origin-Server Pool Using Fusion Middleware Control

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

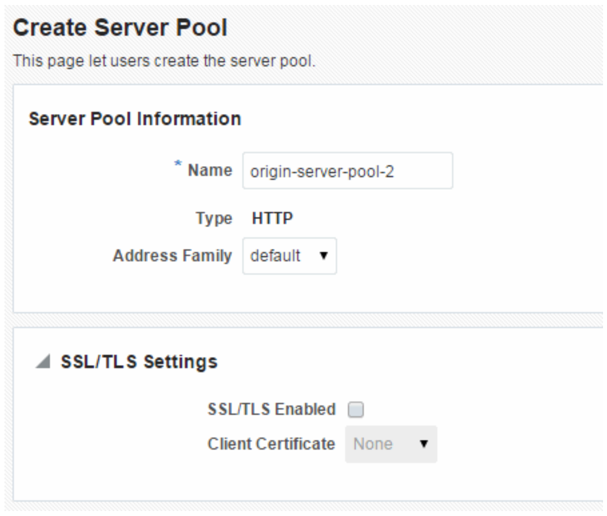
1. Log in to Fusion Middleware Control, as described in Section 1.7.2, "Displaying Fusion Middleware Control."
2. Click the WebLogic Domain button at the upper left corner of the page.
3. Select Administration > OTD Configurations.

   A list of the available configurations is displayed.
4. Select the configuration for which you want to 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.

Figure 5–1 Create Origin-Server Pool



Create Server Pool

This page let users create the server pool.

Server Pool Information

* Name: origin-server-pool-2

Type: HTTP

Address Family: default

SSL/TLS Settings

SSL/TLS Enabled: ☐

Client Certificate: None

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 Section 3.3, "Activate 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)
```

For more information about `otd_createOriginServerPool`, see *WebLogic Scripting Tool Command Reference for Oracle Traffic Director*.

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)
```

For more information about `otd_createOriginServerPool`, see *WebLogic Scripting Tool Command Reference for Oracle Traffic Director*.

5.2 Viewing a List of Origin-Server Pools

You can view a list of origin-server pools by using either Fusion Middleware Control or the WLST.

Note: For information about invoking WLST, see Section 1.7.1, "Accessing WebLogic Scripting Tool."

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 Section 1.7.2, "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 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.

You can view the properties of an origin-server pool in detail by clicking on its name.

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.

For more information about the WLST commands mentioned in this section, see *WebLogic Scripting Tool Command Reference for Oracle Traffic Director*.

5.3 Modifying an Origin-Server Pool

You can change the properties of an origin-server pool 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 should redeploy the configuration as described in Section 3.3, "Activate Configuration Changes."
- For information about invoking WLST, see Section 1.7.1, "Accessing WebLogic Scripting Tool."

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 Section 1.7.2, "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.

For more information about assigning weights to origin servers, see Section 6.3, "Modifying an Origin Server."

- Configure health-check settings. For more information, see Section 5.7, "Configuring Health-Check Settings for Origin-Server Pools."
- Specify whether Oracle Traffic Director should dynamically discover Oracle WebLogic Server managed servers in a cluster. For more information, see Section 5.5, "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. For more information, see Chapter 6, "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 Section 3.3, "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)
  ```

For a list of the properties that you can set or change by using the `otd_setOriginServerPoolProperties` and `otd_setHealthCheckProperties` commands, see *WebLogic Scripting Tool Command Reference for Oracle Traffic Director*.

5.4 Deleting an Origin-Server Pool

You can delete an origin-server pool by using either Fusion Middleware Control or the WLST.

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 Section 7.4, "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 Section 3.3, "Activate Configuration Changes."
- For information about invoking WLST, see Section 1.7.1, "Accessing WebLogic Scripting Tool."

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 Section 1.7.2, "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)
```

For more information about `otd_deleteOriginServerPool`, see the *WebLogic Scripting Tool Command Reference for Oracle Traffic Director*.

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

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.

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. 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.

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.

5.5.1 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`. For more information, see Section 5.7, "Configuring Health-Check Settings for Origin-Server Pools."

5.5.2 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 should redeploy the configuration as described in Section 3.3, "Activate Configuration Changes."
- For information about invoking WLST, see Section 1.7.1, "Accessing WebLogic Scripting Tool."

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 Section 1.7.2, "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 Oracle 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 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.

For more information about `otd_setHealthCheckProperties`, see *WebLogic Scripting Tool Command Reference for Oracle Traffic Director*.

5.6 Configuring a Custom Maintenance Page

It configures Oracle Traffic Director to serve a custom response code, and HTML page, when back-end servers maintenance required. Providing this type of message is better than having a gateway time-out, or creating other resources to host static content.

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)
```

For information about `otd_enableOriginServerPoolMaintenance`, `otd_disableOriginServerPoolMaintenance`, and `otd_getOriginServerPoolMaintenanceProperties`, see *WebLogic Scripting Tool Command Reference for Oracle Traffic Director*.

5.7 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: For information about invoking WLST, see Section 1.7.1, "Accessing WebLogic Scripting Tool."

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. For more information, see Section 5.5, "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.

Configurable Health-Check Settings

Table 5–1 lists the health-check settings that you can configure for each origin-server pool in a configuration.

Table 5–1 Health-Check Parameters

Parameter	Default Value
The type of connection—HTTP, TCP, or COMMAND—that Oracle Traffic Director should attempt with the origin server to determine its health. ■ TCP connection: Oracle Traffic Director attempts to open a TCP connection to each origin server. ■ HTTP request: Oracle Traffic Director sends an HTTP GET or OPTIONS request to each origin server in the pool, and checks the response to determine the availability *and* health of the origin server. **Note:** If you want to enable dynamic discovery of Oracle WebLogic Server managed servers in a cluster, then the health-check connection type must be set to HTTP. ■ COMMAND: Oracle Traffic Director uses an external executable created by the customer to monitor the health of specific origin servers. This mechanism is useful when you want to have a protocol-level health check monitor for the origin servers, which provide different services.	HTTP
The frequency at which health-check requests should be sent.	30 seconds
The duration after which a health-check request should be timed out if no response is received from the origin server.	5 seconds
The number of times that Oracle Traffic Director should attempt to connect to an origin server in the pool, before marking it as unavailable.	5
The HTTP request method—GET or OPTIONS—that should be sent.	OPTIONS
The URI that should be sent for HTTP requests.	/
The HTTP response codes that Oracle Traffic Director can accept as indicators of a healthy origin server. By default, Oracle Traffic Director accepts response codes from `1xx` to `4xx` as indicators of a healthy origin server.	
For HTTP GET health-check requests, a regular expression for the response body that Oracle Traffic Director can accept as the indicator of a healthy origin server	

Table 5–1 (Cont.) Health-Check Parameters

Parameter	Default Value
For HTTP GET health-check requests, the maximum number of bytes in the response body that Oracle Traffic Director should consider when comparing the response body with the specified acceptable response body.	2048

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 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 Fusion Middleware Control, as described in Section 1.7.2, "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)
```

For more information about the commands mentioned in this section, see *WebLogic Scripting Tool Command Reference for Oracle Traffic Director*.

5.7.1 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.

5.7.1.1 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/myhcscript` 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/myhcscript'
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.

5.7.1.2 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/myhcscript -h server1.myserver.com -p 389 -t 10
```

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

Table 5–2 Argument Parameters

Option	Meaning
`-h`	Origin server host.
`-p`	Origin server port.
`-t`	Health-check timeout.

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

```
/path/myhcscript --secure -d /dbpath
```

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

```
/path/myhcscript --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.

5.7.1.3 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 5–3 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.

6 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. A group of origin servers providing the same service is called an *origin server pool*.

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
- Managing Ephemeral Ports
- Removing an Origin Server from a Pool

6.1 Adding an Origin Server to a Pool

You can add an origin server to an origin-server pool by using either Fusion Middleware Control or the WLST.

Note:

- 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 Section 3.3, "Activate Configuration Changes."
- For information about using WLST see Section 1.7.1, "Accessing WebLogic Scripting Tool."

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`.

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:

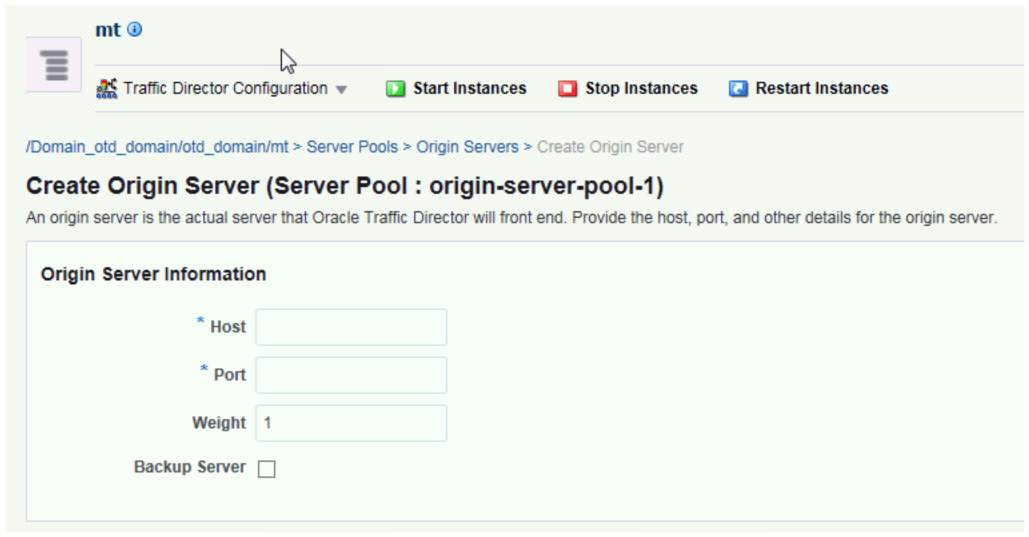
1. Log in to Fusion Middleware Control, as described in Section 1.7.2, "Displaying Fusion Middleware Control."
2. Click the WebLogic Domain button at the upper left corner of the page.
3. Select Administration > OTD Configurations.

   A list of the available configurations is displayed.
4. Select the configuration for which you want to 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 opens

Figure 6–1 New Origin Server


mt
Traffic Director Configuration
Start Instances
Stop Instances
Restart Instances
/Domain_otd_domain/otd_domain/mt > Server Pools > Origin Servers > Create Origin Server

Create Origin Server (Server Pool : origin-server-pool-1)

An origin server is the actual server that Oracle Traffic Director will front end. Provide the host, port, and other details for the origin server.

Origin Server Information

* Host	
* Port	
Weight	1
Backup Server	☐

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)
```

For more information about `otd_createOriginServer`, see *WebLogic Scripting Tool Command Reference for Oracle Traffic Director*.

6.2 Viewing a List of Origin Servers

You can view a list of origin servers by using either Fusion Middleware Control or the WLST.

Note: For information about invoking WLST, see Section 1.7.1, "Accessing WebLogic Scripting Tool."

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 Section 1.7.2, "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.

You can view and edit the properties of an origin server by clicking on its name.

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.

For more information about the `otd_listOriginServers` and `otd_getOriginServerProperties` commands, see *WebLogic Scripting Tool Command Reference for Oracle Traffic Director*.

6.3 Modifying an Origin Server

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. For more information about those properties, see the *Before you begin* section.
- 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.

You can change the properties of an origin server by using either Fusion Middleware Control or the WLST.

Note:

- When you change the properties of an origin server in 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 Section 3.3, "Activate Configuration Changes."
- For information about invoking WLST, see Section 1.7.1, "Accessing WebLogic Scripting Tool."

Changing the Properties of 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 Section 1.7.2, "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.
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)
```

For a list of the properties that you can change by using `otd_setOriginServerProperties`, see *WebLogic Scripting Tool Command Reference for Oracle Traffic Director*.

6.4 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.

6.5 Removing an Origin Server from a Pool

You can remove an origin server from a pool by using either Fusion Middleware Control or the WLST.

Note:

- When dynamic discovery is enabled (see Section 5.5, "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 reconfigure the Oracle Traffic Director instance, the instance might not start if no valid origin servers remain in the pool.
- For information about invoking WLST, see Section 1.7.1, "Accessing WebLogic Scripting Tool."

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 Section 1.7.2, "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)
```

For more information about `otd_deleteOriginServer`, see *WebLogic Scripting Tool Command Reference for Oracle Traffic Director*.

7 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:

7.1 Creating a Virtual Server

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 the 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 Section 3.3, "Activate Configuration Changes."
- For information about using WLST, see Section 1.7.1, "Accessing WebLogic Scripting Tool."

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 Chapter 9, "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.

  For more information, see Section 10.1.6, "About Strict SNI Host Matching."

- The name of the origin-server pool to which the virtual server should forward requests. For information about creating origin-server pools, see Chapter 5, "Managing Origin-Server Pools."

Creating a Virtual Server Using Fusion Middleware Control

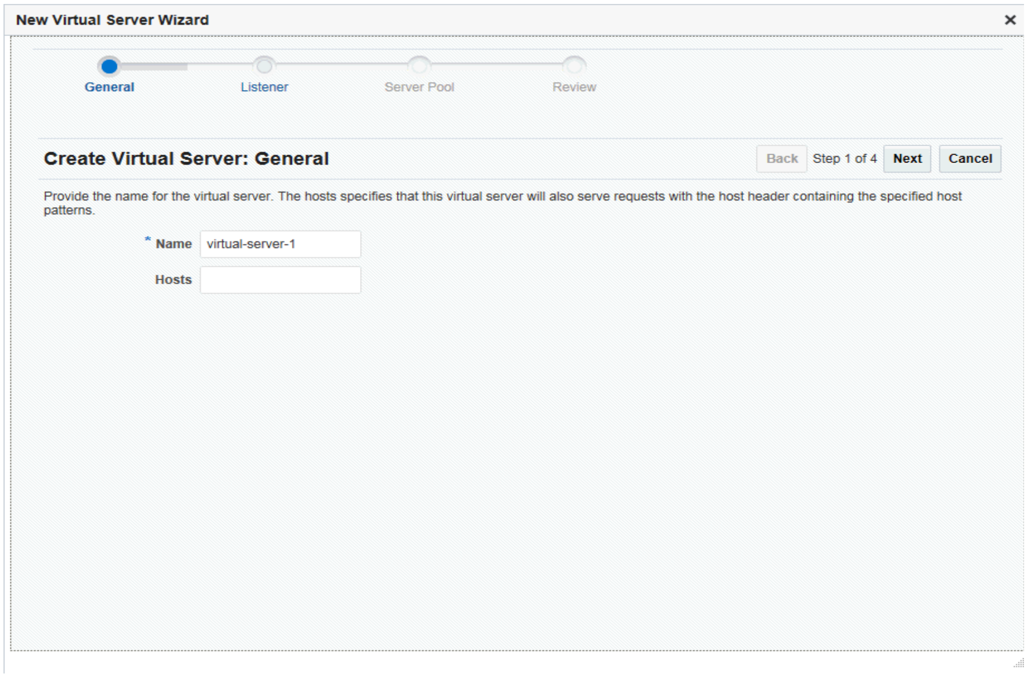
To create a virtual server by using the Fusion Middleware Control, do the following:

1. Log in to Fusion Middleware Control, as described in Section 1.7.2, "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 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 starts.

Figure 7–1 New Virtual Server Wizard



Create Virtual Server: General

Provide the name for the virtual server. The hosts specifies that this virtual server will also serve requests with the host header containing the specified host patterns.

* Name virtual-server-1

Hosts

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)
```

For more information about `otd_createVirtualServer`, see *WebLogic Scripting Tool Command Reference for Oracle Traffic Director*.

7.2 Viewing a List of Virtual Servers

You can view a list of virtual servers by using either Fusion Middleware Control or the WLST.

Note: For information about invoking WLST, see Section 1.7.1, "Accessing WebLogic Scripting Tool."

Viewing 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 Section 1.7.2, "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.

   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.

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.

For more information about the `otd_listVirtualServers` and `otd_getVirtualServerProperties` commands, see *WebLogic Scripting Tool Command Reference for Oracle Traffic Director*.

7.3 Modifying a Virtual Server

You can modify virtual servers by using either Fusion Middleware Control or the WLST.

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 Section 3.3, "Activate Configuration Changes."
- For information about invoking WLST, see Section 1.7.1, "Accessing WebLogic Scripting Tool."

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 Section 1.7.2, "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 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. For more information about how Oracle Traffic Director uses host patterns, see the *Before you begin* section.
   - Add and remove HTTP listeners. For information about creating HTTP listeners, see Section 9.1, "Creating a Listener."
   - Enable SSL/TLS, by associating an RSA or an ECC certificate (or both) with the virtual server. For more information, see Section 10.1.3, "Associating Certificates with Virtual Servers."
   - 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. For more information, see Section 12.3, "Configuring URI Access to Statistics Reports."
   - 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. For more information, see Section 7.4, "Configuring Routes."
- **Routes**: Create, change, and delete rules for routing requests to origin servers. For more information, see Section 7.4, "Configuring Routes."
- **Caching**: Create, change, and delete rules for caching responses received from origin servers. For more information, see Section 7.10, "Configuring Caching Parameters."
- **Compression**: Create, change, and delete rules for compressing responses from origin servers before forwarding them to the clients. For more information, see Section 15.10, "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. For more information, see Section 10.7, "Preventing Denial-of-Service Attacks."
- **Bandwidth Limits**: Enable, change, and delete rules for limiting the number and rate of requests received by the virtual server. For more information, see Section 10.7, "Preventing Denial-of-Service Attacks."
- **Content Serving**: Create, change, and delete rules for static content serving to origin servers.
- **Webapp Firewall**: Enable or disable webapp firewall rule set, specify rule set patterns and install rule set files. For more information, see Section 10.5, "Managing Web Application Firewalls."
- **Logging**: Define a server log file and location that is specific to the virtual server. For more information, see Section 11.3, "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 (see Table 7–1) that you can use to change specific parameters of a virtual server.

Table 7–1 WLST Commands for Modifying a Virtual Server

Task/s	CLI 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: Section 10.1.3, "Associating Certificates with Virtual Servers" and Section 11.3, "Configuring Log Preferences")	`otd_setVirtualServerProperties`

Create and manage routes (see Section 7.4, "Configuring Routes")	`otd_createRoute`
	`otd_listRoutes`
	`otd_deleteRoute`
	`otd_setRouteProperties`
	`otd_getRouteProperties`

Create and manage caching rules (see Section 7.7, "Caching in Oracle Traffic Director"	`otd_createCacheRule`
	`otd_listCacheRules`
	`otd_deleteCacheRule`
	`otd_getCacheRuleProperties`
	`otd_setCacheRuleProperties`

Create and manage compression rules (see Section 15.10, "Enabling and Configuring Content Compression")	`otd_createCompressionRule`
	`otd_setCompressionRuleProperties`
	`otd_deleteCompressionRule`
	`otd_listCompressionRules`
	`otd_getCompressionRuleProperties`

Change request limiting settings (see Section 10.7, "Preventing Denial-of-Service Attacks")	`otd_createRequestLimit`
	`otd_deleteRequestLimit`
	`otd_getRequestLimitProperties`
	`otd_listRequestLimits`
	`otd_setRequestLimitProperties`

Create and manage content rules (see Section 7.11, "Content Serving")	`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)
```

For more information about the WLST commands, see *WebLogic Scripting Tool Command Reference for Oracle Traffic Director*.

7.4 Configuring Routes

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. For more information, see *WebLogic Scripting Tool Command Reference for Oracle Traffic Director*.

You can configure routes in a virtual server by using either Fusion Middleware Control or the WLST.

Note: For information about invoking WLST, see Section 1.7.1, "Accessing WebLogic Scripting Tool."

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 Section 1.7.2, "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 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, and String Interpolation" in the *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)
```

For more information about the WLST commands mentioned in this section, see *WebLogic Scripting Tool Command Reference for Oracle Traffic Director*.

7.5 Copying a Virtual Server

You can copy a virtual server by using either Fusion Middleware Control or the WLST.

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 Section 3.3, "Activate Configuration Changes."
- For information about invoking WLST, see Section 1.7.1, "Accessing WebLogic Scripting Tool."

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 Section 1.7.2, "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)
```

For more information about `otd_copyVirtualServer`, see *WebLogic Scripting Tool Command Reference for Oracle Traffic Director*.

7.6 Deleting a Virtual Server

You can delete virtual servers by using either Fusion Middleware Control or the WLST.

Note: For information about invoking WLST, see Section 1.7.1, "Accessing WebLogic Scripting Tool."

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 Section 1.7.2, "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)
```

For more information about `otd_deleteVirtualServer`, see *WebLogic Scripting Tool Command Reference for Oracle Traffic Director*.

7.7 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. For more information, see `http://www.w3.org/Protocols/rfc2616/rfc2616-sec13.html`.

7.8 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.

7.9 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

7.10 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 Section 1.7.2, "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 Section 1.7.2, "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:

1. 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.

   ---

2. 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" 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.

3. 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" 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)
  ```

For more information about the WLST commands mentioned in this section, see *WebLogic Scripting Tool Command Reference for Oracle Traffic Director*.

7.11 Content Serving

Content-rule supports only static content serving. No dynamic content serving is supported and it can be created only based on uri-prefix. Uri-prefix should be unique across all the content rules. OTD admin supports static content serving only for content-rule, mime types and file cache.

OTD 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.

Configuring 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 Section 1.7.2, "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.

      In the **Name** field, enter a name for the new content rule.

   b. In the **URI Prefix** field, enter the specified URI for that content rule.

   c. 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_listMimeTypes` 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)
  ```

For more information about the WLST commands mentioned in this section, see *WebLogic Scripting Tool Command Reference for Oracle Traffic Director*.

8 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:

8.1 Creating a TCP Proxy

You can create TCP proxies by using either Fusion Middleware Control or the WLST.

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 Section 3.3, "Activate Configuration Changes."
- For information about using WLST, see Section 1.7.1, "Accessing WebLogic Scripting Tool."

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. For information about creating origin-server pools, see Chapter 5, "Managing Origin-Server Pools."

Creating a TCP Proxy Using Fusion Middleware Control

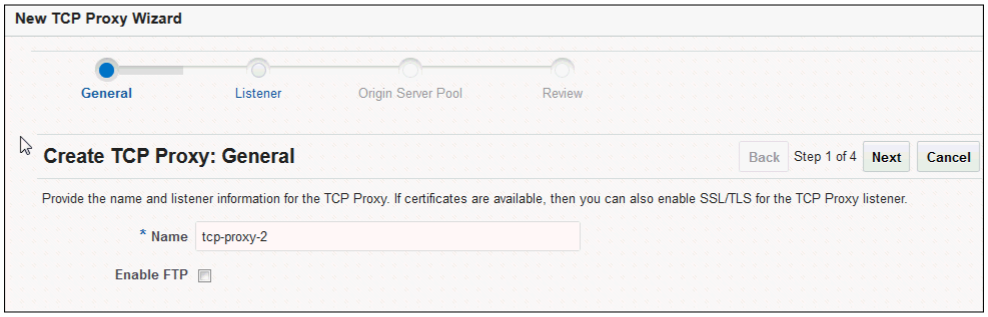
To create a TCP Proxy by using the Fusion Middleware Control, do the following:

1. Log in to Fusion Middleware Control, as described in Section 1.7.2, "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 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 starts.

Figure 8–1 New TCP Proxy Wizard



Create TCP Proxy: General

Provide the name and listener information for the TCP Proxy. If certificates are available, then you can also enable SSL/TLS for the TCP Proxy listener.

* Name: tcp-proxy-2

Enable FTP: ☐

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. For more information, see Section 10.1.2, "Configuring SSL/TLS for a 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 Section 3.3, "Activate 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-ftps` and `client-explicit-ftps` being 'false', 'true' and 'true' respectively. These properties can be modified later using `otd_setTcpProxyProperties`.

For more information about `otd_createTcpProxy`, see *WebLogic Scripting Tool Command Reference for Oracle Traffic Director*.

8.2 Viewing a List of TCP Proxies

You can view a list of TCP proxies by using either Fusion Middleware Control or the WLST.

Note: For information about invoking WLST, see Section 1.7.1, "Accessing WebLogic Scripting Tool."

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 Section 1.7.2, "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.

You can view the properties of a proxy in detail by clicking on its name.

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.

For more information about the `otd_listTcpProxies` and `otd_getTcpProxyProperties` commands, see *WebLogic Scripting Tool Command Reference for Oracle Traffic Director*.

8.3 Modifying a TCP Proxy

You can modify TCP proxies by using either Fusion Middleware Control or the WLST.

Note: For information about invoking WLST, see Section 1.7.1, "Accessing WebLogic Scripting Tool."

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 Section 1.7.2, "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 Section 9.1, "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-ftps' and 'client-explicit-ftps' as 'false', 'true' and 'true' respectively.

```
props = {}
props['configuration'] = 'foo'
props['tcp-proxy'] = 'bar'
props['client-explicit-ftps'] = 'true'
otd_setTcpProxyProperties(props)
```

For a list of the properties that you can set or change by using the `otd_setTcpProxyProperties` commands, see *WebLogic Scripting Tool Command Reference for Oracle Traffic Director*.

8.4 Deleting a TCP Proxy

You can delete TCP proxies by using either Fusion Middleware Control or the WLST.

Note: For information about invoking WLST, see Section 1.7.1, "Accessing WebLogic Scripting Tool."

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 Section 1.7.2, "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 Section 3.3, "Activate 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)
```

For more information about `otd_deleteTcpProxy`, see *WebLogic Scripting Tool Command Reference for Oracle Traffic Director*.

9 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:

9.1 Creating a Listener

You can create listeners by using either Fusion Middleware Control or the WLST.

Note: For information about using WLST, see Section 1.7.1, "Accessing WebLogic Scripting Tool."

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 Section 7.1, "Creating a Virtual Server."

- For HTTP listeners: 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.

  For more information about creating TCP proxies, see Section 8.1, "Creating a TCP Proxy."

Creating an HTTP Listener Using Fusion Middleware Control

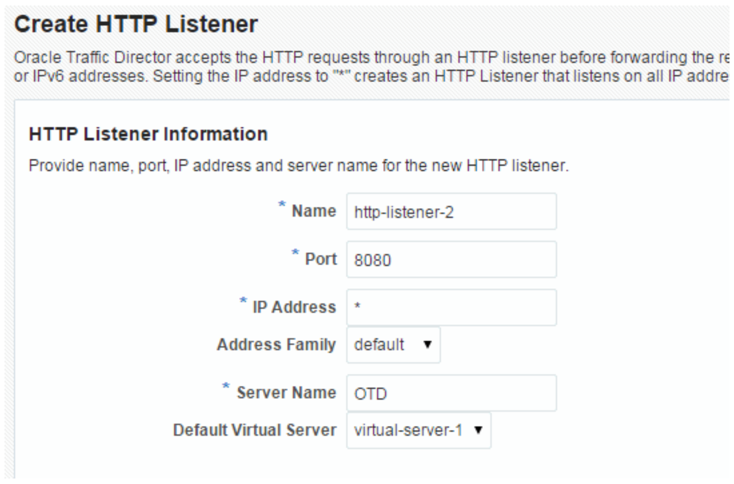
To create an HTTP listener by using the Fusion Middleware Control, do the following:

1. Log in to Fusion Middleware Control, as described in Section 1.7.2, "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 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 starts.

Figure 9–1 New HTTP Listener Wizard



Create HTTP Listener

Oracle Traffic Director accepts the HTTP requests through an HTTP listener before forwarding the re
or IPv6 addresses. Setting the IP address to "*" creates an HTTP Listener that listens on all IP addre

HTTP Listener Information

Provide name, port, IP address and server name for the new HTTP listener.

* Name	http-listener-2
* Port	8080
* IP Address	*
Address Family	default ▼
* Server Name	OTD
Default Virtual Server	virtual-server-1 ▼

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

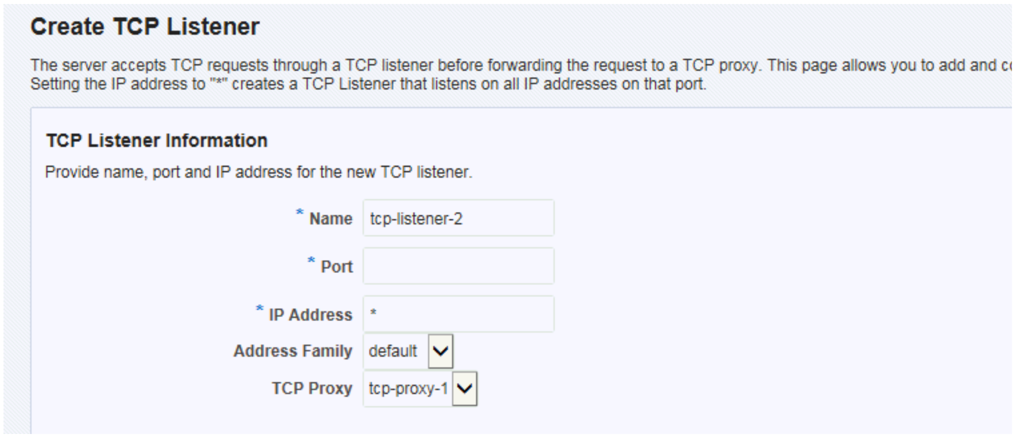
To create a TCP listener by using the Fusion Middleware Control, do the following:

1. Perform steps 1, 2, and 3 of *Creating an HTTP Listener Using 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 starts.

Figure 9–2 New TCP Listener Wizard



Create TCP Listener

The server accepts TCP requests through a TCP listener before forwarding the request to a TCP proxy. This page allows you to add and c
Setting the IP address to "*" creates a TCP Listener that listens on all IP addresses on that port.

TCP Listener Information

Provide name, port and IP address for the new TCP listener.

* Name	tcp-listener-2
* Port	
* IP Address	*
Address Family	default
TCP Proxy	tcp-proxy-1

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)
  ```

For more information about `otd_createHttpListener` and `otd_createTcpListener`, see *WebLogic Scripting Tool Command Reference for Oracle Traffic Director*.

9.2 Viewing a List of Listeners

You can view a list of HTTP or TCP listeners by using either Fusion Middleware Control or the WLST.

Note: For information about invoking WLST, see Section 1.7.1, "Accessing WebLogic Scripting Tool."

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 Section 1.7.2, "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.

  For more information about the `otd_listHttpListeners` and `gotd_getHttpListenerProperties` commands, see *WebLogic Scripting Tool Command Reference for Oracle Traffic Director*.

- 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 an TCP listener in detail by running the `otd_getTcpListenerProperties` command.

  For more information about the `otd_listTcpListeners` and `otd_getTcpListenerProperties` commands, see *WebLogic Scripting Tool Command Reference for Oracle Traffic Director*.

9.3 Modifying a Listener

You can modify listeners by using either Fusion Middleware Control or the WLST.

Note: For information about invoking WLST, see Section 1.7.1, "Accessing WebLogic Scripting Tool."

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 Section 1.7.2, "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. For more information, see Section 10.1.2, "Configuring SSL/TLS for a 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. For more information, see Section 15.4, "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 Section 3.3, "Activate 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)
  ```

For a list of the properties that you can set or change by using the `otd_setTcpListenerProperties` and `SslProperties` commands, see *WebLogic Scripting Tool Command Reference for Oracle Traffic Director*.

9.4 Deleting a Listener

You can delete HTTP or TCP listeners by using either Fusion Middleware Control or the WLST.

Note: For information about invoking WLST, see Section 1.7.1, "Accessing WebLogic Scripting Tool."

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 Section 1.7.2, "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 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: For HTTP listeners: If the HTTP listener is associated with any virtual servers, the prompt shows the names of those virtual servers.

8. To proceed with the deletion, click **Yes**.

   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)
  ```

For more information about `otd_deleteHttpListener` and `otd_deleteTcpListener`, see *WebLogic Scripting Tool Command Reference for Oracle Traffic Director*.

9.5 Configure OTD to listen on privileged ports

You can configure OTD to listen on privileged ports by the following steps.

1. To create listen sockets for previlaged ports, OTD bundles a binary named 'portbind'.

2. If OTD needs to listen at privileged ports, an admin needs to provide root ownership and setuid privileges to 'portbind'
   - chown root portbind.
   - chmod 4750 portbind.
3. It should be ensured that the server user has the required group ownerships for the above to work
4. The OTD watchdog process uses 'portbind' to create listen sockets that require privileged ports

9.6 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 will ensure that even when Oracle Traffic Director is loaded, it is still available to service status requests.

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.

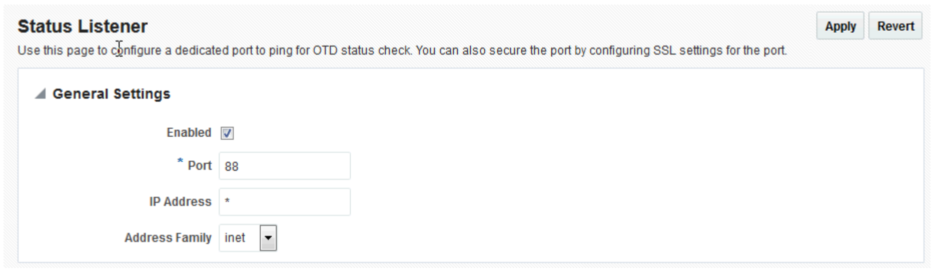
9.6.1 Configuring Status Listener using Fusion Middleware Control

To configure Status Listener by using the Fusion Middleware Control, do the following:

1. Log in to Fusion Middleware Control, as described in "Displaying 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.

Figure 9–3 General Settings Section



Status Listener

Use this page to configure a dedicated port to ping for OTD status check. You can also secure the port by configuring SSL settings for the port.

General Settings

Enabled ✓

* Port 88

IP Address *

Address Family inet

8. Click **Apply**.

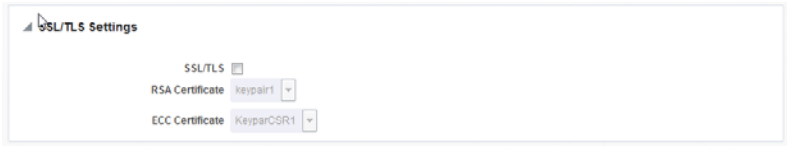
- The details of the Status Listener that you just configured are displayed on the **Status Listener** page.
- The **SSL/TLS Settings** and **Advanced Settings** sections are displayed.

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.

Figure 9–4 SSL/TLS Settings Section



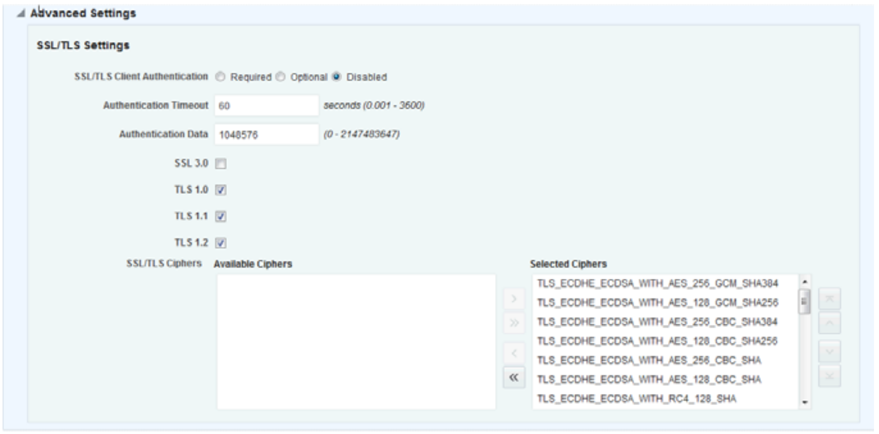
SSL/TLS Settings

SSL/TLS ☐
RSA Certificate: keypair1
ECC Certificate: KeyparCSR1

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.

Figure 9–5 Advanced Settings Section



Advanced Settings

SSL/TLS Settings

SSL/TLS Client Authentication: Required Optional [✓ Disabled]
Authentication Timeout: 60 seconds (0.001 - 3600)
Authentication Data: 1048576 (0 - 2147483647)
SSL 3.0 ☐
TLS 1.0 ✓
TLS 1.1 ✓
TLS 1.2 ✓
SSL/TLS Ciphers
Available Ciphers
Selected Ciphers
TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA
TLS_ECDHE_ECDSA_WITH_RC4_128_SHA

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.

9.6.2 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 reconfigure 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 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 the 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)
  ```

For information about the WLST commands mentioned in this section, see *WebLogic Scripting Tool Command Reference for Oracle Traffic Director*.

Part III

Advanced Administration

Part III contains the following chapters:

- Chapter 10, "Managing Security" describes how to secure access to the administration server; how to enable SSL/TLS for Oracle Traffic Director virtual servers, manage certificates; and how to manage certificates, PKCS#11 tokens, and certificate revocation lists.
- Chapter 11, "Managing Logs" provides an overview of the access and server logs; and describes how you can view logs, configure log preferences, and rotate logs.
- Chapter 12, "Monitoring Oracle Traffic Director Instances" describes the methods you can use to monitor Oracle Traffic Director instances.
- Chapter 13, "Event Notifications"describes events for which you can receive notifications.
- Chapter 14, "Configuring Oracle Traffic Director for High Availability" describes the high-availability features of Oracle Traffic Director. It describes how to configure Oracle Traffic Director instances in a failover group and set up Oracle Traffic Director to monitor the health of the origin servers in the back end.
- Chapter 15, "Tuning Oracle Traffic Director for Performance" describes the various parameters that you can tune to improve the performance of Oracle Traffic Director instances.
- Chapter 16, "Diagnosing and Troubleshooting Problems" provides information to help you understand and solve problems that you might encounter while using Oracle Traffic Director.
- Appendix A, "Metrics Tracked by Oracle Traffic Director" lists the names of the various metrics that Oracle Traffic Director tracks.
- Appendix B, "Web Application Firewall Examples and Use Cases" provides some basic information about how the web application firewall works.
- Appendix C, "Securing Oracle Traffic Director Deployment" provides information about the steps that you can take to secure your Oracle Traffic Director deployment.

10 Managing Security

This chapter describes how you can secure access to the Oracle Traffic Director administration server and enable SSL/TLS for Oracle Traffic Director virtual servers. It also describes how to configure client authentication and how you can use Oracle Traffic Director to secure access to origin servers.

This chapter contains the following sections:

Note: For information about some steps that you can take to secure Oracle Traffic Director in your environment, see Appendix C, "Securing Oracle Traffic Director Deployment."

10.1 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. The information in this section is aimed at readers who are familiar with the concepts of SSL/TLS, certificates, ciphers, and keys. For basic information about those concepts, see Section 10.1.7, "SSL/TLS Concepts."

This section contains the following subsections:

10.1.1 Overview of the SSL/TLS Configuration Process

To enable SSL/TLS for an Oracle Traffic Director instance, you must associate an RSA or ECC certificate, or both, with one more listeners of the instance. Additionally, you can associate an RSA or ECC certificate, or both, directly with virtual servers. The process of configuring SSL/TLS for Oracle Traffic Director instances involves the following steps:

1. Obtain the required certificates, which could be self-signed, issued by a third-party Certificate Authority (CA) like VeriSign or a certificate that you generated.

   For more information, see the following sections:

   - Section 10.3.1, "Generating a Keypair"
   - Section 10.3.2, "Obtaining a CA-Signed Certificate"
2. Install the certificates as described in Section 10.3.3, "Importing a Certificate."
3. Associate the certificates with the required HTTP or TCP listeners as described in Section 10.1.2, "Configuring SSL/TLS for a Listener."

   You can also associate certificates directly with virtual servers as described in Section 10.1.3, "Associating Certificates with Virtual Servers." For information about the logic that Oracle Traffic Director uses to select the certificate to be sent to a client during the SSL/TLS handshake, see Section 10.1.5, "Certificate-Selection Logic."
4. Configure ciphers supported for the HTTP or TCP listeners as described in Section 10.1.4, "Configuring SSL/TLS Ciphers for a Listener."

10.1.2 Configuring SSL/TLS for a Listener

You can configure a listener to receive HTTPS or TCP requests by using either Fusion Middleware Control or the WLST. Before you start, obtain the required certificates and install them as described in sections Section 10.3.1, "Generating a Keypair", Section 10.3.2, "Obtaining a CA-Signed Certificate", and Section 10.3.3, "Importing a Certificate".

Note:

- When you modify listeners, 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 Section 3.3, "Activate Configuration Changes."
- If you associate new certificates with a listener or remove previously associated certificates, for the changes to take effect, you must restart the instances. It is not sufficient to merely deploy the updated configuration.
- For information about invoking WLST, see Section 1.7.1, "Accessing WebLogic Scripting Tool."

Configuring SSL/TLS for a Listener Using Fusion Middleware Control

To configure SSL/TLS for an HTTP or TCP listener by using Fusion Middleware Control, do the following:

1. Log in to Fusion Middleware Control, as described in Section 1.7.2, "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 > Advanced Settings > SSL/TLS Settings.

   The SSL/TLS settings page i s 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:

    - Enable and disable settings for client authentication. For more information, see Section 10.6, "Configuring Client Authentication."
    - Enable and disable strict SNI host matching. For more information, see the Section 10.1.6, "About Strict SNI Host Matching." section.
    - Enable and disable the following TLS-specific features:
      - **Version Rollbacks**

        Select this check box if you want Oracle Traffic Director to detect and block attempts at rolling back the TLS version. For example, if the client requested TLS 1.0, but an attacker changed it to a lower version (say, SSL 3.0), Oracle Traffic Director can detect and block the rollback even if it supports the lower version.
      - **Session Ticket Extension**

        If enabled, TLS sessions can be resumed without storing the session state of each client on the server. Oracle Traffic Director encapsulates the session state of each client in a ticket and forwards the ticket to the client. The client can subsequently resume the TLS session by using the previously obtained session ticket.

- Enable and disable SSL and TLS ciphers. For more information, see Section 10.1.4, "Configuring SSL/TLS Ciphers for a Listener."

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 listener was saved, is displayed in the Console Messages pane.

12. 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_getTcpListernerSslProperties` 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
ssl3=true
tls10=true
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)
```

10.1.3 Associating Certificates with Virtual Servers

You can associate one RSA and one ECC certificate with each virtual server, by using either Fusion Middleware Control or the WLST. For information about the logic that Oracle Traffic Director uses to select the certificate to be sent to a client during the SSL/TLS handshake, see Section 10.1.5, "Certificate-Selection Logic."

Before you start, obtain the required certificates and install them as described in sections Section 10.3.1, "Generating a Keypair", Section 10.3.2, "Obtaining a CA-Signed Certificate", and Section 10.3.3, "Importing a Certificate".

Note:

- When you modify virtual servers, 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 Section 3.3, "Activate Configuration Changes."
- If you associate new certificates with a virtual server or remove previously associated certificates, for the changes to take effect, you must restart the instances. It is not sufficient to merely deploy the updated configuration.
- For information about invoking WLST, see Section 1.7.1, "Accessing WebLogic Scripting Tool."

Associating Certificates with Virtual Servers Using Fusion Middleware Control

To associate certificates with virtual servers by using Fusion Middleware Control, do the following:

1. Log in to Fusion Middleware Control, as described in Section 1.7.2, "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 associate certificates with virtual servers.
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 > Advanced Settings > SSL/TLS Settings.

   The SSL/TLS settings page i s 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.

    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 listener was saved, is displayed in the Console Messages pane.

12. Restart the instances of the configuration by clicking **Start Instances/Restart Instances** in the OTD Configuration pane.

Associating Certificates with Virtual Servers Using WLST

- To view the certificates that are currently associated with a virtual server, run the `otd_getVirtualServerProperties` command, as shown in the following example:

```
props = {}
props['configuration'] = 'foo'
props['virtual-server'] = 'bar'
otd_getVirtualServerProperties(props)
```

- 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'
props['http-listener'] = 'http-listener-1'
otd_setVirtualServerProperties(props)
```

For more information about the WLST commands mentioned in this section, see *WebLogic Scripting Tool Command Reference for Oracle Traffic Director*.

Note: Make sure 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.

10.1.4 Configuring SSL/TLS Ciphers for a Listener

During the SSL/TLS handshake, the client and server inform each other about the SSL and TLS ciphers that they support and then negotiate the cipher—typically, the strongest—that they will use for the SSL/TLS session. For basic conceptual information about ciphers, see *About Ciphers*.

You can configure the ciphers that Oracle Traffic Director supports for a listener by using either Fusion Middleware Control or the WLST.

Configuring Ciphers for a Listener Using Fusion Middleware Control

To configure the ciphers supported for an HTTP or TCP listener by using the Fusion Middleware Control, do the following:

1. Log in to Fusion Middleware Control, as described in Section 1.7.2, "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 ciphers.
5. Click the Traffic Director Configuration In the Common Tasks pane.
6. Select Administration > Listeners.

   The Listeners page is displayed. It shows a list of the HTTP/TCP listeners defined for the configuration

You can view the properties of a HTTP/TCP listener by clicking on its name.

7. Select the HTTP/TCP listener for which you want to enable and configure SSL/TLS.

   The Listener Settings page is displayed.

8. Select Settings > Advanced Settings > SSL/TLS Settings.

   The SSL/TLS settings page i s displayed.

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 for a Listener Using WLST

- To view the ciphers that are currently enabled for an HTTP or TCP listener, run the `otd_getHttpListenerSslProperties` or `otd_getTcpListernerSslProperties` command, as shown in the following example:

  ```
  props = {}
  props['configuration'] = 'foo'
  props['http-listener'] = 'http-listener-1'
  otd_getHttpListenerSslProperties(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` or `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 or `otd_getTcpListenerSslProperties` command.

For more information about the WLST commands mentioned in this section, see *WebLogic Scripting Tool Command Reference for Oracle Traffic Director*.

Cipher Suites Supported by Oracle Traffic Director

During the SSL/TLS handshake, Oracle Traffic Director and clients negotiate the cipher suites to be used. Table 10–1 lists the cipher suites supported in Oracle Traffic Director. You can view this list by running the `otd_getVirtualServerSslProperties` WLST, as described earlier in this section. The name of each cipher suite indicates the key-exchange algorithm, the hashing algorithm, and the encryption algorithm, as depicted in.

- **Protocols supported**
  - `TLS`: TLS 1.0, 1.1, 1.2
  - `SSL`: SSL 3
- **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_56`: 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

Table 10–1 Cipher Suites Supported in Oracle Traffic Director

Cipher Suite	FIPS 140 Compliant?
SSL_RSA_WITH_RC4_128_SHA	
SSL_RSA_WITH_3DES_EDE_CBC_SHA	
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA	Yes
TLS_ECDHE_RSA_WITH_RC4_128_SHA	
TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA	Yes
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA	Yes
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA	Yes
TLS_ECDHE_ECDSA_WITH_RC4_128_SHA	
TLS_ECDHE_ECDSA_WITH_3DES_EDE_CBC_SHA	Yes
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA	Yes
TLS_RSA_WITH_AES_128_CBC_SHA	Yes
TLS_RSA_WITH_AES_256_CBC_SHA	Yes
TLS_RSA_WITH_AES_128_CBC_SHA256	
TLS_RSA_WITH_AES_256_CBC_SHA256	
TLS_RSA_WITH_AES_128_GCM_SHA256	

Table 10–1 (Cont.) Cipher Suites Supported in Oracle Traffic Director

Cipher Suite	FIPS 140 Compliant?
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_256_CBC_SHA384	
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	

Source for FIPS 140-compliance information:
`http://www.mozilla.org/projects/security/pki/nss/ssl/fips-ssl-ciphersuites.html`

10.1.5 Certificate-Selection Logic

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/TCP listener
- A certificate associated with the default virtual server of the listener
- A certificate associated with the listener

Oracle Traffic Director uses the following logic to determine the certificate that should be sent to the client during the SSL/TLS handshake.

Table 10–2 Certificate-Selection Logic

Condition	Case A	Case B	Case C	Case D
Client sent SNI host extension	Yes	Yes	Yes	No
A *matching*[1] virtual server is found.	Yes	No	No	--
The matching virtual server has a certificate of a type—RSA or ECC— that matches the ciphers sent by the client.	Yes	--	--	--
The default virtual server of the listener has a certificate of a type—RSA or ECC— that matches the ciphers sent by the client.	--	Yes	--	--
The listener has a certificate of a type—RSA or ECC— that matches the ciphers sent by the client.	--	--	Yes	Yes
Certificate selected:	Certificate of the matching virtual server	Certificate of the default virtual server	Certificate of the listener	Certificate of the listener

[1] A *matching* virtual server is a virtual server that is bound to the listener and has a host pattern that matches the `Host:` header sent by the client.

10.1.6 About Strict SNI Host Matching

When a client sends an HTTPS request to an SSL/TLS-enabled Oracle Traffic Director instance, the server needs to send a certificate to the client to initiate the SSL/TLS handshake. If the host name in the request does not match the server name (common name, CN) in the certificate provided by the server, a warning message is displayed by the client to the user. To continue with the SSL/TLS handshake process, the user must then explicitly choose to trust the certificate.

If an Oracle Traffic Director instance contains multiple, name-based virtual servers configured with a single IP address and port combination, to determine the appropriate certificate that should be sent to the client, the server needs to know the value of the `Host` header in the HTTP request, which it cannot read until after the SSL/TLS connection is established.

An extension to the SSL and TLS protocols, called Server Name Indication (SNI), addresses this issue, by allowing clients to provide the requested host name during the SSL/TLS handshake in the SNI host extension. Oracle Traffic Director uses the host name in the SNI host extension to determine the virtual server certificate that it should send to the client. For information about associating certificates with virtual servers, see Section 10.1.3, "Associating Certificates with Virtual Servers."

Support for SNI is enabled by default for SSL/TLS-enabled HTTP listeners in Oracle Traffic Director. For stricter control, like if you want to prevent non-SNI clients from accessing name-based virtual servers, you should enable the Strict SNI Host Matching parameter.

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.

10.1.7 SSL/TLS Concepts

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 SSL v3 and Transport Layer Security (TLS) v1.

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.

SSL v3 and TLS v1 support a variety of ciphers. Clients and servers may support different *cipher suites* (sets of ciphers), depending on factors such as the protocol they support, the 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 will use to communicate.

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

10.2 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:

- Section 10.2.1, "About One-Way and Two-Way SSL/TLS"
- Section 10.2.2, "Configuring One-Way SSL/TLS Between Oracle Traffic Director and Origin Servers"
- Section 10.2.3, "Configuring Two-Way SSL/TLS Between Oracle Traffic Director and Origin Servers"
- Section 10.2.4, "Converting a non-SSL Oracle Traffic Director Instance to an SSL Oracle Traffic Director Instance"

10.2.1 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.

10.2.2 Configuring One-Way SSL/TLS Between Oracle Traffic Director and Origin Servers

To configure one-way SSL/TLS between Oracle Traffic Director and origin servers, you must export the origin servers' certificates in PKCS#12 format, install them in the certificate database of Oracle Traffic Director, and, optionally, configure Oracle Traffic Director to trust the certificates.

Note:

- The procedure described in this section is for a scenario where all of the servers in the origin-server pool use certificates issued by the same CA. In such a scenario, you can configure one-way SSL/TLS by importing just the root certificate of the CA that signed the certificates for the origin servers into the certificates database of Oracle Traffic Director.
- If the origin servers use self-signed certificates or certificates issued by different CAs, you should individually export and import each of the server certificates or the individual root certificates of the CAs that signed the server certificates.
- If the `WebLogic Server Plug-In Enabled` attribute, which can be accessed using the Weblogic Server Fusion Middleware Control, is set to `true` and when Oracle Traffic Director terminates an SSL connection, 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.

1. Export the root certificate of the CA that issued certificates to the origin servers into the PKCS#12 format.

   - **For Oracle WebLogic Server origin servers**:

     Use the `keytool` command available in Java SE 6.

     Syntax:

     ```
     > $JAVA_HOME/bin/keytool -exportcert -alias alias -file file -keystore
     keystore -storepass storepass -rfc
     ```

     `alias` is the nickname of the certificate to be exported, `file` is the name for the exported certificate, `keystore` is the name of the custom Oracle WebLogic Server identity store file, and `storepass` is the password for the specified keystore.

     Example:

     ```
     > $JAVA_HOME/bin/keytool -exportcert -alias wlsos1 -file wls_os_cert
      -keystore $DOMAIN_HOME/soa_domain/soa_keystore.jks -storepass stpass -rfc
     ```

     For more information about `keytool`, see the documentation at:

     `http://docs./javase/6/docs/technotes/tools/windows/keytool.html`

   - **For Oracle HTTP Server origin servers**:

     Use the `exportWalletObject` WebLogic Scripting Tool (WLST) command.

     Syntax:

     ```
     exportWalletObject(instName, compName, compType, walletName, password,
     type, path, DN)
     ```

     Example:

```
> exportWalletObject('inst1', 'ohs1', 'ohs','wallet1', 'password',
'Certificate', '/tmp','cn=soa.example.com')
```

This command exports the certificate with the DN `cn=soa.example.com` from the wallet `wallet1`, for Oracle HTTP Server instance `ohs1`. The trusted certificate is exported to the directory `/tmp`.

For more information about the `exportWalletObject` command, see the documentation at `exportWalletobject` in *Oracle Fusion Middleware Infrastructure Security WLST Command Reference*.

2. Install the root certificate, which you just exported, in the certificates database of Oracle Traffic Director by using the `importKeyStoreCertificate` WLST.

> **Note:** For information about installing a certificate using Fusion Middleware Control, see Section 10.3.3, "Importing a Certificate"

Syntax:

```
-----BEGIN CERTIFICATE-----
(Server SSL certificate)
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
(Intermediate certificate)
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
(Root certificate)
-----END CERTIFICATE-----
```

Example:

```
svc = getOpssService("KeyStoreService")

# generate a key pair with the proper DN
svc.generateKeyPair(appStripe='OTD', name='myconfig', password='',
alias='mycert', keypassword='', dn='CN=test., OU=Webtier, O=\'Oracle
Corporation\', ST=California, C=US', keysize='1024')
```

> **Note:** If the origin servers use self-signed certificates or certificates issued by different CAs, do the following instead of steps 1 and 2:
>
> 1. Export each server certificate, or each root certificate of the CAs that signed the server certificates, individually, by using the same commands used in step 1.
> 2. Install each certificate, which you exported in the previous step, in the certificates database of Oracle Traffic Director, by using the importKeyStoreCertificate WLST, as described in step 2 but with `--cert-type=server`.

3. If required, configure Oracle Traffic Director to verify the host name in the origin server's certificate by using the `otd_setRouteProperties` WLST.

Syntax:

```
otd_setRouteProperties(props)
```

Example:

```
props = {}
props['configuration'] = 'foo'
props['virtual-server'] = 'bar'
props['route'] = 'route-1'
props['websocket-idle-timeout'] = '1200'
otd_setRouteProperties(props)
```

To view a list of the virtual servers in a configuration and the routes defined for a virtual server, use the `otd_listVirtualServers` and `otd_listRoutes` WLST commands, respectively.

Note: If you choose to configure Oracle Traffic Director to validate the host name in the origin server's certificate during the SSL/TLS handshake, then you must do the following:

- Ensure that the server name (CN) in the origin server's certificate matches the origin server's host name as specified in the origin-server pool of the Oracle Traffic Director configuration. For more information about configuring origin-server pools, see Chapter 5, "Managing Origin-Server Pools."
- Ensure that dynamic discovery is disabled (default setting). For more information about dynamic discovery, see Section 5.5, "Configuring an Oracle WebLogic Server Cluster as an Origin-Server Pool."

Otherwise, when the origin server presents its certificate, Oracle Traffic Director cannot validate the host name in the certificate, and so the SSL/TLS handshake will fail.

10.2.3 Configuring Two-Way SSL/TLS Between Oracle Traffic Director and Origin Servers

To configure two-way SSL/TLS between Oracle Traffic Director and origin servers, do the following:

Note: For more information, about the WLST commands mentioned in this section, see *WebLogic Scripting Tool Command Reference for Oracle Traffic Director* or run the commands with the `--help` option.

---

1. Perform the procedure for configuring one-way SSL/TLS, as described in Section 10.2.2, "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 Section 10.3.2, "Obtaining a CA-Signed Certificate."
3. Install the CA-issued server certificate in the Oracle Traffic Director configuration, as described in Section 10.3.3, "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_enableRouteAuth` WLST.

   Syntax:

   ```
   otd_enableRouteAuth(props)
   ```

Example:

```
props = {}
props['configuration'] = 'foo'
props['virtual-server'] = 'bar'
props['route'] = 'route-1'
props['auth-user'] = 'baz'
props['auth-password'] = 'qux'
otd_enableRouteAuth(props)
```

To view a list of the virtual servers in a configuration and the routes defined for a virtual server, use the `otd_listVirtualServers` and `otd_listRoutes` WLST commands, respectively.

5. Deploy the updated configuration to the Oracle Traffic Director instances by using the `activate` command.

```
pullComponentChanges(<instance_name>)
```

6. Export the root certificate of the CA that signed the certificate for the Oracle Traffic Director instance, 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', or 'Certificate Chain', and `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')
```

7. Import the root certificate that you exported in the previous step into the trust keystore for the Oracle WebLogic Server origin servers (and the Oracle wallet for Oracle HTTP Server origin servers).

   - **For Oracle WebLogic Server origin servers**:

     Use the `keytool` command available in Java SE 8.

     Syntax:

     ```
     > $JAVA_HOME/bin/keytool -importcert -v -trustcacerts -alias alias
     -file cert_file -keystore keystore_file -storepass keystore_password
     -noprompt
     ```

`alias` is the nickname of the CA-issued root CA exported in the previous step, `file` is the name of the exported certificate file, `keystore` is the name of the custom Oracle WebLogic Server identity store file, and `storepass` is the password for the specified keystore.

Example:

```
> $JAVA_HOME/bin/keytool -importcert -v -trustcacerts -alias rootca1
 -file /tmp/rootca1.pem -keystore $DOMAIN_HOME/soa_domain/soa_keystore.jks
 -storepass stpass -noprompt
```

For more information about `keytool`, see the documentation at:

http://docs./javase/8/docs/technotes/tools/windows/keytool.html

- **For Oracle HTTP Server origin servers**:

  Use the `importWalletObject` WLST command.

  Syntax:

  ```
  importWalletObject(instName, compName, compType, walletName, password,
  type, filePath)
  ```

  Example:

  ```
  > importWalletObject('inst1', 'ohs1', 'ohs','wallet1', 'password',
  'TrustedCertificate','/tmp/rootca1.pem')
  ```

  For more information about the `importWalletObject` command, see the section on *importwalletobject* in *Oracle Fusion Middleware Infrastructure Security WLST Command Reference*.

8. 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
     ```

10.2.4 Converting a non-SSL Oracle Traffic Director Instance to an SSL Oracle Traffic Director Instance

To convert a non-SSL Oracle Traffic Director instance to an SSL Oracle Traffic Director instance, do the following:

1. Create a wallet, as described in “Creating a Wallet” in *Administering Oracle Fusion Middleware*.

2. Import required root, intermediate, and server certificates to a wallet as described "Importing a Certificate or Trusted Certificate" in *Administering Oracle Fusion Middleware*
3. Create a new HTTPS listener for port 4443 as described section Section 9.1, "Creating a Listener".
4. Restart the services as described in Section 4.3, "Starting, Stopping, and Restarting Oracle Traffic Director Instances".

10.3 Managing Certificates

This section contains the following topics:

- Section 10.3.1, "Generating a Keypair"
- Section 10.3.2, "Obtaining a CA-Signed Certificate"
- Section 10.3.3, "Importing a Certificate"
- Section 10.3.4, "Viewing a List of Certificates"
- Section 10.3.5, "Renewing a Server Certificate"
- Section 10.3.6, "Deleting a Certificate"

Note:

- The information in this section is aimed at readers who are familiar with the concepts of SSL, certificates, ciphers, and keys. For basic information about those concepts, see Section 10.1.7, "SSL/TLS Concepts."
- For information about invoking WLST, see Section 1.7.1, "Accessing WebLogic Scripting Tool."

10.3.1 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.

  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.
- The key size (for RSA) or curve (for ECC).

For RSA keys, you can specify 512, 1024, 2048, 4096 bits. Long keys provide better encryption, but Oracle Traffic Director would need more time to generate them.

For 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).

Generating a Keypair Using Fusion Middleware Control

To create a self-signed certificate by using Fusion Middleware Control, do the following:

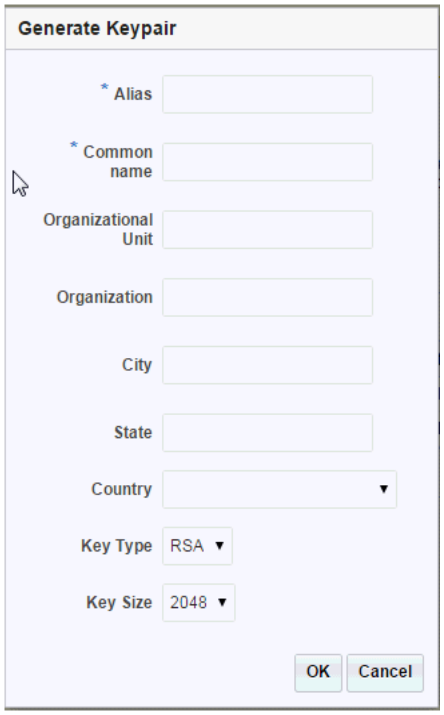
1. Log in to Fusion Middleware Control, as described in Section 1.7.2, "Displaying Fusion Middleware Control."
2. Click the WebLogic Domain button at the upper left corner of the page.
3. Select Administration > OTD Configurations.

   A list of the available configurations is displayed.
4. Select the configuration for which you want to create an self-signed 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 **Generate Keypair** button in the common task bar.

   The New Generate Keypair wizard opens.

Figure 10–1 New Generate Keypair Wizard



Key Type RSA

Key Size 2048

New Generate Keypair wizard

****

8. Click **OK**. The new certificate is displayed in the certificate list.
9. 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.

Generate 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., OU=Webtier, O=\'Oracle Corporation\', ST=California,
C=US', keysize='1024')
```

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

For more information, about `generateKeyPair`, see the *WebLogic Scripting Tool Command Reference for Oracle Traffic Director* or run the command with the `--help` option.

10.3.2 Obtaining a CA-Signed Certificate

To obtain a certificate signed by a Certificate Authority (CA), you should submit a *Certificate Signing Request* (*CSR*) to the CA, pay the prescribed fee if required, and wait for the CA to approve the request and grant the certificate.

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 CSR by using either Fusion Middleware Control or the WLST of Oracle Traffic Director.

Generating a CSR Using Fusion Middleware Control

To generate a CSR by using Fusion Middleware Control, do the following:

1. Log in to Fusion Middleware Control, as described in Section 1.7.2, "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**

You can now send the CSR with the required certificate-signing fee to a CA of your choice.

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.

For more information, about `exportKeyStoreCertificateRequest`, see the *WebLogic Scripting Tool Command Reference for Oracle Traffic Director* or run the command with the `--help` option.

After obtaining the CA-signed certificate in response to your CSR, you should import the certificate in the appropriate configuration, as described in Section 10.3.3, "Importing a Certificate."

10.3.3 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 Certificate Using Fusion Middleware Control

To Import a Certificate or trusted certificate by using Fusion Middleware Control, do the following:

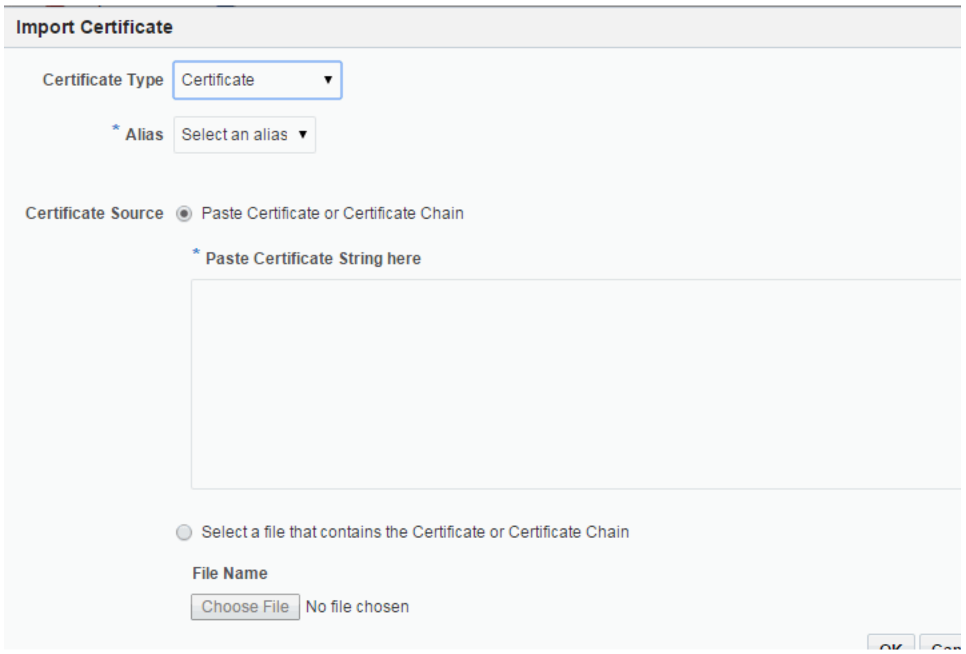
1. Log in to Fusion Middleware Control, as described in Section 1.7.2, "Displaying Fusion Middleware Control."
2. Click the WebLogic Domain button at the upper left corner of the page.
3. Select Administration > OTD Configurations.

   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, see Figure 10–2.

Figure 10–2 Import Certificate



Import Certificate

Certificate Type: Certificate

* Alias: Select an alias

Certificate Source: ● Paste Certificate or Certificate Chain

* Paste Certificate String here

○ Select a file that contains the Certificate or Certificate Chain

File Name

Import Certificate wizard

****

8. Choose **Certificate**, **Trusted Certificate**, or **Certificate Chain** from the **Certificate Type** drop-down.

Note:

- If you are importing a self-signed certificate, select **Trusted Certificate** from the **Certificate Type** drop-down.
- If you are importing a CA-signed certificate, select **Certificate Chain** from the **Certificate Type** drop-down.
- If you are importing a **Certificate Chain,** ensure that the Certificate chain obtained from the CA is created in the PKCS7 format.

9. Choose the **Alias** from the drop-down if you are importing a **Certificate** or a **Certificate Chain**. Specify the name of the **Alias** if you are importing a **Trusted Certificate**.
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 Certificate Using WLST

To Import a Certificate, run the `importKeyStoreCertificate` command, as shown in the following example:

```
svc.importKeyStoreCertificate(appStripe='OTD', name='myconfig', password='',
alias='ca-cert', keypassword='', type='Certificate',
filepath='/scratch/cacert.crt')
```

To Import a Certificate Chain, run the `importKeyStoreCertificate` command, as shown in the following example:

```
svc.importKeyStoreCertificate(appStripe='TEST_
STRIPE',name='sample',password='welcome1', alias='test_cert',
keypassword='welcome1',type='CertificateChain',filepath='/scratch/mwport/rpolavar/
CERTS/testCertChain.crt')
```

This command installs the server certificate with the nickname `soa-cert` in the configuration `soa`. To install a CA certificate, specify `ca` for the `--cert-type` option.

Note: If you are importing a **Certificate Chain,** ensure that the Certificate chain obtained from the CA is created in the PKCS7 format.

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

For more information about importKeyStoreCertificate, see the *WebLogic Scripting Tool Command Reference for Oracle Traffic Director* or run the command with the `--help` option.

10.3.4 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 Fusion Middleware Control, do the following:

1. Log in to Fusion Middleware Control, as described in Section 1.7.2, "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 it displays a list of certificates.

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')
```

> **Note:** If the pin is enabled for a token in the specified configuration, a prompt to enter the token pin is displayed when you run the `otd_listCertificates` and `getKeyStoreCertificates` commands.

For more information about the WLST commands mentioned in this section, see *WebLogic Scripting Tool Command Reference for Oracle Traffic Director* or run the command with the `--help` option.

10.3.5 Renewing a Server Certificate

To renew a certificate, do the following:

1. Log in to Fusion Middleware Control, as described in Section 1.7.2, "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 renew certificates.
4. In the navigation pane, expand **SSL** and select **Server Certificates**.

   The resulting page displays the installed server certificates.

   > **Note:** If the pin is enabled for a token in the selected configuration, the installed certificates are not displayed. Instead, a message to enter the token pins is displayed on the page.
   >
   > 1. Click **Cache Token Pin**.
   > 2. In the resulting dialog box, enter the pins for the tokens, and click **OK**.

5. Click the **Renew** button for the certificate that you want to renew.

   The Renew Server Certificate dialog box is displayed.
6. Specify the new validity period and click **Next**.
7. Click **Renew Certificate**.
8. Click **Close**.
   - A message is displayed in the Console Messages pane, confirming that the certificate has been renewed for the specified period.
   - The new expiry date for the certificate is displayed on the Server Certificates page.

10.3.6 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 in a configuration by using the Fusion Middleware Control, do the following:

1. Log in to Fusion Middleware Control, as described in Section 1.7.2, "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 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

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.

For more information about deleteKeyStoreEntry, see the *WebLogic Scripting Tool Command Reference for Oracle Traffic Director* or run the command with the `--help` option.

10.4 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.

This section contains the following topics:

- Section 10.4.1, "Installing and Deleting CRLs Manually"
- Section 10.4.2, "Update CRLs Automatically"

Note:

- The information in this section is aimed at readers who are familiar with the concepts of SSL, certificates, ciphers, and keys. For basic information about those concepts, see Section 10.1.7, "SSL/TLS Concepts."
- For information about invoking WLST, see Section 1.7.1, "Accessing WebLogic Scripting Tool."

10.4.1 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

To install a downloaded CRL by using the Fusion Middleware Control, do the following:

1. Log in to Fusion Middleware Control, as described in Section 1.7.2, "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 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.

For more information about the WLST commands mentioned in this section, see the *WebLogic Scripting Tool Command Reference for Oracle Traffic Director* or run the commands with the `--help` option.

10.4.2 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 by using the Fusion Middleware Control, do the following:

1. Log in to Fusion Middleware Control, as described in Section 1.7.2, "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. 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

   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.
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**.

    The New CRL Update Event dialog box is displayed.
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, do the following:

1. 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)
```

For more information about the WLST commands mentioned in this section, see the *WebLogic Scripting Tool Command Reference for Oracle Traffic Director* or run the commands with the `--help` option.

10.5 Managing Web Application Firewalls

A Web Application Firewall (WAF) is a filter or server plugin that applies a set of rules, called rule sets, to an HTTP request. Web Application Firewalls are useful for establishing an increased security layer in order to identify and prevent attacks. It acts as a firewall for applications hosted within the origin server. In addition, it enables administrators to inspect any part of an HTTP request, such as headers and body, and configure conditions to accept or reject the HTTP request based on the condition.

Several free and commercial versions of web application firewall modules are available for use. The web application firewall module for Oracle Traffic Director supports ModSecurity 2.8, which is an intrusion detection and prevention engine for web applications. The ModSecurity rule sets can be customized to shield applications from common attacks such as cross-site scripting (XSS) and SQL injection. Based on various criterion, such as HTTP headers, environment variables and CGI variables, ModSecurity filters and rejects incoming requests. For more information about ModSecurity, see `https://github.com/SpiderLabs/ModSecurity/wiki/Reference-Manual#wiki-Introduction`.

Among the many providers who have published different versions of the rule sets for ModSecurity, Oracle Traffic Director has been tested with the Open Web Application Security Project (OWASP), which is an open-source application security project, and is one of the most commonly used rule set providers. For more information, see `https://www.owasp.org/index.php/Category:OWASP_ModSecurity_Core_Rule_Set_Project`.

This section contains the following topics:

- Section 10.5.1, "Overview of Web Application Firewalls"
- Section 10.5.2, "Configuring Web Application Firewalls"
- Section 10.5.3, "Listing the Rule Set Files"
- Section 10.5.4, "Removing Rule Set Files"
- Section 10.5.5, "Supported Web Application Firewall Directives, Variables, Operators, Actions, Functions, Persistent Storages and Phases"

10.5.1 Overview of Web Application Firewalls

With Oracle Traffic Director, Web Application Firewalls can be enabled (or disabled) for each virtual server in your configuration. This in turn applies a set of rules, and acts as a firewall for the web applications deployed on the origin servers. For more information about origin servers and virtual servers, see Chapter 6, "Managing Origin Servers" and Chapter 7, "Managing Virtual Servers" respectively.

Oracle Traffic Director supports rule sets at both virtual server level and configuration level. Note that rules defined at the virtual server level will override rules defined at the configuration level. When deployed, these rules and the configuration changes are pushed to the instances, reconfiguring the instances. For more information about the Web Application Firewall works, see Appendix B, "Web Application Firewall Examples and Use Cases."

10.5.2 Configuring Web Application Firewalls

To configure Web Application Firewalls, you can either download an open source web application firewall rule sets or create your own rule sets. For example, download the ModSecurity Core Rule Set (CRS) from the OWASP repository, and unzip the rule sets to any folder. Oracle Traffic Director supports rules in the following directories:

- `base_rules`
- `optional_rules`
- `slr_rules`

Note: Web Application Firewall supports the ModSecurity 2.8 directives that are used by the configurations within the `base_rules`, `optional_rules` and `slr_rules` directories of OWASP ModSecurity Core Rule Set. However, it does not support Apache core config directives such as `<IfDefine...>` and `<Location...>`, and the ones supported by other Apache modules such as `RequestHeader`, `Header` and so on.

Note: Make sure that all the dependent files are installed before the Web Application Firewall rule file is installed. If not, the Web Application Firewall rule file fails to install.

After unzipping the above directories, the files in these directories can be edited and uploaded to the administration server. These rule set files are then pushed to the Oracle Traffic Director instances when deployed. For more information, see Section 10.5.2.1, "Enabling and Installing Web Application Firewall Rule Sets."

Note:

- Though the server can be configured to pick up the rule set files from a directory outside the `config` directory, rule file management will not be supported. When Oracle Traffic Director is configured for high availability, it is recommended that the web application firewall rule sets are placed within the `config` directory.
- Using unsupported directives, variables, operators, actions, phases, functions and storages can cause server startup errors. For example, installing the rule set file `modsecurity_crs_42_tight_security.conf` without removing the unsupported action `ver` can cause Oracle Traffic Director to display the following error message when you start the server:

  ```
  [ERROR:16] [OTD-20016] Syntax error on line 20 of
  /scratch/rgoutham/instance1/net-config1/config/ruleset/config1/
  modsecurity_crs_42_tight_security.conf:
  [ERROR:16] [OTD-20016] Error parsing actions: Unknown action:
  ver
  [ERROR:32] [OTD-20008] Failed to parse VS
  webapp-firewall-ruleset (ruleset/config1/*.conf)
  [ERROR:32] [OTD-10422] Failed to set configuration
  [ERROR:32] server initialization failed
  ```

  To avoid getting this error, modify the rule set file, and remove or comment out unsupported directives, variables, operators, actions, phases, functions and storages, and then start the server.

10.5.2.1 Enabling and Installing Web Application Firewall Rule Sets

You can enable and install web application firewall rule sets by using either Fusion Middleware Control or the WLST.

Note:

- When you enable and install a web application firewall rule set, you are, in effect, modifying a configuration. So for the new rule set to take effect in the Oracle Traffic Director instances, you should redeploy the configuration as described in Section 3.3, "Activate Configuration Changes."
- For information about invoking WLST, see Section 1.7.1, "Accessing WebLogic Scripting Tool."

Enabling and Installing Web Application Firewall Rule Sets Using Fusion Middleware Control

To configure web application firewall for a virtual server by using the Fusion Middleware Control, do the following:

1. Log in to Fusion Middleware Control, as described in Section 1.7.2, "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 web application firewall.
5. Click the Traffic Director Configuration In the Common Tasks pane.
6. Select Advanced Configuration > Web Application Firewall
7. The Web Application Firewall page is displayed.
   a. Click **Install Rule Set Files**.

      In the Install Rule Set Files dialog box, either browse to the folder where you unzipped the rule set files and select the rule set file or enter the full path to the location of the rule set file. To install multiple rule set files, install them one at a time.

      After you install one or more rule set files, the following text is added to the Rule Set Pattern field:

      ```
      ruleset/<virtual-server-id>/*.conf
      ```

      Note:

      - When you install rule set files at the configuration level, the rule set pattern appears as follows:

        ```
        ruleset/*.conf
        ```

      - If required, you can add custom rule set patterns. However, rule sets outside the `ruleset/<virtual-server-id>` directory (if at the virtual server level) or the `ruleset` directory (if at the configuration level) cannot be viewed or deleted using the Oracle Traffic Director Fusion Middleware Control or WLST. These rule sets will need to be managed manually.

Enabling Web Application Firewall Using WLST

To enable web application firewall using WLST, run the `otd_enableWebAppFirewall` command.

For example, the following command enables web application firewall for the virtual server `bar` in the configuration `foo`.

```
props = {}
props['configuration'] = 'foo'
props['virtual-server'] = 'bar'
otd_enableWebappFirewall(props)
```

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

For more information about `otd_enableWebAppFirewall`, see the *WebLogic Scripting Tool Command Reference for Oracle Traffic Director* or run the command with the `--help` option.

Installing Web Application Firewall Rule Sets Using WLST

To install web application firewall rule sets using WLST, run the `otd_installVirtualServerWebappFirewallRulesetFile` command.

For example, the following command installs the web application firewall rule set `modsecurity_crs_20_protocol_violations.conf` for the virtual server `bar` in the configuration `foo`.

```
props = {}
props['configuration'] = 'foo'
props['virtual-server'] = 'bar'
props['file-path'] = '/export/rulesets/baz.conf'
otd_installVirtualServerWebappFirewallRulesetFile(props)
```

To install web application firewall rule sets at the configuration level, run the `otd_installWebappFirewallRulesetFile` command. For example, the following command installs the web application firewall rule set `modsecurity_crs_50_outbound.conf` for the configuration `foo`.

```
props = {}
props['configuration'] = 'foo'
props['file-path'] = '/export/rulesets/baz.conf'
otd_installVirtualServerWebappFirewallRulesetFile(props)
```

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

For more information about `otd_installWebappFirewallRulesetFile` and `otd_installVirtualServerWebappFirewallRulesetFile`, see the *WebLogic Scripting Tool Command Reference for Oracle Traffic Director* or run the command with the `--help` option.

Note: You can use the `otd_setConfigurationProperties` and `otd_setVirtualServerProperties` commands to set the value of `otd_deleteWebappFirewallRulesetFile/otd_deleteVirtualServerWebappFirewallRulesetFile` property at the configuration level and virtual server level respectively. For more information, see the *WebLogic Scripting Tool Command Reference for Oracle Traffic Director*.

10.5.3 Listing the Rule Set Files

You can view the list of rule set files by using either Fusion Middleware Control or the WLST.

Note: For information about invoking WLST, see Section 1.7.1, "Accessing WebLogic Scripting Tool."

Viewing the List of Rule Set Files Using Fusion Middleware Control

To view the list of rule set files by using the Fusion Middleware Control, do the following:

1. Log in to Fusion Middleware Control, as described in Section 1.7.2, "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 rule set files.

5. Click the Traffic Director Configuration In the Common Tasks pane.
6. Select Advanced Configuration > Web Application Firewall
7. On the Web Application Firewall page, the Rule Set Files table lists the installed rule set files. To view the contents of these files either click and select individual rule files, or click the **Name** check box to select all the rules files.
8. Click **View**.

   The contents of each rule file is displayed in the Rule set file contents window.

Viewing the List of Rule Set Files Using WLST

While it is not possible to view the contents of individual rule set files using CLI, you can view the list of installed rule set files. To view the list of rule set files, run the `otd_listVirtualServerWebappFirewallRulesetFiles` command.

For example, the following command lists the installed web application firewall rule set files for the virtual server `bar` in the configuration `foo`:

```
props = {}
props['configuration'] = 'foo'
props['virtual-server'] = 'bar'
otd_listVirtualServerWebappFirewallRulesetFiles(props)
```

To view the list of web application firewall rule sets that are installed at the configuration level, run the `otd_listWebappFirewallRulesetFiles` command. For example, the following command lists the web application firewall rule sets that are installed at the configuration level for the configuration `foo`.

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

You can view the properties of a web application firewall by running the `otd_getWebappFirewallProperties` command.

For more information about the `otd_listWebappFirewallRulesetFiles, otd_listVirtualServerWebappFirewallRulesetFiles` and otd_getWebappFirewallProperties commands, see the *WebLogic Scripting Tool Command Reference for Oracle Traffic Director* or run the command with the `--help` option.

10.5.4 Removing Rule Set Files

You can remove rule set files by using either Fusion Middleware Control or the WLST.

Note: For information about invoking WLST, see Section 1.7.1, "Accessing WebLogic Scripting Tool."

Removing Rule Set Files Using Fusion Middleware Control

To remove rule set files for a particular virtual server:

1. Log in to Fusion Middleware Control, as described in Section 1.7.2, "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 rule set files.
5. Click the Traffic Director Configuration In the Common Tasks pane.
6. Select Advanced Configuration > Web Application Firewall
7. On the Web Application Firewall page, either click and select individual rule files or click the **Name** check box to select all the rule files.
8. Click the **Delete** button. At the confirmation prompt, click **OK**.

   A message is displayed in the Console Message pane confirming that the rule set files were deleted.

Removing Rule Set Files Using WLST

To remove rule set files for a particular virtual server, run the `otd_deleteVirtualServerWebappFirewallRulesetFile` command, as shown in the following example:

```
props = {}
props['configuration'] = 'foo'
props['virtual-server'] = 'bar'
props['ruleset-file'] = 'baz.conf'
otd_deleteVirtualServerWebappFirewallRulesetFile(props)
```

To remove rule set files for run the `otd_deleteWebappFirewallRulesetFile` command, as shown in the following example:

```
props = {}
props['configuration'] = 'foo'
props['ruleset-file'] = 'baz.conf'
otd_deleteVirtualServerWebappFirewallRulesetFile(props)
```

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

For more information about `otd_deleteWebappFirewallRulesetFile`, `otd_deleteVirtualServerWebappFirewallRulesetFile` see the *WebLogic Scripting Tool Command Reference for Oracle Traffic Director* or run the command with the `--help` option.

> **Note:** The `otd_disableWebAppFirewall` command can be used to disable a web application firewall. For more information, see the *WebLogic Scripting Tool Command Reference for Oracle Traffic Director*.

10.5.5 Supported Web Application Firewall Directives, Variables, Operators, Actions, Functions, Persistent Storages and Phases

Oracle Traffic Director supports various ModSecurity 2.8 directives, variables, operators, actions, functions, persistent Storages and phases.

Supported Web Application Firewall Directives

Oracle Traffic Director supports the following ModSecurity 2.8 directives. For more information and to see the full list of ModSecurity directives, including unsupported directives, see `https://github.com/SpiderLabs/ModSecurity/wiki/Reference-Manual#wiki-Configuration_Directives`.

```
SecAction
```

```
SecArgumentSeparator
SecAuditEngine
SecAuditLog
SecAuditLog2
SecAuditLogDirMode
SecAuditLogFileMode
SecAuditLogParts
SecAuditLogRelevantStatus
SecAuditLogStorageDir
SecAuditLogType
SecComponentSignature
SecContentInjection
SecCookieFormat
SecDataDir (see note below)
SecDebugLog
SecDefaultAction
SecDebugLogLevel
SecGeoLookupDb
SecInterceptOnError
SecMarker
SecPcreMatchLimit (see note below)
SecPcreMatchLimitRecursion (see note below)
SecRequestBodyAccess
SecRequestBodyInMemoryLimit (see note below)
SecRequestBodyNoFilesLimit (see note below)
SecRequestBodyLimitAction
SecResponseBodyAccess
SecResponseBodyLimit
SecResponseBodyLimitAction (see note below)
SecResponseBodyMimeType
SecResponseBodyMimeTypesClear
SecRule
SecRuleEngine (see note below)
SecRuleRemoveById
SecRuleRemoveByMsg
SecRuleRemoveByTag
SecRuleUpdateActionById
SecRuleUpdateTargetById
SecTmpDir
SecUnicodeMapFile (see note below)
SecUnicodeCodePage (see note below)
SecUploadDir
SecUploadFileLimit
SecUploadFileMode
SecUploadKeepFiles
SecWebAppId (see note below)
SecCollectionTimeout
```

Note:

- `SecWebAppId` can be specified within virtual server specific web application firewall configuration file to associate the application namespace to a particular virtual server.
- The directive `SecRequestBodyLimitAction` enables you to set action against requests that hit `SecRequestBodyNoFilesLimit`. However, the directive `SecRequestBodyLimit` is not supported by Oracle Traffic Director and hence, you cannot set action against this directive.
- Oracle Traffic Director does not support the directive `SecRequestBodyLimit`, which is used for configuring the maximum request body size that ModSecurity accepts for buffering. In place of this directive, the following options can be used:

  Option 1: Use the directives, `SecRequestBodyNoFilesLimit` and `SecRequestBodyLimitAction`. Example:

  ```
  SecRequestBodyNoFilesLimit 100
  SecRequestBodyLimitAction Reject
  ```

  Option 2: For Reject behavior, Oracle Traffic Director can be configured to check a request's `Content-Length` header in obj.conf. In addition, `max-unchunk-size` value can be set in server.xml.

  Similarly, for ProcessPartial behavior, `body-buffer-size` element in server.xml can be set to the desired limit. In this case, only the first part of the body that fits the limit is processed and the rest is passed through.
- If the directive `SecRuleEngine` is specified within the configuration file(s) specified by the `webapp-firewall-ruleset` element, then it will be ignored. However, this condition is not applicable if `SecRuleEngine` is set to `DetectionOnly` mode.
- The directive `SecRequestBodyInMemoryLimit` is ignored if the header `Content-Type` is set to `x-www-form-urlencoded`.
- The directives `SecDataDir`, `SecPcreMatchLimit`, `SecPcreMatchLimitRecursion`, `SecUnicodeCodePage`, and `SecUnicodeMapFile` can only be used at configuration level. The scope of these directives is considered to be Main. All the other directives can be used at both virtual server level and configuration level. The scope of these directives is considered to be Any. If a directive with Main scope is specified within the virtual server level configuration file, then an error will be logged and the server will fail to start.

Supported Web Application Firewall Variables

Oracle Traffic Director supports the following ModSecurity 2.8 variables. For more information and to see the full list of ModSecurity variables, including the unsupported variables, see https://github.com/SpiderLabs/ModSecurity/wiki/Reference-Manual#wiki-Variables.

```
ARGS
ARGS_COMBINED_SIZE
ARGS_GET
ARGS_GET_NAMES
ARGS_NAMES
ARGS_POST
ARGS_POST_NAMES
AUTH_TYPE
DURATION
ENV
FILES
FILES_COMBINED_SIZE
FILES_NAMES
FILES_SIZES
GEO
HIGHEST_SEVERITY
MATCHED_VAR
MATCHED_VARS
MATCHED_VAR_NAME
MATCHED_VARS_NAMES
MODSEC_BUILD
MULTIPART_BOUNDARY_QUOTED
MULTIPART_BOUNDARY_WHITESPACE
MULTIPART_DATA_AFTER
MULTIPART_DATA_BEFORE
MULTIPART_FILE_LIMIT_EXCEEDED
MULTIPART_HEADER_FOLDING
MULTIPART_INVALID_QUOTING
MULTIPART_INVALID_HEADER_FOLDING
MULTIPART_LF_LINE
MULTIPART_MISSING_SEMICOLON
MULTIPART_CRLF_LF_LINES
MULTIPART_STRICT_ERROR
MULTIPART_UNMATCHED_BOUNDARY
PERF_COMBINED
PERF_GC
PERF_LOGGING
PERF_PHASE1
PERF_PHASE2
PERF_PHASE3
PERF_PHASE4
PERF_PHASE5
PERF_SREAD
PERF_SWRITE
QUERY_STRING
REMOTE_ADDR
REMOTE_PORT
REMOTE_USER
REQBODY_ERROR
REQBODY_ERROR_MSG
REQBODY_PROCESSOR
REQBODY_PROCESSOR_ERROR
REQUEST_BASENAME
REQUEST_BODY (see note below)
REQUEST_BODY_LENGTH
REQUEST_COOKIES
REQUEST_COOKIES_NAMES
REQUEST_FILENAME
```

```
REQUEST_HEADERS (see note below)
REQUEST_HEADERS_NAMES
REQUEST_LINE
REQUEST_METHOD
REQUEST_PROTOCOL
REQUEST_URI
REQUEST_URI_RAW
RESPONSE_BODY
RESPONSE_CONTENT_LENGTH
RESPONSE_CONTENT_TYPE
RESPONSE_HEADERS
RESPONSE_HEADERS_NAMES
RESPONSE_PROTOCOL
RESPONSE_STATUS
RULE
SERVER_ADDR
SERVER_NAME
SERVER_PORT
SESSIONID
TIME
TIME_DAY
TIME_EPOCH
TIME_HOUR
TIME_MIN
TIME_MON
TIME_SEC
TIME_WDAY
TIME_YEAR
TX
UNIQUE_ID
URL_ENCODED_ERROR
USERID
WEBAPPID
WEBSERVER_ERROR_LOG (see note below)
XML
```

Note:

- The `REQUEST_BODY` variable, which holds raw request body, will contain the body content that is available after it passes through other filters.
- In open source ModSecurity, apache error log for each request/response can be collected and stored in the `WEBSERVER_ERROR_LOG` variable, and printed in the `auditlog` action. However, Oracle Traffic Director does not support this feature.
- As request headers with the same name are concatenated into a single one, the header count is always 1. Hence, `&REQUEST_HEADERS:<any header name>` will always return 1 in spite of how many same request headers were sent.

Supported Web Application Firewall Operators

Oracle Traffic Director supports the following ModSecurity 2.8 operators. For more information and to see the full list of ModSecurity operators, including unsupported operators `https://github.com/SpiderLabs/ModSecurity/wiki/Reference-Manual#wiki-Operators`.

```
beginsWith
contains
containsWord
endsWith
eq
ge
geoLookup
gt
inspectFile
ipMatch
le
lt
pm
pmf
pmFromFile
rbl (see note below)
rx
streq
strmatch
validateByteRange
validateDTD
validateSchema
validateUrlEncoding
validateUtf8Encoding
verifyCC
verifyCPF
verifySSN
within
```

Note: ModSecurity 2.8 does not support the directive `SecHttpBlKey`. Hence use of Project Honey Pot (dnsbl.httpbl.cong) as RBL, which requires `SecHttpBlKey`, is not supported.

Supported Web Application Firewall Actions

Oracle Traffic Director supports the following ModSecurity 2.8 actions. For more information and to see the full list of ModSecurity actions, including the unsupported actions, see `https://github.com/SpiderLabs/ModSecurity/wiki/Reference-Manual#wiki-Actions`.

```
allow
append
auditlog (see note below)
block
capture
chain
ctl
deny (see note below)
deprecatevar
drop (see note below)
exec
expirevar
id
initcol
log
logdata
msg
multiMatch
```

```
noauditlog
nolog
pass
pause
phase
prepend
redirect
rev
sanitiseArg
sanitiseMatched
sanitiseMatchedBytes
sanitiseRequestHeader
sanitiseResponseHeader
severity
setuid
setsid
setenv
setvar
skip
skipAfter
status
t
tag
xmlns
```

Note:

- In open source ModSecurity, apache error log for each request/response can be collected and stored in the `WEBSERVER_ERROR_LOG` variable, and printed in the `auditlog` action. However, Oracle Traffic Director does not support this feature.
- Actions that change HTTP response status, such as deny, will not successfully change the response status when it is invoked in phase 4. In such a scenario, the following error message is logged in the server log:

  ```
  " ModSecurity: Access denied with code 403 (phase 4)."
  ```

- When `drop` action is invoked in phase 4, Oracle Traffic Director will send out HTTP headers to the client and then drop the connection.
- When `deny` action is invoked in phase 4, Oracle Traffic Director strips the response body, instead of sending 403 response status. This might cause the following warning to appear in the server log:

  ```
  Response content length mismatch (0 bytes with a content length
  of <original content length>)
  ```

Supported Web Application Firewall Transformation Functions

Oracle Traffic Director supports the following ModSecurity 2.8 transformation functions. For more information and to see the full list of ModSecurity transformation functions, see `https://github.com/SpiderLabs/ModSecurity/wiki/Reference-Manual#wiki-Transformation_functions`.

```
base64Decode
```

```
sqlHexDecode
base64DecodeExt
base64Encode
cmdLine
compressWhitespace
cssDecode
escapeSeqDecode
hexDecode
hexEncode
htmlEntityDecode
jsDecode
length
lowercase
none
normalisePath
normalisePathWin
parityEven7bit
parityOdd7bit
parityZero7bit
removeNulls
removeWhitespace
replaceComments
removeCommentsChar
removeComments
replaceNulls
urlDecode
urlDecodeUni
urlEncode
sha1
trimLeft
trimRight
trim
```

Supported Web Application Firewall Persistent Storages

Oracle Traffic Director supports the following ModSecurity 2.8 persistent storages. For more information and to see the full list of ModSecurity persistent storages, see https://github.com/SpiderLabs/ModSecurity/wiki/Reference-Manual#wiki-Persistant_Storage.

```
GLOBAL
IP
RESOURCE
SESSION
USER
```

Supported Web Application Firewall Phases

Oracle Traffic Director supports the following ModSecurity 2.8 phases. For more information and to see the full list of ModSecurity phases, see https://github.com/SpiderLabs/ModSecurity/wiki/Reference-Manual#wiki-Processing_Phases.

```
Phase:1 - Request headers stage
Phase:2 - Request body stage
Phase:3 - Response headers stage
Phase:4 - Response body stage
Phase:5 - Logging
```

Note:

- Actions that change HTTP response status, such as deny, will not successfully change the response status when it is invoked in phase 4.
- When `drop` action is invoked in phase 4, Oracle Traffic Director will send out HTTP headers to the client and then drop the connection.
- When `deny` action is invoked in phase 4, Oracle Traffic Director strips the response body, instead of sending 403 response status. This might cause the following warning to appear in the server log:

  ```
  Response content length mismatch (0 bytes with a content length
  of <original content length>)
  ```

10.6 Configuring Client Authentication

Client authentication is the verification of a client by the Oracle Traffic Director virtual server or TCP proxy, based on the certificate that the client provides.

Client authentication is not enabled by default. You can configure the Oracle Traffic Director listeners to require clients to provide a certificate, by using either Fusion Middleware Control or the WLST.

Note: For information about invoking WLST, see Section 1.7.1, "Accessing WebLogic Scripting Tool."

Configuring Client Authentication Using Fusion Middleware Control

To enable client authentication for a listener by using Fusion Middleware Control, do the following:

1. Log in to Fusion Middleware Control, as described in Section 1.7.2, "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 client authentication for listeners.
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 > Advanced Settings > SSL/TLS Settings.
9. 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.

10. Specify the **Authentication Timeout** and **Maximum Authentication Data** parameters.

    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 listener was saved, is displayed in the Console Messages pane.

Configuring Client Authentication Using WLST

To enable client authentication for an HTTP or TCP listener, run the otd_setVirtualServerSslProperties command for virtual server.

```
props = {}
props['configuration'] = 'foo'
props['virtual-server'] = 'bar'
props['client-auth'] = 'false'
otd_setVirtualServerSslProperties(props)
```

For more information about the WLST commands mentioned in this section, see the *WebLogic Scripting Tool Command Reference for Oracle Traffic Director* or run the commands with the `--help` option.

10.7 Preventing Denial-of-Service Attacks

A denial-of-device (DoS) attack is an attempt by a malicious user to prevent legitimate users from accessing a service, by sending continuous requests to the server.

To prevent DoS attacks, you can configure Oracle Traffic Director virtual servers to reject requests when the frequency of requests or the number of concurrent connections exceeds a specified limit. For more granular control over requests, you can define several request limits and configure each limit to be applied to requests that match specified URL patterns and query string patterns, request headers that match specified values, and so on.

This section contains the following subsections:

- Section 10.7.1, "Request Limiting Parameters"
- Section 10.7.2, "Configuring Request Limits for a Virtual Server"

10.7.1 Request Limiting Parameters

You can specify multiple request limits for a virtual server. For each request limit, you can configure several parameters:

- You can make each request limit applicable to requests fulfilling a specified condition that you specify using expressions such as the following:

  ```
  $path = "*.jsp"
  $url = "/images/*"
  $ip =~ "^130\.35\.46\..*"
  ```

You can use any variable or a combinations of variables to specify the condition for a limit. For more information about building expressions for request limit conditions, see "Using Variables, Expressions, and String Interpolation" in the *Configuration File Reference for Oracle Traffic Director* .

- In each request limit, you can specify the number of concurrent requests (`max-connections`) and the average number of requests per second (`max-rps`).

  For example, if you specify a limit (say, `max-rps`=20), Oracle Traffic Director continuously tracks the request rate by recalculating it at a compute interval that you specify (default: 30 seconds), based on the number of requests received during that interval. When the specified request limit is reached, Oracle Traffic Director rejects all subsequent requests.

- You can also specify an optional attribute that Oracle Traffic Director should monitor when applying request limits. Oracle Traffic Director uses separate counters to track the request statistics for each monitored attribute.

  For example, to specify that Oracle Traffic Director should track the request rate separately for each client IP, you can specify the variable `$ip` as the monitor attribute. When the request rate exceeds the specified limit for any client, subsequent requests from *that* client are rejected, but requests from other clients continue to be served.

  You can also combine variables when specifying the attribute to be monitored. For example, to limit requests from clients that request the same URIs too frequently, you can specify `$ip:$uri` as the attribute to be monitored. When the request rate from any client for any single URI exceeds the limit, further requests to the same URI from that client are rejected, but requests from that client to other URIs, as well as requests from other clients to any URI continue to be served.

- For requests that Oracle Traffic Director rejects, it returns the HTTP response code that you specify. The default status code is `503 (service unavailable).`

- After a specified limit—`max-connections` or `max-rps`—is reached, Oracle Traffic Director continues to reject all subsequent requests until a specified *continue condition* is satisfied. You can specify one of the following continue conditions:

  – **Threshold** (default): Service resumes when the request rate falls below the specified limit.

  – **Silence**: Service resumes when the incoming request falls to zero for an entire interval.

10.7.2 Configuring Request Limits for a Virtual Server

You can configure request limits for a virtual server by using either Fusion Middleware Control or the WLST.

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 Section 3.3, "Activate Configuration Changes."
- For information about invoking WLST, see Section 1.7.1, "Accessing WebLogic Scripting Tool."

Configuring Request Limits Using Fusion Middleware Control

To configure request limits by using the Fusion Middleware Control, do the following:

1. Log in to Fusion Middleware Control, as described in Section 1.7.2, "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 request limits.
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. In the navigation pane, expand **Virtual Servers**, expand the name of the virtual server for which you want to configure request limits, and select **Request Limits**.

   The Request Limits page is displayed. It lists the request limits that are currently defined for the virtual server.

   Creating a Request Limit

   a. Click **New Request Limit**.

      The New Request Limit dialog box is displayed.

      In the **Name** field, enter a name for the new request limit.

      In the **Connections** field, specify the maximum number of concurrent connections to the virtual server.

      In the **Requests Per Second** field, specify the maximum number of requests that the virtual server can accept per second.

      Note: You must specify at least one of the limits—maximum number of connections or maximum number of requests per second.

      In the **Monitor Attribute** field, specify the attribute in the request header, which the virtual server should monitor for applying the request limit. If you do not specify this parameter, the request limit is applied to all requests.

      In the **Applies To** field, select the default one, the request limit is applied to all requests.

   b. Click **OK** and then new **Create Request Limit**.is created.

      The request limit that you just created is displayed on the Request Limits page.

   Editing a Request Limit

   To change the settings of a request limit, do the following:

   a. Click the **Name** of the request limit.

      The Editing Request Limit page 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.

While editing a request limit, in addition to changing the parameters that you specified while creating the request limit, you can set and change the `requests-per-second` compute interval, and the HTTP status code that the virtual server should return for requests that it rejects when the specified limits are reached. In addition, you can edit the condition that you have set by clicking **Edit**, which allows you to edit the condition either manually or using the condition builder. You can also delete old expressions and add new ones.

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.

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 Request Limit

To delete a request limit, click the **Delete** button. At the confirmation prompt, click **OK**.

Configuring Request Limits Using WLST

- To create a request limit, run the `otd_createRequestLimit` command.

  Examples:

  – The following command creates a request limit named request-limit-1 in the virtual server bar of the configuration `foo`.

  ```
  props = {}
  props['configuration'] = 'foo'
  props['virtual-server'] = 'bar'
  props['request-limit'] = 'request-limit-1'
  props['max-connections'] = '2048'
  otd_createRequestLimit(props)
  ```

  Note that the value of the `--condition` option 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 request limits defined for a virtual server, run the `otd_listRequestLimits` command, as shown in the following example:

  ```
  props = {}
  props['configuration'] = 'foo'
  props['virtual-server'] = 'bar'
  otd_listRequestLimits(props)
  ```

- To view the properties of a request limit, run the `otd_getRequestLimitProperties` command, as shown in the following example:

  ```
  props = {}
  props['configuration'] = 'foo'
  props['virtual-server'] = 'bar'
  props['request-limit'] = 'request-limit-1'
  props['event-notification-interval'] = '60'
  otd_getRequestLimitProperties(props)
  ```

- To change the properties of a request limit, run the `otd_setRequestLimitProperties` command.

  For example, the following command changes the request-per-second compute interval of the request limit request-limit-1 in the virtual server `bar` of the configuration `foo`.

  ```
  props = {}
  props['configuration'] = 'foo'
  props['virtual-server'] = 'bar'
  props['request-limit'] = 'request-limit-1'
  props['max-connections'] = '1024'
  otd_setRequestLimitProperties(props)
  ```

- To delete a request limit, run the `otd_deleteRequestLimit` command, as shown in the following example:

  ```
  props = {}
  props['configuration'] = 'foo'
  props['virtual-server'] = 'bar'
  props['request-limit'] = 'request-limit-1'
  otd_deleteRequestLimit(props)
  ```

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

For more information about the WLST commands mentioned in this section, see *WebLogic Scripting Tool Command Reference for Oracle Traffic Director*.

10.8 Configure SSL Pass through on OTD

Oracle Traffic Director (OTD) is a software based application delivery controller for Oracle engineered systems and includes the ability to terminate SSL, load-balance, limit, and throttle the incoming requests while front ending Oracle Fusion Middleware and Business Application deployments like SOA, ATG (E-Commerce), PeopleSoft, E-Business Suite etc.

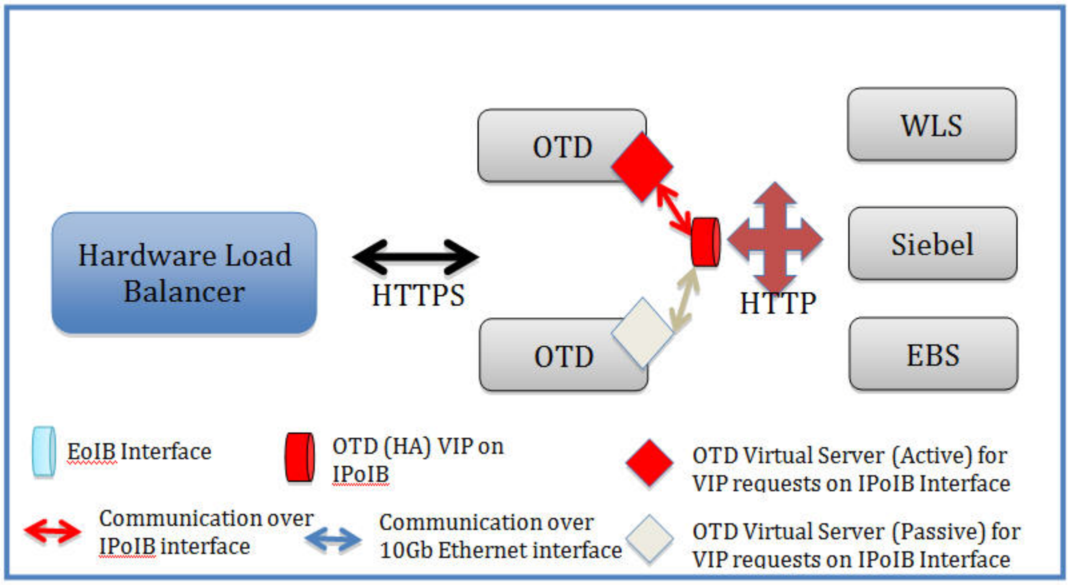
Oracle Traffic Director (OTD) offers the following key capabilities:

- "Full fledged Layer-7 software load balancer with advanced load balancing algorithms and traffic management capabilities based on either HTTP headers or HTTP entity data.
- A Layer-5 (Single connection-TCP) load balancer to front-end TCP traffic like WebLogic RMI t3, LDAP protocols etc.
- ModSecurity based Web Application Firewall (WAF) capability to protect back-end application from common security vulnerabilities like SQL injection, command injection vulnerabilities etc.
- High Availability (Active-Passive / Active-Active) fail-over capability within load balancer tier.

In addition, OTD offers the ability to offload SSL cipher processing, while handling HTTP(s)/SSL requests, to the underlying Intel processor for better throughput. However, in production systems, customers terminate SSL at an external hardware load balancer. This document will go through the list of configuration steps necessary to handle this use case.

10.8.1 Configure OTD to pass through SSL information from an external (hardware) Load Balancer

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.


Hardware Load Balancer
HTTPS
OTD
OTD
HTTP
WLS
Siebel
EBS
EoIB Interface
OTD (HA) VIP on IPoIB
OTD Virtual Server (Active) for VIP requests on IPoIB Interface
Communication over IPoIB interface
Communication over 10Gb Ethernet interface
OTD Virtual Server (Passive) for VIP requests on IPoIB Interface

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 (OTD) / reverse proxy solution.

1. Configure Hardware Load Balancer to send additional header to OTD
   - "If you terminate SSL within OTD, you will need to configure your hardware load balancer to send header - 'WL-Proxy-SSL:true'- to OTD. Refer to section -Configure F5-BigIP to send specific header to OTD- for more information.
   - "If you are terminating SSL within your hardware load balancer and also using OTD/OHS 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. This step is similar to inserting WL-Proxy-SSL header as documented within Configure F5-BigIP to send specific header to OTD- section.
2. Enable WLS Plug-In Enabled configuration settings within WebLogic Server:
   - If the origin server is neither a WebLogic Server nor using the underlying WebLogic Server container, then you can skip this step!
   - For more information, refer to section -Configure WebLogic to receive SSL information from Web Tier / Traffic Director-section.
3. Configure OTD 11g (using Admin Console UI) to pass through SSL:

- Login to OTD Admin Console with the appropriate credentials.
- Expand the Virtual Server and select the corresponding / relevant 'Route' screen.
- Update Route (Virtual Server -> Routes -> Default -> Advanced) and disable OTD from rewriting any 'Location or 'Content-Location' headers. This is important when SSL termination is happening at an external load balancer.
- Change the 'Rewrite-headers' to be empty.
- Under 'Advanced Settings' for the Route, uncheck all SSL related headers sent to the Origin Server (as shown below) and save and deploy this configuration.
- Save the changes and click on the 'Deploy Changes' button.

At the end of the above step, Oracle Traffic Director now does NOT rewrite any headers (typically this happens when the application tier redirects the request to another location) and can pass through the external hardware load balancer provided SSL information, via the incoming request headers, within the incoming requests.

10.8.2 Configure WebLogic to receive SSL information from Web Tier / Traffic Director

When you front-end WebLogic application server with Oracle's Web Tier solutions such as WebLogic Proxy Plug-In, Oracle HTTP Server, or with Oracle Traffic Director, then you will need to explicitly turn on 'WebLogic Plug-In Enabled' settings within WebLogic console. This flag can be tuned on either within a specific WebLogic Managed Server or at WebLogic cluster level. We recommend turning on this flag at WebLogic cluster level.

This ensures that WebLogic is able to appropriately rewrite the headers when SSL termination happens within Traffic Director / Web Tier.

Open http://<wls-admin-hostname>:7001/console and login with weblogic user/password.

(If you have Weblogic cluster enabled, then we recommend doing this at cluster level).

(If you do not have enabled Weblogic cluster, then you can enable this setting within WLS Managed Server). Within 'Configuration->General' settings tab, expand 'Advanced Settings' and scroll down to check "Weblogic Plugin Enabled" to 'Yes' and click "Save".

10.8.3 Configure F5-BigIP to send specific header to OTD

Oracle Traffic Director can offload cipher processing to take advantage of the underlying hardware processors to achieve overall better SSL performance compared to generic reverse proxy solutions.

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 below step covers how to configure F5-BigIp to send this header (Source: F5 Big IP product documentation).

To create a new HTTP profile for SSL:

1. On the Main tab, expand Local Traffic, and then click Profiles. The HTTP Profiles screen opens.
2. In the upper right portion of the screen, click the Create button. The New HTTP Profile screen opens.

3. In the Name box, type a name for this profile. In our example, we type bea-ssl.
4. 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.
5. In the Request Header Insert row, check the Custom box. In the box, type: WL-Proxy-SSL:true.
6. In the Redirect Rewrite row, check the Custom box. From the Redirect Rewrite list, select Match.
7. Modify any of the other settings as applicable for your network. In our example, we leave the settings at their default levels.
8. Click the Finish button.

11 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:

11.1 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. The instance logs are enabled by default and initialized when the instance is started for the first time. In addition to the instance logs, 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.

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 Section 11.3, "Configuring Log Preferences."

11.1.1 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. For more information, see Section 11.3, "Configuring Log Preferences."

11.1.2 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. For more information, see Section 11.3, "Configuring Log Preferences."

Table 11–1 lists the log levels that you can specify for the server log.

Table 11–1 Server Log Levels

Log Level	Description
`INCIDENT_ERROR:1`	A serious problem caused by unknown reasons. You should contact Oracle for support.

`ERROR:1`
`ERROR:16`
`ERROR:32`
	A serious problem that requires your immediate attention.

`WARNING:1`	A potential problem that you should review.
`NOTIFICATION:1` (default)	A major lifecycle event, such as a server being started or restarted.

`TRACE:1`
`TRACE:16`
`TRACE:32`
	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.

11.2 Viewing Logs

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` WLST 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 Section 15.9, "Configuring Access-Log Buffer Settings")).
- For information about using WLST, see Section 1.7.1, "Accessing WebLogic Scripting Tool."

Viewing Logs Using Fusion Middleware Control

To view log data for a node, an instance, or virtual server within an instance by using the Fusion Middleware Control, do the following:

1. Log in to Fusion Middleware Control, as described in Section 1.7.2, "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")
  ```

For more information about `displayLogs`, see *WebLogic Scripting Tool Command Reference for Oracle Traffic Director*.

11.3 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.

Note: For information about invoking WLST, see Section 1.7.1, "Accessing WebLogic Scripting Tool."

Configuring Log Preferences Using Fusion Middleware Control

To configure log preferences for a configuration or a virtual server by using the Fusion Middleware Control, do the following:

1. Log in to Fusion Middleware Control, as described in Section 1.7.2, "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**.
    - A message, confirming that the updated configuration was saved, is displayed in the Console Messages pane.

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)
  ```

For more information about the WLST commands mentioned in this section, see *WebLogic Scripting Tool Command Reference for Oracle Traffic Director*.

11.4 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, 2011 would be `server-201108252300.log`.

After log rotation, the server and access logs are re-initialized.

For information about how to rotate logs, see Section 11.5, "Rotating Logs Manually" and Section 11.6, "Configuring Oracle Traffic Director to Rotate Logs Automatically."

Note: Rotate Access Log event will also rotate TCP access logs.

11.5 Rotating Logs Manually

You can rotate the server and access logs of Oracle Traffic Director instances manually by using either Fusion Middleware Control or the WLST. 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.

Note: For information about invoking WLST, see Section 1.7.1, "Accessing WebLogic Scripting Tool."

Rotating Logs Manually Using Fusion Middleware Control

To rotate logs by using the Fusion Middleware Control, do the following:

1. Log in to Fusion Middleware Control, as described in Section 1.7.2, "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 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
   a. 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-yyyymmddhhmm.log
      $ /usr/bin/gzip server-yyyymmddhhmm.log
      ```

   b. 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:

   a. In the navigation pane, select **Instances**.

      The Instances page is displayed.

   b. 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 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-yyyymmddhhmm.log
$ /usr/bin/gzip server-yyyymmddhhmm.log
```

For more information about `otd_rotateLog` and `otd_setLogProperties`, see *WebLogic Scripting Tool Command Reference for Oracle Traffic Director*.

11.6 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.

Note: For information about invoking WLST, see Section 1.7.1, "Accessing WebLogic Scripting Tool."

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 Section 1.7.2, "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 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-yyyymmddhhmm.log
   $ /usr/bin/gzip server-yyyymmddhhmm.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)
```

For more information about `otd_createEvent`, see *WebLogic Scripting Tool Command Reference for Oracle Traffic Director*.

12

Monitoring Oracle Traffic Director Instances

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 Appendix A, "Metrics Tracked by Oracle Traffic Director."

This chapter describes the monitoring capabilities of Oracle Traffic Director. It contains the following sections:

12.1 Methods for Monitoring Oracle Traffic Director Instances

Table 12–1 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 12–1 Methods for Monitoring Oracle Traffic Director Instances

Monitoring Method	Requirements	Advantages
WLST ■ View runtime statistics for various subsystems for an instance: To view in plain-text format: `otd_getPerfDump` To view in XML format: `otd_getStatsXml` ■ Display runtime statistics about all instances, or a specific instance, from metric tables collected by DMS: `displayMetricTables` See Section 12.4, "Viewing Statistics Using WLST," and Section 12.7, "Monitoring Using DMS."	Administration server must be running.	Enabled by default. Accessible even when request-processing threads are hanging.
Browser ■ Detailed statistics for a specific virtual server in XML format ■ Summary report for a specific virtual server in plain-text format See Section 12.5, "Viewing stats-xml and perfdump Reports Through a Browser."	Must be enabled and configured explicitly. See Section 12.3, "Configuring URI Access to Statistics Reports."	The administration server need not be running. It is sufficient if the instance is running.
SNMP	Must be configured explicitly. See Section 12.6, "Monitoring Using SNMP."	Statistics available through network management systems.

12.2 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 by using either Fusion Middleware Control or the WLST.

Note: For information about invoking WLST, see Section 1.7.1, "Accessing WebLogic Scripting Tool."

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 Section 1.7.2, "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 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)
```

  For more information about `otd_getStatsProperties` and `otd_setStatsProperties`, see *WebLogic Scripting Tool Command Reference for Oracle Traffic Director*.

12.3 Configuring URI Access to Statistics Reports

As described in Section 12.1, "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 Section 12.8, "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 Section 12.9, "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 Section 1.7.2, "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 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

Note: For information about invoking WLST, see Section 1.7.1, "Accessing WebLogic Scripting Tool."

- 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)
  ```

For more information about the WLST commands mentioned in this section, see *WebLogic Scripting Tool Command Reference for Oracle Traffic Director*.

12.4 Viewing Statistics Using WLST

By using WLST, you can view statistics for one or all instances of a configuration.

Note: For information about invoking WLST, see Section 1.7.1, "Accessing WebLogic Scripting Tool."

- 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 Section 12.8, "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 Section 12.9, "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 Section 12.7, "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')
```

For more information about the WLST commands mentioned in this section, see the *WebLogic Scripting Tool Command Reference for Oracle Traffic Director*.

12.5 Viewing stats-xml and perfdump Reports Through a Browser

If you enable URI access to statistics as described in Section 12.3, "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
```

12.6 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.

You can use SNMP to monitor the Oracle Traffic Director instances. 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:

- Section 12.6.1, "Configuring Oracle Traffic Director Instances for SNMP Support"
- Section 12.6.2, "Configuring the SNMP Subagent"
- Section 12.6.3, "Starting and Stopping the SNMP Subagent"
- Section 12.6.4, "Viewing Statistics Using snmpwalk"

12.6.1 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 Section 1.7.2, "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 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 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)
  ```

For more information about the custom WLST commands mentioned in this section, see *WebLogic Scripting Tool Command Reference for Oracle Traffic Director*.

12.6.2 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:

```
INSTANCE_HOME/admin-server/config/snmpagt.conf
```

You can configure the SNMP subagent's settings by editing the `snmpagt.conf` file. Table 12–2 lists the key SNMP subagent parameters.

Table 12–2 SNMP Subagent Configuration Parameters

Parameter in smnpagt.conf	Description	Default Value
`agentAddress`	Ports at which the SNMP subagent receives requests	11161
`statInterval`	Statistics update frequency (seconds)	5
`cacheTimeOut`	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.

12.6.2.1 SNMP v3 User configuration

SNMP is not defaultly configured on installation, there is no access control added to OTD SNMP. To provide access control User need to add one or more snmp v3 users.

Adding a snmp v3 user

To start or stop the SNMP subagent on a node by using the Fusion Middleware Control, do the following:

1. Stop the snmp agent if running and add the following lines

Persistent file<oracle_home>/otd_domain/config/fmwconfig/components/OTD/snmp/store/ snmpagt.conf
`createUser username SHA authpassphrase [DES|AES] [privpassphrase]`

   SHA is the authentication types to use

   DES and AES are the privacy protocols to use

   authpassphrase is the password to authenticate the username

   privpassphrase is the password to encrypt the snmp request and response

Configuration file<oracle_home>/otd_domain/config/fmwconfig/components/OTD/snmp/config /snmpagt.conf
`rouser username [noauth|auth|priv]`

User can configure authentication level.

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 as mentioned, OTD SNMP agent need to start and stop once. While stopping user authentication parameters provided in "createUser" line will be encrypted and stored in to the persistent file.
3. Start the OTD SNMP agent. Make sure to start one or more instances.
4. Run the snmpwalk command. User can run the snmpcmd from remote host too.

snmpcmd(snmpwalk/snmpget)

```
snmpwalk  -m <path to OTD mib>/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
```

Deleting a snmp v3 user

Simple enough. Just remove the entries from the conf file and persistent file.

Simplifying commands by setting defaults

The most of the command line of snmpwalk can be added in to ~/.snmp/snmp.conf.

default options

defSecurityNam	SNMPv3 username
defAuthType	authentication method (either MD5 or SHA)
defSecurityLevel	security level for the user. i.e authNoPriv, authPriv etc
defAuthPassphrase	authpassphrase
defPrivType	privacy protocol to use. DES|AES
defPrivPassphrase	privpassphrase
defVersion	3
defaultport	11161
mibirds	+<path to ORACLE-TRAFFICDIRECTOR-MIB.txt>
mibs	+ORACLE-TRAFFICDIRECTOR-MIB

Run as agentx

OTD SNMP supports agentx protocol and for communicate as agentx, it needs another SNMP agent which supports agentx. OS SNMP agent (snmpd) of OEL 6 and Solaris 11 supports agentx. When OTD SNMP runs as agentx, It can not accept snmpcmds. All the requests should be made to the SNMP agent which is running as master. Master agent will for forward the requests to the OTD SNMP agent.

Configure OTD SNMP agent as agentx

The user can modify the transport layer as specified in man page of snmpd. OTD SNMP default agentx transport specifier is below:

agentxsocket tcp:127.0.0.1:705

Configure OS SNMP agent as agentx

/etc/snmp/snmpd.conf

```
master agentx

agentxsocket tcp:127.0.0.1:705
```

snmpd daemon must be started manually and connection details are logged in snmpd.log, snmp requests should be sent to snmp port of snmpd(default 161).

12.6.3 Starting and Stopping the SNMP Subagent

You can start and stop the SNMP subagent on a node by using either Fusion Middleware Control or the WLST.

Starting and Stopping the SNMP Subagent Using Fusion Middleware Control

To start or stop the SNMP subagent on a node by using the Fusion Middleware Control, do the following:

1. Log in to Fusion Middleware Control, as described in Section 1.7.2, "Displaying Fusion Middleware Control."
2. Click the **Nodes** button that is situated near the upper left corner of the page.

   A list of available nodes is displayed.
3. From the list of nodes, select the node for which you want to start or stop the SNMP subagent.

   The General Settings page is displayed.

   - To start the SNMP subagent, click **Start SNMP Subagent**. The status changes to **Running**.
   - To stop the subagent, click **Stop SNMP Subagent**. The status changes to **Running**.
4. Specify the parameters that you want to change, and then click **Save**.

   A message is displayed in the Console Messages pane indicating that the updated settings are saved.
5. Restart the administration server by clicking **Restart** in the Common Tasks pane.

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)
  ```

Note: Alternatively, you can start the SNMP agent in **agentx** mode, by configuring the sub-agent configuration file.

In agentx mode, the SNMP agent needs to communicate with the operating-system master agent (`snmpd`). So you must configure `snmpd` to listen to the agentx protocol, by doing the following:

1. Enable agentx by adding the following token to the operating-system master agent (`snmpd`) located at (`/etc/snmp/snmpd.conf`). This token enables the master agent to connect to the agentx paths you specify.

   ```
   master agentx
   ```

2. Specify the socket path and socket path permissions in the *`ORACLE_HOME`*`/admin-server/config/snmpagt.conf` file, as shown in the following example:

   Before configuring for agentx

   ```
   agentuser admin123
   agentxsocket /tmp/snmpagt-e6d7cd20/snmpagt.socket
   ```

   After configuring for agentx

   ```
   agentxsocket /tmp/snmpagt-e6d7cd20/snmpagt.socket
   agentxperms 0755 0755 admin123 admin123
   ```

3. Start `snmpd` daemon manually.

- 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)
  ```

For more information about the custom WLST commands mentioned in this section, see *WebLogic Scripting Tool Command Reference for Oracle Traffic Director*.

12.6.4 Viewing Statistics Using snmpwalk

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.

Note: Prior to using `snmpwalk`, if required, you can set most of the `snmpwalk` options in the `snmp.conf` file, located at `<user-home>/.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`. For example, `snmp.conf` enables you to set the following options:

```
defaultport  11161
defversion   3
defcommunity  public
mibdirs   +<path to ORACLE-TRAFFICDIRECTOR-MIB.txt> #
mibdirs   + <otd_install_root>/lib/snmp #
mibs  +ORACLE-TRAFFICDIRECTOR-MIB
```

After setting the above options, `snmpwalk` can be run as follows:

```
snmpwalk  <hostname> ORACLE-TRAFFICDIRECTOR-MIB::originServerTable
```

For information about all the options that can be set using snmp.conf, see the man-pages for snmp.conf.

SNMP Version 2c

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`).

The following is the syntax of the `snmpwalk` command:

```
> snmpwalk -c public -v 3 host:port oid
```

- `host` is the host name of the Oracle Traffic Director node that you want to monitor.
- `port` is the listen port of the SNMP subagent on the node. The default port specified in the `snmpagt.conf` file is `11161`.
- `oid` is the unique object identifier series for which you want to view statistics. The OID for the Oracle Traffic Director product is `1.3.6.1.4.1.111.19.190`.

Note: OIDs are assigned and maintained by the Internet Assigned Numbers Authority. In the OID for Oracle Traffic Director, the first six numbers, `1.3.6.1.4.1`, represent private enterprises, `111` is the unique identifier for Oracle and `19.190` represents the Oracle Traffic Director product. For more information about the structure of OIDs, see RFC 2578 (`http://tools.ietf.org/html/rfc2578`).

SNMP Version 3

To monitor statistics by using SNMP v3, do the following:

1. Create an SNMP v3 user by running the following command as the `root` user:

   ```
   $ sudo net-snmp-config --create-snmpv3-user -ro -a SHA1 -A abcd1234 otdadmin
   ```

   This command does the following:

   - Adds the following entry in `/var/net-snmp/snmpd.conf`:

     ```
     createUser otdadmin SHA1 "abcd1234" DES
     ```

- Adds the following entry in `/etc/net-snmp/snmp/snmpd.conf`:

  ```
  rouser otdadmin
  ```

2. Start and stop `snmpd`.

   ```
   $ sudo /etc/init.d/snmpd start
   Starting snmpd:                                            [  OK  ]

   $ sudo /etc/init.d/snmpd stop
   Stopping snmpd:                                            [  OK  ]
   ```

   As a result of starting and stopping `snmpd`, the `createUser` entry in the `/var/net-snmp/snmpd.conf` file changes as shown in the following example:

   ```
   usmUser 1 3 0x80001f8801819ee527 0x676164686100 0x676164686100 NULL
    .1.3.6.1.6.3.10.1.1.2
   0x8b6a9b458c0cb628aa5ba10ebbec48e7 .1.3.6.1.6.3.10.1.2.2
    0x8b6a9b458c0cb628aa5ba10ebbec48e7 ""
   ```

   In this example, `0x80001f8801819ee527` is the generated engine ID.

3. Run the SNMP agent in **agentx** mode.

   Run `snmpwalk` by using the following command. The default port for snmpd is 161

   ```
   snmpwalk -v3 -u otdadmin -l authNoPriv  -a SHA1 -A abcd1234 localhost:161
   1.3.6.1.4.1
   ```

Enabling the snmpwalk Command to Show MIB Object Names Instead of Numeric OIDs

When you run the `snmpwalk` command, the output would be as follows:

```
SNMPv2-SMI::enterprises.111.19.190.1.20.1.2.0.0 = INTEGER: 645
SNMPv2-SMI::enterprises.111.19.190.1.20.1.3.0.0 = Gauge32: 4
SNMPv2-SMI::enterprises.111.19.190.1.20.1.4.0.0 = Gauge32: 4
SNMPv2-SMI::enterprises.111.19.190.1.20.1.10.0.0 = Gauge32: 0
SNMPv2-SMI::enterprises.111.19.190.1.20.1.11.0.0 = Gauge32: 3072
SNMPv2-SMI::enterprises.111.19.190.1.20.1.12.0.0 = Counter64: 0
SNMPv2-SMI::enterprises.111.19.190.1.20.1.13.0.0 = Counter64: 0
SNMPv2-SMI::enterprises.111.19.190.1.20.1.14.0.0 = STRING: "0.0000"
```

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`

To enable the `snmpwalk` command to show MIB object names instead of numeric OIDs, do one of the following:

- Set the `MIBS` environment variable on the host to point to the Oracle Traffic Director MIB.

  ```
  > set env MIBS=+ORACLE-TRAFFICDIRECTOR-MIB
  ```

  Then, run the `snmpwalk` command and either `grep` the output for the required MIB object or explicitly specify the required MIB object name.

  For example, to view statistics for proxy cache parameters for an Oracle Traffic Director instance running on the node `app1`, run the following command:

```
> snmpwalk snmpwalk -c public -v 2c app1:11161
ORACLE-TRAFFICDIRECTOR-MIB::proxyCacheTable

ORACLE-TRAFFICDIRECTOR-MIB::proxyCacheEnabledFlag.0.0 = INTEGER: enabled(1)
ORACLE-TRAFFICDIRECTOR-MIB::proxyCacheCountEntries.0.0 = Counter64: 0
ORACLE-TRAFFICDIRECTOR-MIB::proxyCacheSizeHeap.0.0 = Counter64: 16498
ORACLE-TRAFFICDIRECTOR-MIB::proxyCacheCountContentHits.0.0 = Counter64: 0
ORACLE-TRAFFICDIRECTOR-MIB::proxyCacheCountContentMisses.0.0 = Counter64: 0
ORACLE-TRAFFICDIRECTOR-MIB::proxyCacheCountHits.0.0 = Counter64: 0
...
```

- Specify the Oracle Traffic Director MIB explicitly for the `snmpwalk` command by using the `-m` option.

  For example, to view the origin-server names for an Oracle Traffic Director instance running on the local host, run the following command:

```
> snmpwalk -c public -v 2c -m $ORACLE_
HOME/lib/snmp/ORACLE-TRAFFICDIRECTOR-MIB.txt localhost:11161
ORACLE-TRAFFICDIRECTOR-MIB::originServerName
```

For a list of the SNMP MIB object names that you can use to query for specific statistics, see Appendix A, "Metrics Tracked by Oracle Traffic Director."

For more information about `snmpwalk`, see the documentation at:
`http://www.net-snmp.org/docs/man/snmpwalk.html`.

12.7 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 12–1   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')
```

12.8 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. For more information, see Section 12.4, "Viewing Statistics Using WLST" and Section 12.5, "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_OTD_rechaita" versionServer="Oracle Traffic Director
12.2.1.0.0 B20150908.132940 (Linux)" timeStarted="1442991226" secondsRunning="119"
ticksPerSecond="1000" maxProcs="1" maxThreads="512" flagProfilingEnabled="1"
load1MinuteAverage="0.070000" load5MinuteAverage="0.070000"
load15MinuteAverage="0.050000" rateBytesTransmitted="10087"
rateBytesReceived="4929" requests1MinuteAverage="0.000000"
requests5MinuteAverage="0.000000" requests15MinuteAverage="0.000000"
errors1MinuteAverage="0.000000" errors5MinuteAverage="0.000000"
errors15MinuteAverage="0.000000" responseTime1MinuteAverage="0.000000"
responseTime5MinuteAverage="0.000000" 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="7236" mode="active" timeStarted="1442991226"
countConfigurations="1" sizeVirtual="862916" sizeResident="32288"
fractionSystemMemoryUsage="0.0022">
            <connection-queue connectionQueueId="cq1" countTotalConnections="0"
countQueued="0" peakQueued="0" maxQueued="12288" countOverflows="0"
countTotalQueued="0" ticksTotalQueued="0" countQueued1MinuteAverage="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="0" maxConnections="24576" 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" timeStarted="1442991226"
```

```
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" timeStarted="1442991226"
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" timeStarted="1442991226"
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" timeStarted="1442991226"
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" timeStarted="1442991226" connectionQueueId="cq1">
                <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" timeStarted="1442991226" connectionQueueId="cq1">
                <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" timeStarted="1442991226" connectionQueueId="cq1">
                <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" timeStarted="1442991226" connectionQueueId="cq1">
                <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>
        </process>
        <virtual-server id="OTD" flagEnabled="1">
            <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"/>
            <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="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"/>
            </route>
        </virtual-server>
        <virtual-server id="virtual-server-1" flagEnabled="1">
            <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"/>
            <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="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"/>
            </route>
        </virtual-server>
        <cache flagEnabled="1" countEntries="0" sizeHeapCache="16516"
countContentHits="0" countContentMisses="0" countHits="0"
countRevalidationRequests="0" countRevalidationFailures="0"/>
        <cpu-info cpu="1" percentIdle="99.268188" percentUser="0.426176"
percentKernel="0.305636"/>
        <cpu-info cpu="2" percentIdle="99.476388" percentUser="0.336554"
percentKernel="0.187058"/>
        <cpu-info cpu="3" percentIdle="99.518723" percentUser="0.300850"
percentKernel="0.180426"/>
        <cpu-info cpu="4" percentIdle="99.537347" percentUser="0.292632"
percentKernel="0.170021"/>
    </server>
</stats>
```

12.9 Sample Plain-Text (perfdump) Report

This section contains a sample perfump 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 Section 12.4, "Viewing Statistics Using WLST" and Section 12.5, "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 12.2.1.0.0 B20150908.132940 (Linux)
Server started Tue Sep 22 23:53:45 2015
Process 7236 started Tue Sep 22 23:53:45 2015
```

```
ConnectionQueue
Current/Peak/Limit Queue Length           0/0/12288
Total Connections Queued                  0
Average Queue Length (1, 5, 15 minutes)   0.00, 0.00, 0.00
Average Queuing Delay                     0.00 milliseconds
HTTP Listener http-listener-1
Address                 0.0.0.0:8080
Acceptor Threads        1
Default Virtual Server  OTD
KeepAliveInfo
KeepAliveCount       0/24576
KeepAliveHits        0
KeepAliveFlushes     0
KeepAliveRefusals    0
KeepAliveTimeouts    0
KeepAliveTimeout     30 seconds
SessionCreationInfo
Active Sessions             0
Keep-Alive Sessions         0
Keep-Alive threads          4
HTTP Sessions current/max   8/516
TCP Sessions current/max    4/4
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               16516
Thread Pool NativePool
Idle/Peak/Limit              1/1/128
Work Queue Length/Peak/Limit  0/0/0
DNSCacheInfo
enabled            yes
CacheEntries       0/1024
HitRatio           0/0 (  0.00%)
Async DNS disabled
Performance Counters
Total number of requests                  0
Average Request processing time      0.0000
Total Request processing time        0.0000
default-bucket (Default bucket)
Counter Name              Average        Total      Percent
-------------------------------------------------------------
Number of Requests                           0    (  0.00%)
Number of Invocations                        0    (  0.00%)
Latency                    0.0000       0.0000    (  0.00%)
Function Processing Time   0.0000       0.0000    (  0.00%)
Total Response Time        0.0000       0.0000    (  0.00%)
HTTP Origin Servers
No HTTP origin servers are configured
TCP Origin Servers
No TCP origin servers are configured
TCP Proxy
Active Connections                 0
Avg Duration                       0.00 seconds
Requests (timeout/aborted/total)   0/0/0
```

13 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. OTD sends information about the events that it has detected through a HTTP POST message to this URL. The HTTP POST request will contain event information in JSON format. Oracle Traffic Director supports notifications for the following two events:

- Origin server status change event
- Request limit exceeded event

13.1 Origin server status change event

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.

13.1.1 Subscribing to origin server status event using Fusion Middleware Control

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 (or) right corner of the page.
3. Select Administration > OTD Configurations.

   A list of available OTD Configurations is displayed.

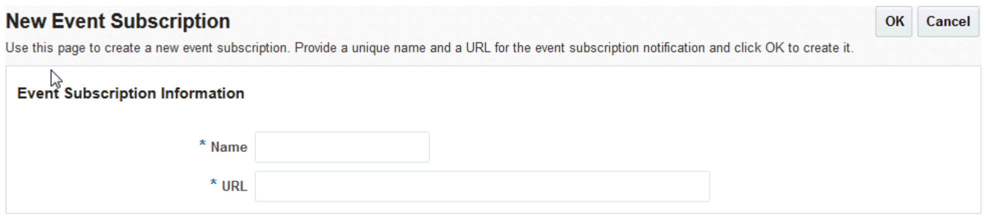
   Select a configuration for which Event Subscriptions will be enabled.
4. Select Configuration > Advanced Configuration > Event Subscriptions.

   The **Events Subscriptions** page is displayed.

5. In the Common Tasks pane, click **Create** under Event Subscription.

   The New Event Subscription wizard starts.

Figure 13–1 New Event Subscription Wizard



New Event Subscription

Use this page to create a new event subscription. Provide a unique name and a URL for the event subscription notification and click OK to create it.

Event Subscription Information

* Name	
* URL	

6. 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.
7. 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.

13.1.2 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)
```

For more information about the WLST commands mentioned in this section, see *WebLogic Scripting Tool Command Reference for Oracle Traffic Director*.

To know the details of the command and its each parameter/option, use help, as shown below:

```
help('otd_createEventSubscription')
```

13.1.3 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 13–1 JSON properties common to all events

JSON property	Description
event-type	Type of the event. Value: `origin-server-status-change`.
domain-name	Name of the domain where Oracle Traffic Director instance is configured.
instance-name	Name of the Oracle Traffic Director instance.
timestamp	The time when the event occurred and detected by OTD.

Table 13–2 JSON property specific to origin-server-status-change event

JSON property	Description
pool-name	Origin server pool name to which the origin server is associated with.
host	Origin server host for which the status being sent.
port	Origin server port.
status	Online or offline.
reason	Reason for Oracle Traffic Director marking the origin server as offline or online.
protocol	Health check protocol used.

13.1.3.1 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"
      ]
   }
```

13.1.3.2 Example

```
Content-Type: application/json

Content:

{
   "v1.0":{
      "event-type":"origin-server-status-change",
      "domain-name":"base_domain",
      "instance-name":"otd1",
      "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"
   }
}
```

13.1.4 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.

13.2 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. For more information on configuring request limits to a server, see Section 10.7.2, "Configuring Request Limits for a Virtual Server".

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.

13.2.1 Subscribing to request limit exceeded event Using WLST

- Create an event subscription. For more information, see "Creating an event subscription" in "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. See section Section 10.7.2, "Configuring Request Limits for a Virtual Server".

13.2.2 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 13–3 JSON properties common to all events

JSON property	Description
event-type	Type of the event. Value: `request-limit-exceeded`.
domain-name	Name of the domain where Oracle Traffic Director instance is configured.
instance-name	Name of the Oracle Traffic Director instance.

Table 13–4 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. For example, if monitor=$ip is specified in the -request-limit rule, the JSON property `monitor` will be set to the value of the "$ip"variable, the Client IP address. If the monitor is not specified in the request-limit rule, this property will be set to `unnamed`.
total-queue-overflows	Total requests rejected due to queue overflow.
total-queue-timeouts	Total requests rejected due to timeout while waiting in the queue.

13.2.2.1 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"
  ]
}
```

13.2.2.2 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
   }
 }
 ]
}
}
```

14 Configuring Oracle Traffic Director for High Availability

This chapter describes the high-availability capabilities of Oracle Traffic Director. It contains the following sections:

- Overview of High-Availability
- Creating and Managing Failover Groups

14.1 Overview of High-Availability

In the context of Oracle Traffic Director instances, high availability includes the following capabilities:

- Receive and serve client requests without downtime caused by hardware failures, kernel crashes, and network issues.
  - You can set up a highly available traffic routing and load-balancing service for your enterprise applications and services by configuring two Oracle Traffic Director instances to provide active-active or active-passive failover.
  - If an Oracle Traffic Director process crashes, it restarts automatically.

    Oracle Traffic Director provides two levels of availability, application level and node level. Application level availability is the default feature and does not require any additional configuration. Application level availability ensures that the load balancing service is monitored through the Oracle Traffic Director Watchdog daemon and is available even during application level failures such as process crash. This feature ensures that Oracle Traffic Director as a software load balancer can continue to front-end requests to back-end applications even if there is a software issue within the load balancing service. The node level availability ensures that Oracle Traffic Director continues to front-end requests to back-end applications even if the system/vServer crashes because of issues such as CPU failure or memory corruption. For node level availability, Oracle Traffic Director must be installed on two compute nodes or vServers, and a failover group must be configured between them.

    To provide high availability to the Oracle Traffic Director instance itself, each load balancer server instance includes at least three processes, a watchdog process, a primordial process, and one or more load balancer processes. The watchdog process spawns the primordial, which then spawns the load balancer processes. The watchdog process and the primordial process provide a limited level of high availability within the Oracle Traffic Director processes. If the load balancer process or primordial process terminates abnormally for any reason, then Oracle Traffic Director watchdog is responsible for restarting these services, to ensure that Oracle Traffic Director as a software load balancer service continues to be available. An Oracle Traffic Director instance will have exactly one watchdog process, one primordial process and one or more load balancer processes.

  – Most configuration changes to Oracle Traffic Director instances can be deployed dynamically, without restarting the instances and without affecting requests that are being processed. For configuration changes that require instances to be restarted, the Fusion Middleware Control displays a prompt to restart the instances. However, this prompt is not displayed in the WLST commands, an error is logged in the server log.

- Distribute client requests reliably to origin servers in 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. For more information, see Section 5.7, "Configuring Health-Check Settings for Origin-Server Pools."
  - In each origin-server pool, you can designate a few servers as backup servers. Oracle Traffic Director sends requests to the backup servers only when none of the primary servers in the pool is available. For more information, see Section 5.3, "Modifying an Origin-Server Pool."
  - You can reduce the possibility of requests being rejected by origin servers due to a connection overload, by specifying the maximum number of concurrent connections that each origin server can handle.

    For each origin server, you can also specify the duration over which the rate of sending requests to the server is increased. This capability helps minimize the possibility of requests getting rejected when a server that was offline is in the process of restarting.

    For more information, see Section 6.3, "Modifying an Origin Server."

14.1.1 High Availability in Network Topology

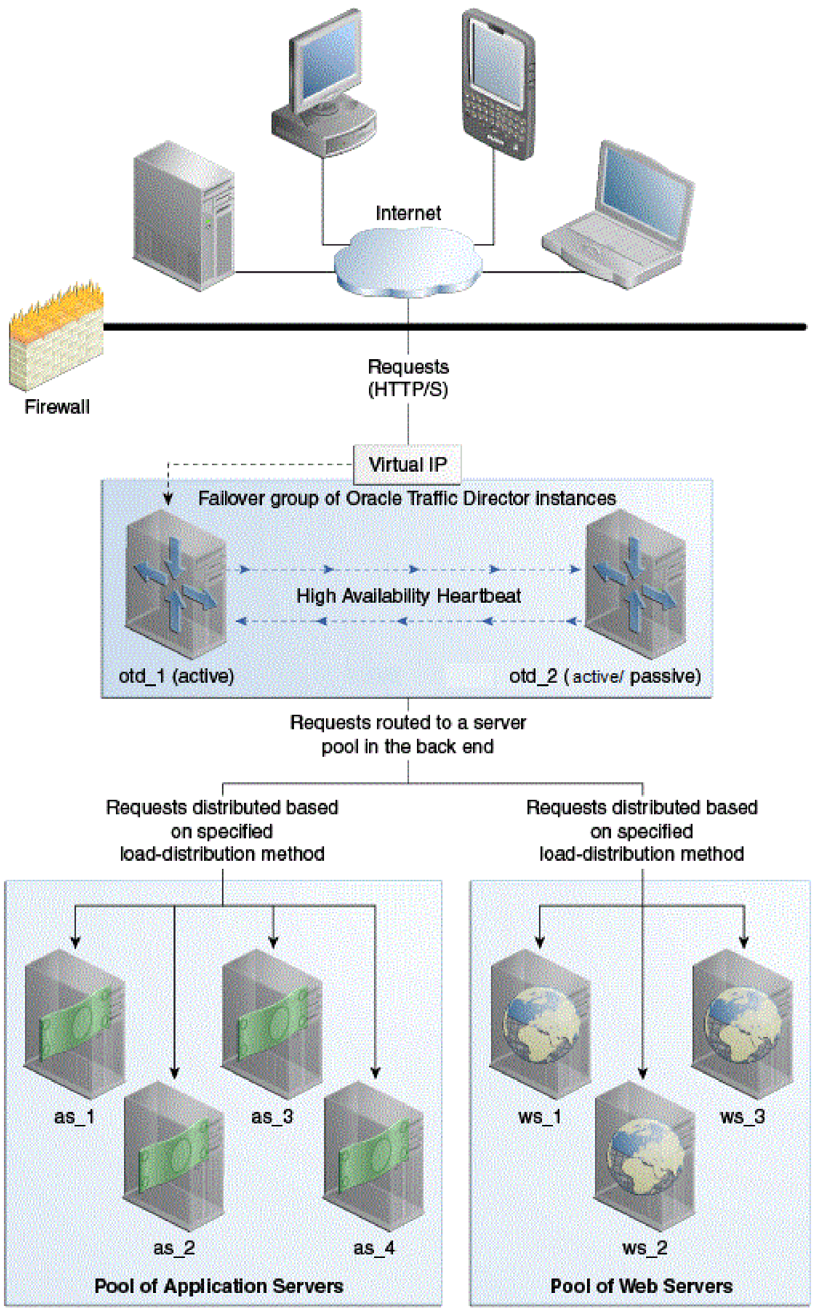
You can ensure high availability of Oracle Traffic Director instances by combining two or more Oracle Traffic Director instances in a *failover group* represented by one or two virtual IP (VIP) addresses. Both the hosts in a failover group must run the same operating system version, use identical patches and service packs, and run Oracle Traffic Director instances of the same configuration.

Note:

- You can create multiple failover groups for the same instance, but with a distinct VIP address for each failover group.
- On Oracle SuperCluster and Exalogic (Solaris), Oracle Traffic Director can be configured for high availability only when installed on a global zone. In addition, all administration nodes must be running on the global zone.

Figure 14–1 shows Oracle Traffic Director network topology.

Figure 14–1 Oracle Traffic Director Network Topology



The topology shown in Figure 14–1 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 Section 14.1.2, "Failover configuration modes".

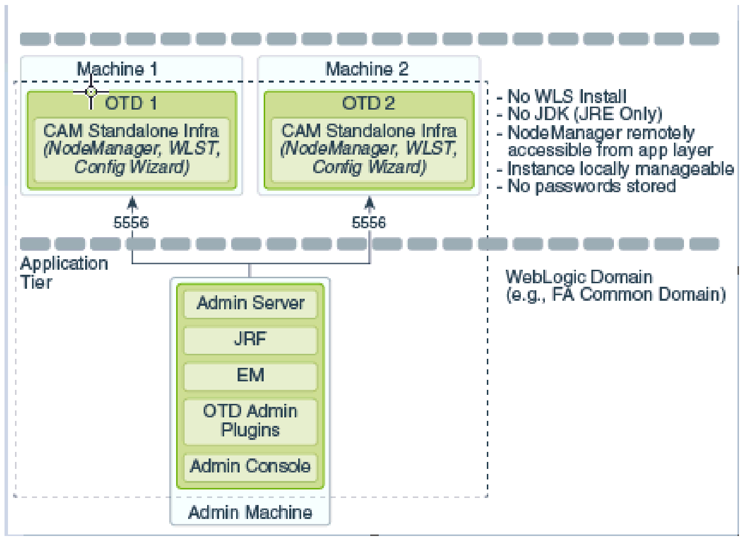
Note that Figure 14–1 shows only two server pools in the back end, but you can configure Oracle Traffic Director to route requests to servers in multiple server pools.

14.1.2 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.

Figure 14–2 High Availability deployment of Oracle Traffic Director

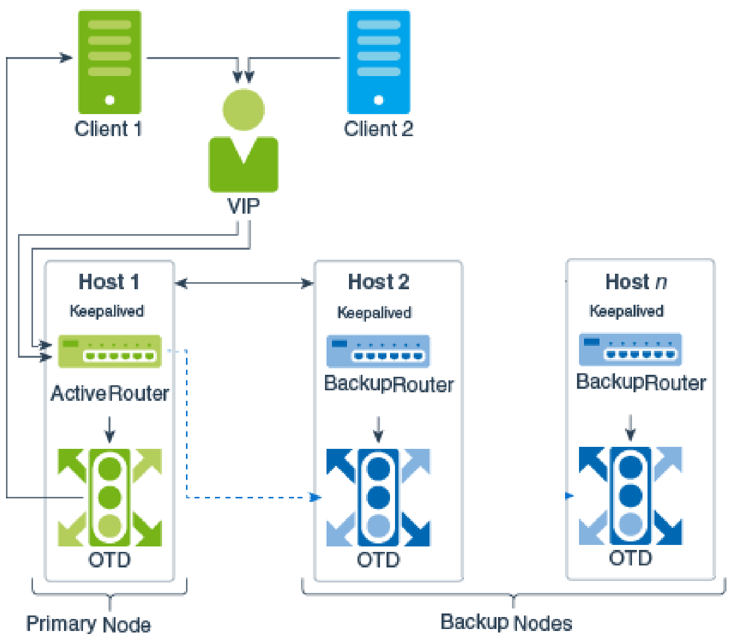

Machine 1
Machine 2
OTD 1
CAM Standalone Infra (NodeManager, WLST, Config Wizard)
OTD 2
CAM Standalone Infra (NodeManager, WLST, Config Wizard)
- No WLS Install
- No JDK (JRE Only)
- NodeManager remotely accessible from app layer
- Instance locally manageable
- No passwords stored
5556
5556
Application Tier
WebLogic Domain (e.g., FA Common Domain)
Admin Server
JRF
EM
OTD Admin Plugins
Admin Console
Admin Machine

14.1.3 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. To improve resource utilization, you can configure the two Oracle Traffic Director instances in active-active mode with two virtual IP addresses. Each instance caters to requests received on one virtual IP address *and* backs up the other instance.

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 vrrpd (native) for Solaris.

Figure 14–3 Failover in Active-Passive Mode



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`.

Note: Ensure to install Keepalived v1.2.12(minimum version required).

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.

Caution: On a 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`.

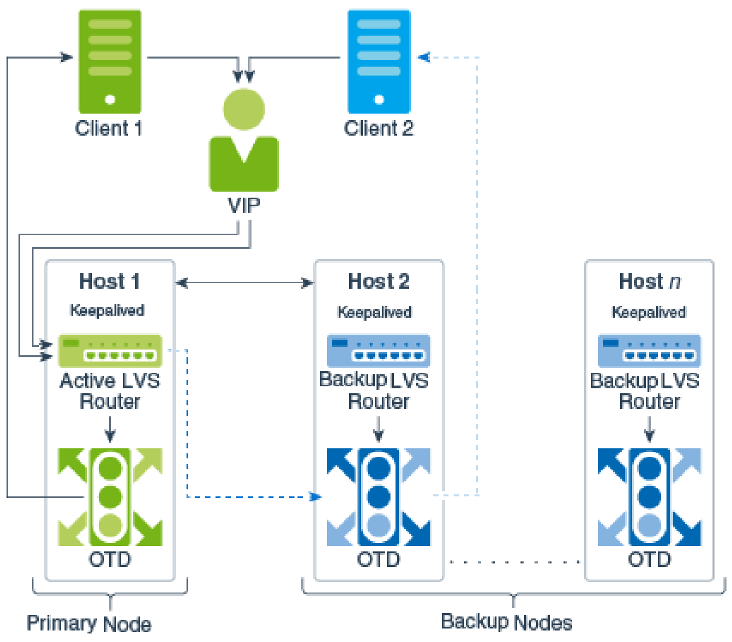
14.1.4 Failover in Active-Active Mode

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.

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)

Figure 14–4 Failover in Active-Active Mode



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 OTD instances
- Share the existing connection information to the backup nodes via multicasting.
- To check the integrity of the services on each OTD instance. In case, any OTD fails then that OTD instance will be removed from the LVS configuration and when it

comes back online then it will be added again.

14.2 Creating and Managing Failover Groups

This section contains the following topics:

- Section 14.2.1, "Creating Failover Groups"
- Section 14.2.2, "Managing Failover Groups"

14.2.1 Creating Failover Groups

This section describes how to implement a highly available pair of Oracle Traffic Director instances by creating failover groups.

Note: For information about invoking WLST, see Section 1.7.1, "Accessing WebLogic Scripting Tool."

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 Oracle Traffic Director **nodes** that you want to configure as primary and backup nodes in the failover group. The nodes should be in the same subnet.

  Note that the administration nodes that you select should have Oracle Traffic Director instances present on them for the specified configuration.
- Identify the **network interface** for each node.

  For each network interface that is currently up on the host, the administration server compares the network part of the interface's IP address with the network part of the specified VIP. 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.

- Make sure that the router ID for each failover group is unique. For every subsequent failover group that you create, the default router ID is decremented by one: 254, 253, and so on.

14.2.1.1 Creating Failover Groups Using Fusion Middleware Control

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

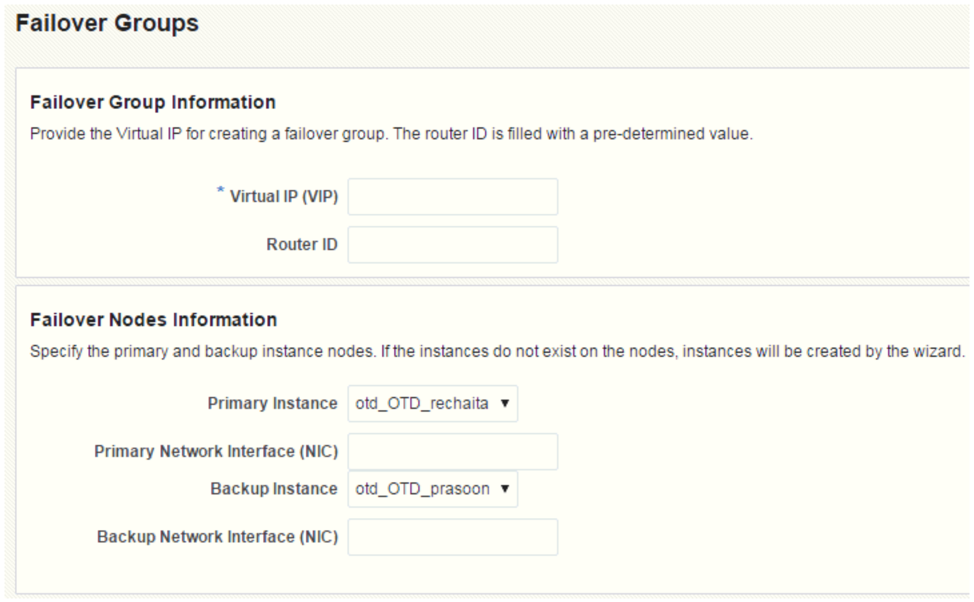
1. Log in to Fusion Middleware Control, as described in Section 1.7.2, "Displaying Fusion Middleware Control."
2. Click the WebLogic Domain button at the upper left corner of the page.
3. Select Administration > OTD Configurations.

   A list of the available configurations is displayed.
4. Select the configuration for which you want to create a 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.

Figure 14–5 New Failover Group Wizard



Failover Groups

Failover Group Information

Provide the Virtual IP for creating a failover group. The router ID is filled with a pre-determined value.

* Virtual IP (VIP)	
Router ID	

Failover Nodes Information

Specify the primary and backup instance nodes. If the instances do not exist on the nodes, instances will be created by the wizard.

Primary Instance	otd_OTD_rechaita ▼
Primary Network Interface (NIC)	
Backup Instance	otd_OTD_prasoon ▼
Backup Network Interface (NIC)	

8. Follow the on-screen prompts to complete creation of the failover group by using the details—virtual IP address, network interface, host names of administration nodes, 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.

Note: At this point, the two nodes form an active-passive pair. To convert them into an active-active pair, create another failover group with the same two nodes, but with a different VIP and with the primary and backup roles reversed.

14.2.1.2 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: When 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 more information about `otd_startFailover`, see *WebLogic Scripting Tool Command Reference for Oracle Traffic Director.*

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)
```

For more information about `otd_createFailoverGroup`, see *WebLogic Scripting Tool Command Reference for Oracle Traffic Director*.

14.2.2 Managing Failover Groups

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. For more information about `otd_startFailover` or `otd_stopFailover`, see the *WebLogic Scripting Tool Command Reference for Oracle Traffic Director*.

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. Note that to change the VIP or any property of a failover group, you should delete the failover group and create it afresh.

You can view, modify, and delete failover groups by using either Fusion Middleware Control or the WLST.

Note: For information about invoking WLST, see Section 1.7.1, "Accessing WebLogic Scripting Tool."

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 Section 1.7.2, "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 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 node as the primary or backup node 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)
```

For more information, see *WebLogic Scripting Tool Command Reference for Oracle Traffic Director*.

15

Tuning Oracle Traffic Director for Performance

This chapter describes how you can use statistical data about Oracle Traffic Director instances and virtual servers to identify potential performance bottlenecks. It also describes configuration changes that you can make to improve Oracle Traffic Director performance.

This chapter contains the following sections:

15.1 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.

15.2 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 Section 15.12.1, "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.

15.3 Tuning the Thread Pool and Connection Queue

This section contains the following topics:

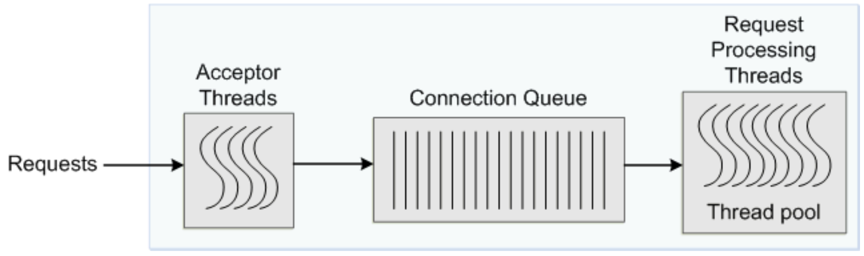
- Section 15.3.1, "About Threads and Connections"
- Section 15.3.2, "Reviewing Thread Pool Metrics for an Instance"
- Section 15.3.3, "Reviewing Connection Queue Metrics for an Instance"
- Section 15.3.4, "Tuning the Thread Pool and Connection Queue Settings"

15.3.1 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.

Figure 15–1 depicts the connection handling process.

Figure 15–1   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.

15.3.2 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 Section 15.3.3, "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.

15.3.3 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 the average response time, consider increasing the number of threads in the thread pool.

15.3.4 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.

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 Section 1.7.2, "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 **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'
```

For more information about the WLST commands mentioned in this section, see *WebLogic Scripting Tool Command Reference for Oracle Traffic Director*.

15.4 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 the CLI, as described in Section 9.3, "Modifying a Listener."

15.5 Tuning Keep-Alive Settings

This section contains the following topics:

- Section 15.5.1, "About Keep-Alive Connections"

- Section 15.5.2, "Reviewing Keep-Alive Connection Settings and Metrics"
- Section 15.5.3, "Tuning Keep-Alive Settings"

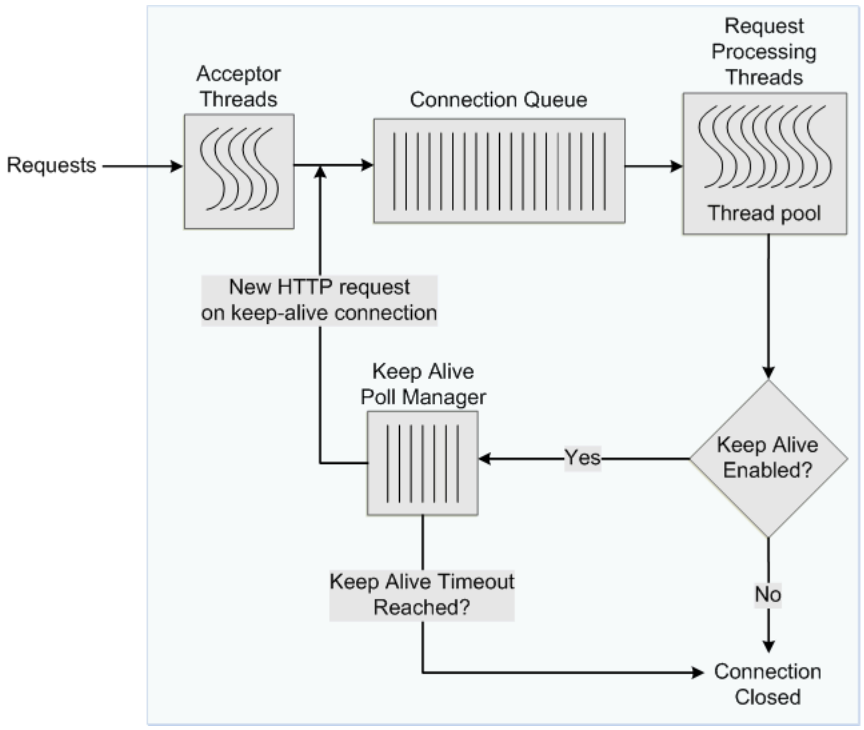
15.5.1 About Keep-Alive Connections

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 helps reduce 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.

Figure 15–2 depicts the connection handling process when keep-alive is enabled.

Figure 15–2 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. When a keep-alive request is received, if there are more open connections waiting for requests than the specified maximum number, the oldest connection is closed. In addition, you can specify the period after which inactive keep-alive connections should be closed.

15.5.2 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 `perdump` 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 Section 15.2, "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.

15.5.3 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 Section 1.7.2, "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 > 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)
  ```

For more information about the WLST commands mentioned in this section, see the *WebLogic Scripting Tool Command Reference for Oracle Traffic Director*.

15.6 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 Section 1.7.2, "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 > 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 unchunk timeout, run the following command:

  ```
  props = {}
  props['configuration'] = 'foo'
  props['unchunk-timeout'] = '120'
  otd_setHttpProperties(props)
  ```

For more information about the WLST commands mentioned in this section, see *WebLogic Scripting Tool Command Reference for Oracle Traffic Director*.

15.7 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:

- Section 15.7.1, "Viewing DNS Cache Settings and Metrics"
- Section 15.7.2, "Configuring DNS Cache Settings"

15.7.1 Viewing DNS Cache Settings and Metrics

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.

15.7.2 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 Section 1.7.2, "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 **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)
```

For more information about the WLST commands mentioned in this section, see *WebLogic Scripting Tool Command Reference for Oracle Traffic Director*.

15.8 Tuning SSL/TLS-Related Settings

This section contains the following topics:

- Section 15.8.1, "SSL/TLS Session Caching"
- Section 15.8.2, "Ciphers and Certificate Keys"

15.8.1 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 About SSL). 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 Section 1.7.2, "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
  max-ssl3-tls-session-age=86400
  ```

- 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)
  ```

For more information about the WLST commands mentioned in this section, see the *WebLogic Scripting Tool Command Reference for Oracle Traffic Director*.

15.8.2 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 and RC4—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 Section 10.1.4, "Configuring SSL/TLS Ciphers for a Listener."

For information about SNI host matching, see Section 10.1.6, "About Strict SNI Host Matching."

- The initial SSL/TLS handshake process takes less time for RSA certificates with small key sizes—512, 1024 and 2048 bits—than for certificates with large key size— 4096 bits.

  For information about creating self-signed certificates and certificate-signing requests, see Section 10.3, "Managing Certificates."

15.9 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 Section 1.7.2, "Displaying Fusion Middleware Control."Section 1.7.2, "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 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 more information about the WLST commands mentioned in this section, see *WebLogic Scripting Tool Command Reference for Oracle Traffic Director*.

For information about viewing logs, configuring log preferences, rotating logs, and so on, see Chapter 11, "Managing Logs."

15.10 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 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 Section 1.7.2, "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 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.

   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:

1. 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.

   ---

2. 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.

3. 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)
```

For more information about the WLST commands mentioned in this section, see the *WebLogic Scripting Tool Command Reference for Oracle Traffic Director*.

15.11 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 Section 1.7.2, "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 on the Route Settings page:

Section of the Route Settings Page	Field/s
Advanced Settings: Client Configuration for Connections with Origin Servers	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*.

15.12 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.

15.12.1 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.

15.12.2 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.

15.12.3 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 15–1, " Tuning Solaris for Performance Benchmarking".

15.12.4 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/rdsk/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
```

15.12.5 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 `svc_t 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.

15.12.6 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.

15.12.7 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.

15.12.8 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 15–1 Tuning Solaris for Performance Benchmarking

Parameter	Scope	Default Value	Tuned Value	Comments
`rlim_fd_cur`	`/etc/system`	256	65536	Soft limit
`rlim_fd_max`	`/etc/system`	65536	65536	Process open file descriptors limit; accounts for the expected load (for the associated sockets, files, and pipes if any).
`_time_wait_interval`	`ipadm set-prop`	60000	600000	Set on clients as well.
`_conn_req_max_q`	`ipadm set-prop`	128	1024	
`_conn_req_max_q0`	`ipadm set-prop`	1024	4096	
`_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_anon_port`	`ipadm set-prop`	32768	65535	Set on clients as well.

Table 15–1 (Cont.) Tuning Solaris for Performance Benchmarking

Parameter	Scope	Default Value	Tuned Value	Comments
`send_buf`	`ipadm set-prop`	49152	128000	To increase the transmit buffer.
`recv_buf`	`ipadm set-prop`	128000	1048576	To increase the receive buffer.

16 Diagnosing and Troubleshooting Problems

This chapter describes 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

16.1 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./technetwork/middleware/ias/downloads/fusion-certification-100350.html`

2. Look for a solution to the problem in Section 16.2, "Solutions to Common Errors."

3. Check whether the information in Section 16.3, "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 Table 11–1, " Server Log Levels". 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 Section 11.3, "Configuring Log Preferences."

5. Contact Oracle for support, as described in Section 16.4, "Contacting Oracle for Support."

16.1.1 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. For more information, see Section 14.2, "Creating and 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 you provide the netmask bits and not the actual netmask value. For more information, see Section 14.2.1, "Creating Failover Groups."

16.2 Solutions to Common Errors

This section provides solutions to the following problems:

- Section 16.2.1, "Startup failure: could not bind to port"
- Section 16.2.2, "Unable to start server with HTTP listener port 80"
- Section 16.2.3, "Oracle Traffic Director consumes excessive memory at startup"
- Section 16.2.4, "Operating system error: Too many open files in system"
- Section 16.2.5, "Unable to stop instance after changing the temporary directory"
- Section 16.2.6, "Unable to restart the administration server"
- Section 16.2.7, "Oracle Traffic Director does not maintain session stickiness"

16.2.1 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 Section 9.3, "Modifying a Listener."

16.2.2 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.

For more information, see the Linux `xinetd` documentation.

16.2.3 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 (Section 15.5.3, "Tuning Keep-Alive Settings"), the size of the connection queue (Section 15.3.4, "Tuning the Thread Pool and Connection Queue Settings"), and the maximum number of connections to individual origin servers (Section 6.3, "Modifying an Origin Server").

16.2.4 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. For more information, see Section 15.2, "Tuning the File Descriptor Limit."

16.2.5 Unable to stop instance after changing the temporary directory

This error occurs when, after changing the temporary directory for a configuration, you deploy the change without stopping the instances, and then attempt to stop the instances later. The temporary directory is the directory (on the administration node) in which the process ID and socket information for the instances of the configuration are stored.

When this error occurs, the following message is written to the server log:

```
OTD-63585 An error occurred while stopping the server. For details, see the server
log.
```

To Avoid This Error

If you change the temporary directory for a configuration, you should first stop all the instances of the configuration, deploy the changes, and then start the instances.

To Solve This Problem

Kill the Oracle Traffic Director instance.

1. Find out the current temporary directory for the configuration by doing one of the following:

- Run the `otd_getConfigurationProperties` WLST command, as shown in the following example:

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

  temp-path=/tmp/net-test-a46e5844
  ...
  ```

- Log in to Fusion Middleware Control, select the required configuration, and select **Advanced Settings**. On the resulting page, look for the Temporary Directory field.

Note the path to the temporary directory.

2. Find out the process ID of the running instance by running the following command:

   ```
   cat temp_dir/pid
   ```

   `temp_dir` is the full path to the temporary directory that you noted in step 1.

   Note the process ID that this command returns.

3. Kill the process, by running the following command:

   ```
   kill pid
   ```

   `pid` is the process ID that you noted in step 2.

16.2.6 Unable to restart the administration server

In Linux systems, the cron script tmpwatch, located at `/etc/cron.daily/tmpwatch`, is set to execute everyday by default. This script removes all files that are older than 240 hours (10 days) from all `/tmp` directories in the administration server. Hence, if the administration server is not restarted for more than 10 days, the default pid file is removed. This in turn prevents the administration server from being restarted after 10 days.

To Avoid This Problem

- **Change `temp-path` location**: In the file, `<otd-home>/admin-server/config/server.xml`, change the `temp-path` value to a location where the server user has exclusive rights. For example, change it to, `<temp-path>/var/tmp/https-test-1234</temp-path>`. In addition, make sure that the new `temp-path` is not being monitored by the tmpwatch script.
- **Change the cron script**: Remove the value `240 /tmp` from the cron script for tmpwatch. Use the `-X/--exclude-pattern` option to exclude a directory from being monitored by tmpwatch. For more information about using this option, see the man-page for tmpwatch.

16.2.7 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.

Oracle Traffic Director 11.1.1.5 needs to be explicitly configured to honor session persistence when a back-end application server uses HTTP Session cookie other than the default `JSESSIONID`. On the other hand, Oracle Traffic Director 11.1.1.6 honors session persistence on receiving any cookie from the origin server.

Note: Oracle Traffic Director needs additional patches within WebLogic 10.3.x to maintain URI based session stickiness.

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.

16.3 Frequently Asked Questions

This section contains the following subsections:

- Section 16.3.1, "What is a "configuration"?"
- Section 16.3.2, "How do I access Fusion Middleware Control?"
- Section 16.3.3, "Why do I see a certificate warning when I access Fusion Middleware Control for the first time?"
- Section 16.3.4, "Can I manually edit configuration files?"
- Section 16.3.5, "In Fusion Middleware Control, what is the difference between saving a configuration and deploying it?"
- Section 16.3.6, "Why is the "Deployment Pending" message displayed in Fusion Middleware Control?"
- Section 16.3.7, "Why is the "Instance Configuration Deployed" message is displayed in Fusion Middleware Control?"
- Section 16.3.8, "Why does Fusion Middleware Control session end abruptly?"
- Section 16.3.9, "How do I access the WLST?"
- Section 16.3.10, "Why is a certificate warning message displayed when I tried to access the WLST for the first time?"
- Section 16.3.11, "How do I find out the short names for the options of a WLST command?"
- Section 16.3.12, "Why am I unable to select TCP as the health-check protocol when dynamic discovery is enabled?"
- Section 16.3.13, "After I changed the origin servers in a pool to Oracle WebLogic Servers, they are not discovered automatically, though dynamic discovery is enabled. Why?"

16.3.1 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 Section 1.4, "Oracle Traffic Director Terminology."

16.3.2 How do I access Fusion Middleware Control?

See Section 1.7.2, "Displaying Fusion Middleware Control."

16.3.3 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.

16.3.4 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 Section 3.3, "Activate Configuration Changes."

16.3.5 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 Section 3.3, "Activate Configuration Changes."

16.3.6 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 Section 3.3, "Activate Configuration Changes."

16.3.7 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.

16.3.8 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.

16.3.9 How do I access the WLST?

See Section 1.7.1, "Accessing WebLogic Scripting Tool."

16.3.10 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 CLI, the warning message is not displayed.

16.3.11 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.

16.3.12 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.

16.3.13 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 Chapter 5, "Managing Origin-Server Pools."
2. Update the appropriate routes to point to the new pool, as described in Section 7.4, "Configuring Routes."
3. Reconfigure the Oracle Traffic Director instances by using the `softRestart` WLST command, as described in Section 4.4, "Updating Oracle Traffic Director Instances Without Restarting."

16.3.14 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 Section 1.7.2, "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 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`.

  For more information about the WLST commands mentioned in this section, see *WebLogic Scripting Tool Command Reference for Oracle Traffic Director*.

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-alive
]]
[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]
```

16.3.15 How do I enable SSL/TLS for an Oracle Traffic Director instance?

See Section 10.1, "Configuring SSL/TLS Between Oracle Traffic Director and Clients."

16.3.16 How do I find out which SSL/TLS cipher suites are supported and enabled?

See Section 10.1.4, "Configuring SSL/TLS Ciphers for a Listener."

16.3.17 How do I view a list of installed certificates?

See Section 10.3.4, "Viewing a List of Certificates."

16.3.18 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.

16.3.19 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] = {
              extension type renegotiation_info, length [1] = {
   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.

16.3.20 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 serve. 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 Section 11.3, "Configuring Log Preferences."

16.3.21 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 about the WLST commands mentioned in this section, see *WebLogic Scripting Tool Command Reference for Oracle Traffic Director*.

16.3.22 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 Section 7.4, "Configuring Routes".

Note that Oracle Traffic Director cannot maintain case sensitivity of the HTTP request headers while forwarding them to origin servers.

16.3.23 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.

16.4 Contacting Oracle for Support

If you have a service agreement with Oracle, you can contact Oracle Support (`http://support.`) 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./`.

  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 Fusion Middleware Control screen for which you require help.
- Zip file containing the configuration files for the configuration in which you encountered the error.

  INSTANCE_HOME`/admin-server/config-store/`*config_name*`/current.zip`

- Zip file containing the configuration files for the last error-free configuration.

  INSTANCE_HOME`/admin-server/config-store/`*config_name*`/backup/`*date_time*`.zip`

- 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.

A.1 Instance Metrics

This section lists the metrics that Oracle Traffic Director tracks for a server instance. 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
The instance identifier	NA	server: id
The instance software version. Example: Oracle Traffic Director 12.2.1.0.0 B10/11/2014 21:29 (Linux)	NA	server: version
The instance start time	NA	server: timeStarted

Table A–1 (Cont.) Instance Metrics

Metric	Object Name in the SNMP MIB	stats-xml Element: Attribute
Number of seconds the instance has been running	instanceUptime	server: secondsRunning
Maximum number of processes that are part of the instance	NA	server: maxProcs
Maximum number of worker threads in each instance	NA	server: maxThreads
Profiling flag	NA	server: flagProfilingEnabled
Average load in the last 1 minute	instanceLoad1Minute Average	server: load1MinuteAverage
Average load in the last 5 minutes	instanceLoad5Minute Average	server: load5MinuteAverage
Average load for in the last minutes	instanceLoad15MinuteAverage	server: load15MinuteAverage
Number of octets received	instanceInOctets	server: countBytesReceived
Number of octets transmitted	instanceOutOctets	server: countBytesTransmitted
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
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 octets transmitted on the network per second	instanceNetworkInOctets	NA
Number of octets received on the network per second	instanceNetworkOut Octets	NA

A.2 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

Metric	Object Name in the SNMP MIB	stats-xml Element: Attribute
Process identifier	processId	process: pid
Process activity mode	NA	process: mode
Process start time	NA	process: timeStarted
Number of times a configuration has been loaded	NA	process: countConfigurations
Number of request processing threads currently available	processCountThreads	process: countThreads
Number of request processing threads currently idle	processIdleThreads	process: idleThreads
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

A.3 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

Metric	Object Name in the SNMP MIB	stats-xml Element: Attribute
Connection Queue identifier	connectionQueueId	connection-queue: connectionQueueId
Total number of connections added to this connection queue since the instance started	connectionQueueTotal	connection-queue: countTotalConnections
Number of connections currently in connection queue	connectionQueueCount	connection-queue: countQueued
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
Number of connections rejected due to connection queue overflow	connectionQueueOverflows	connection-queue: countOverflows
Total number of connections that have been added to this connection queue since startup	connectionQueueTotalQueued	connection-queue: countTotalQueued
Total number of ticks spent by connections on this queue	connectionQueueTimeQueued	connection-queue: ticksTotalQueued
Average length of the queue in the last one minute	connectionQueue1MinuteAverage	connection-queue: countQueued1MinuteAverage

Table A–3 (Cont.) Connection Queue Metrics

Metric	Object Name in the SNMP MIB	stats-xml Element: Attribute
Average length of the queue in the last five minutes	connectionQueue5MinuteAverage	connection-queue: countQueued5MinuteAverage
Average length of the queue in the last fifteen minutes	connectionQueue15MinuteAverage	connection-queue: countQueued15MinuteAverage

A.4 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

Metric	Object Name in the SNMP MIB	stats-xml Element: Attribute
Thread pool ID. This refers to the thread-pool under the server element whose ID attribute matches this attribute	threadPoolId	thread-pool: threadPoolId
Number of threads in this pool which were idle at the time of gathering the statistics	NA	thread-pool: countIdleThreads
Number of threads in this pool at the time of gathering the statistics	threadPoolCount	thread-pool: countThreads
Maximum number of requests allowed in the queue	threadPoolMax	thread-pool: maxThreads
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
Maximum number of requests that can be in the queue	threadPoolMax	thread-pool: maxQueued

A.5 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

Metric	Object Name in the SNMP MIB	stats-xml Element: Attribute
Indicates if the DNS cache is enabled	dnsCacheEnabled	dns: flagCacheEnabled
Total number of entries in the cache	dnsCountCacheEntries	dns: countCacheEntries
Maximum number of entries in DNS cache	dnsMaxCacheEntries	dns: maxCacheEntries
Total number of times a cache lookup succeeded	dnsCountCacheHits	dns: countCacheHits

Table A–5 (Cont.) DNS Cache Metrics

Metric	Object Name in the SNMP MIB	stats-xml Element: Attribute
Total number of times a cache lookup failed	dnsCountCacheMisses	dns: countCacheMisses
Indicates whether asynchronous DNS lookups are enabled	dnsAsyncEnabled	dns: flagAsyncEnabled
Number of asynchronous lookups	dnsCountAsyncNameLookups	dns: countAsyncNameLookups
Total number of asynchronous DNS address lookups performed	dnsCountAsyncAddrLookups	dns: countAsyncAddrLookups
Number of asynchronous DNS lookups currently in progress	dnsCountAsyncLookupsInProgress	dns: countAsyncLookupsInProgress

A.6 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

Metric	Object Name in the SNMP MIB	stats-xml Element: Attribute
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
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

A.7 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

Metric	Object Name in the SNMP MIB	stats-xml Element: Attribute
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
Origin server which is processing the current request.	NA	thread: originServer

A.8 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	compressionCountRequests	compression: countRequests
Total number of input bytes for compression	compressionBytesInput	compression: bytesInput
Total number of output bytes after compression	compressionBytesOutput	compression: bytesOutput
Average compression ratio per request	compressionRatioAverage	compression: compressionRatioAverage
Overall compression ratio	compressionRatio	compression: compressionRatio

Table A–8 (Cont.) Compression and Decompression Metrics

Metric	Object Name in the SNMP MIB	stats-xml Element: Attribute
Total number of requests decompressed	decompressionCountRequests	decompression: countRequests
Total number of input bytes for decompression	decompressionBytesInput	decompression: bytesInput
Total number of output bytes after decompression	decompressionBytesOutput	decompression: bytesOutput

A.9 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.

The `virtual-server` section contains these child elements:

- `request-bucket`: contains the statistics pertaining to requests.
- `websocket`: contains the statistics for WebSocket connections.
- `webapp-firewall`: contains the statistics for requests processed by the Web Application Firewall.
- `profile-bucket`: contains the statistics pertaining to the performance profile buckets for the requests serviced by the virtual-server.
- `route`: contains statistics related to Route.

Table A–9 Virtual Server Metrics

Metric	Object Name in the SNMP MIB	stats-xml Element: Attribute
Virtual server identifier. Example: www.charity.org my-vsid	vsId	virtual-server: id
Flag indicating whether the virtual server is enabled or not	vsEnabled	virtual-server: flagEnabled
Hosts whose requests this virtual server services.	vsHosts	virtual-server: host
IP addresses including port where this virtual server listens for requests	vsListeners	virtual-server: listenAddresses
Number of requests processed	vsRequests	request-bucket: countRequests
Number of octets received	vsInOctets	request-bucket: countBytesReceived
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

Table A–9 (Cont.) Virtual Server Metrics

Metric	Object Name in the SNMP MIB	stats-xml Element: Attribute
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	websocketCount UpgradedReques ts	websocket:countUpgradeReq uests
Number of WebSocket requests that were denied upgrade by origin server	websocketCount UpgradeRejected Requests	websocket:countUpgradeReq uestsRejected
Number of WebSocket requests that were denied upgrade by server	websocketCount FailedStrictRequ ests	websocket:countUpgradeReq uestsFailed
Total number of requests that were aborted	websocketCount AbortedRequests	websocket:countRequestsAbo rted
Total number of requests that were closed because of timeout	websocketCount TimeoutRequests	websocket:countRequestsTime dout
Number of bytes received from the clients	websocketCount BytesReceived	websocket:countBytesReceive d
Number of bytes transmitted to the clients	websocketCount BytesTransmitted	websocket:countBytesTransmi tted
Number of active WebSocket connections	websocketCount ActiveConnectio ns	websocket:countActiveConne ctions
Average duration of active time in millisecond	websocketMillise condsConnection ActiveAverage	websocket:millisecondsConne ctionActiveAverage
Total number of requests intercepted by webapp firewall	wafCountInterce ptedRequests	webapp-firewall:countReques tsIntercepted
Total number of requests allowed by webapp firewall (allow action)	wafCountAllowe dRequests	webapp-firewall:countReques tsAllowed
Total number of denied requests (deny action)	wafCountDenied Requests	webapp-firewall:countReques tsDenied

Table A–9 (Cont.) Virtual Server Metrics

Metric	Object Name in the SNMP MIB	stats-xml Element: Attribute
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
The name of the profile bucket. This identifies the profile bucket whose statistics are provided by this element.	NA	profile-bucket:profile
The number of SAF function calls that are profiled in this bucket	NA	profile-bucket:countCalls
The number of requests whose SAF calls were profiled	NA	profile-bucket:countRequests
The number of ticks that were spent outside the function call. That is, the time taken to dispatch the call.	NA	profile-bucket:ticksDispatch
The number of ticks that were spent within the function call	NA	profile-bucket:ticksFunctions
Route identifier	routeName	route:routeId
Condition expression which caused this route to be activated	routeCondition	route:condition

A.10 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

Metric	Object Name in the SNMP MIB	stats-xml Element: Attribute
CPU identifier	cpuId	cpu-info: cpuId
Percentage of the time that the CPU is idle	cpuPercentIdle	cpu-info: cpuPercentIdle
Percentage of the time the CPU is spending in user space	cpuPercentUser	cpu-info: cpuPercentUser
Percentage of the time the CPU is spending in kernel space	cpuPercentKernal	cpu-info: cpuPercentKernel

A.11 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.

The `origin-server-pool` section contains the following child elements:

- `origin-server`: provides statistics pertaining to an origin server.
- `service-queue`: provides statistics for the requests waiting to be processed within an origin server pool.

Table A–11 Origin Server Metrics

Metric	Object Name in the SNMP MIB	stats-xml Element: Attribute
Name of the server pool as specified in server.xml	originServerPoolName	origin-server-pool: name
Number of times a request was retried (to same or different origin server)	originServerPoolCountRetries	origin-server-pool: countRetries
Type of the server pool (`tcp`, `http`).	originServerPoolType	origin-server-pool:type
Type of the server pool (`tcp`, `http`).	originServerName	origin-server: name
Flag indicating whether the origin server is currently marked online	originServerStatus	origin-server: flagOnline
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	originServerTimeOnline	origin-server: secondsOnline
Total number of times the origin server was marked offline	originServerCountDetectedOffline	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 connections closed	originServerCountConnectionsClosed	origin-server: countConnectionsClosed
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
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

Table A–11 (Cont.) Origin Server Metrics

Metric	Object Name in the SNMP MIB	stats-xml Element: Attribute
Total number of health check requests	originServerCountHealthCheckRequests	origin-server: countHealthCheckRequests
Flag indicating whether the node was dynamically discovered	originServerDiscoveredStatus	origin-server: flagDiscovered
Flag indicating whether the node is fully ramped up	originServerRampedupStatus	origin-server: flagRampedUp
Type of origin-server (generic/weblogic/undetected)	originServerType	origin-server: type
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 keep-alive connections closed by the origin server	originServerCountConnectionsClosedByOriginServer	origin-server: countConnectionsClosedByOriginServer
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
Dynamically calculated keep-alive timeout value for the origin server	originServerSecondsKeepAliveTimeout	origin-server: secondsKeepAliveTimeout
Average duration of active time in milliseconds	originServerMillisecondsConnectionActiveAverage	origin-server: millisecondsConnectionActiveAverage
Number of requests currently in the queues	serviceQueueCountQueued	service-queue: countQueued
Number of requests currently in the high priority queue	serviceQueueCountQueuedHighPriority	service-queue: countQueuedHighPriority
Number of requests currently in the normal priority queue	serviceQueueCountQueuedNormalPriority	service-queue: countQueuedNormalPriority
Number of requests currently in the low priority queue	serviceQueueCountQueuedLowPriority	service-queue: countQueuedLowPriority
Total number of request timed out while waiting to be processed	serviceQueueCountQueuedTimedout	service-queue: countQueuedTimedout
Total number of requests added to the queues	serviceQueueCountTotalQueued	service-queue: countTotalQueued
Total number of requests added to the high priority queue	serviceQueueCountTotalQueuedHighPriority	service-queue: countTotalQueuedHighPriority
Total number of requests added to the normal priority queue	serviceQueueCountTotalQueuedNormalPriority	service-queue: countTotalQueuedNormalPriority
Total number of requests added to the low priority queue	serviceQueueCountTotalQueuedLowPriority	service-queue: countTotalQueuedLowPriority

Table A–11 (Cont.) Origin Server Metrics

Metric	Object Name in the SNMP MIB	stats-xml Element: Attribute
Total number of sticky requests added to the queues	serviceQueueCountTotalQueuedSticky	service-queue: countTotalQueuedSticky
Total number of sticky requests converted to non-sticky due to stickiness timeout	serviceQueueCountTotalStickyToNonSticky	service-queue: countTotalStickyToNonSticky
Average duration in milliseconds that high priority requests are in the queue	serviceQueueMilliSecondsHighPriorityAverage	service-queue: millisecondsQueuedHighPriorityAverage
Average duration in milliseconds that normal priority requests are in the queue	serviceQueueMilliSecondsNormalPriorityAverage	service-queue: millisecondsQueuedNormalPriorityAverage
Average duration in milliseconds that low priority requests are in the queue	serviceQueueMilliSecondsLowPriorityAverage	service-queue: millisecondsQueuedLowPriorityAverage

A.12 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

A.13 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

Table A–13 (Cont.) Cache Metrics

Metric	Object Name in the SNMP MIB	stats-xml Element: Attribute
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-bucket: 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

A.14 DMS Metrics Tables

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
instanceName	String
configName	String
instanceVersion	String
timeStarted	Long
secondsRunning	Long
ticksPerSecond	Long
maxProcs	Long
maxThreads	Long
flagProfilingEnabled	Boolean
countRequests	Long
countBytesReceived	Long
countBytesTransmitted	Long
countOpenConnections	Long
count2xx	Long
count3xx	Long
count4xx	Long
count5xx	Long
countOther	Long
count200	Long
count302	Long

Table A–14 (Cont.) DMS Metrics: OTD_Instance Metric Table

Column Name	Type
count304	Long
count400	Long
count401	Long
count403	Long
count404	Long
count503	Long
load1MinuteAverage	Double
load5MinuteAverage	Double
load15MinuteAverage	Double
inOctets	Long
outOctets	Long
req1MinuteAverage	Double
req5MinuteAverage	Double
req15MinuteAverage	Double
err1MinuteAverage	Double
err5MinuteAverage	Double
err15MinuteAverage	Double
responseTime1Minute	Double
responseTime5Minute	Double
responseTime15Minute	Double

Table A–15 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

Table A–15 (Cont.) DMS Metrics: OTD_Process Metric Table

Column Name	Type
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
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

Table A–16 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
countHits	Long
countRevalidationRequests	Long
countRevalidationFailures	Long

Table A–17 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

Table A–17 (Cont.) DMS Metrics: OTD_VirtualServer Metric Table

Column Name	Type
count401	Long
count403	Long
count404	Long
count503	Long
waf.countRequestsIntercepted	Long
waf.countAllowedRequests	Long
waf.countRequestsDenied	Long
waf.countRequestsDropped	Long
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

Table A–18 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

Table A–18 (Cont.) DMS Metrics: OTD_Route Metric Table

Column Name	Type
count4xx	Long
count5xx	Long
countOther	Long
count200	Long
count302	Long
count304	Long
count400	Long
count401	Long
count403	Long
count404	Long
count503	Long

Table A–19 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
instanceName	String
serverPoolName	String
configName	String
serverPoolType	String
countRetries	String
serviceQueue.countQueued	Long
serviceQueue.countQueuedHighPriority	Long
serviceQueue.countQueuedLowPriority	Long
serviceQueue.countQueuedNormalPriority	Long
serviceQueue.countQueuedTimedout	Long
serviceQueue.countTotalQueued	Long
serviceQueue.countTotalQueuedHighPriority	Long
serviceQueue.countTotalQueuedLowPriority	Long
serviceQueue.countTotalQueuedNormalPriority	Long
serviceQueue.countTotalQueuedSticky	Long
serviceQueue.countTotalStickyToNonSticky	Long
serviceQueue.millisecondsQueuedHighPriorityAverage	Double
serviceQueue.millisecondsQueuedLowPriorityAverage	Double
serviceQueue.millisecondsQueuedNormalPriorityAverage	Double

Table A–20 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

Table A–21 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

Table A–22 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
countBytesReceived	Long

Table A–22 (Cont.) DMS Metrics: OTD_TcpProxy Metric Table

Column Name	Type
countBytesTransmitted	Long
millisecondsConnectionActiveAverage	String

Table A–23 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

Table A–24 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

Table A–25 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
countBytesReceived	Long
countBytesTransmitted	Long
countOpenConnections	Long
count2xx	Long

Table A–25 (Cont.) DMS Metrics: OTD_Partition Metric Table

Column Name	Type
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. For information about managing and configuring web application firewall, see Section 10.5, "Managing Web Application Firewalls."Chapter 10.5, "Managing Web Application Firewalls."

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.

B.1 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.

B.2 Rules Against Major Attacks

This section provides information about some rules that are used for preventing major attacks on Web applications.

B.2.1 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

# Initalize 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/2.8 (http://www.modsecurity.org/).
Server: Oracle Traffic Director/12.2.1

--5c4adf36-Z--
```

B.2.2 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] Xkjx6010000000000,0 127.0.0.1 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/2.8 (http://www.modsecurity.org/).
Server: Oracle Traffic Director/12.2.1

--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
```

B.2.3 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'"
```

C
Securing Oracle Traffic Director Deployment

This appendix provides information about the steps that you can take to secure your Oracle Traffic Director deployment.

For information about securing access to the Oracle Traffic Director administration server and enabling SSL/TLS, see Chapter 10, "Managing Security."

C.1 Securing Oracle Traffic Director

The following are some of the steps that you can perform to secure Oracle Traffic Director in your environment:

- Ensure Oracle Traffic Director server instance is running as non-`root` and not listening on all interfaces. For information about starting Oracle Traffic Director instances, see Section 4.3, "Starting, Stopping, and Restarting Oracle Traffic Director Instances."

  > **Note:** For each Oracle Traffic Director configuration that you instantiate on an administration node, a subdirectory named `net-`*`config_name`* is created in the `INSTANCE_HOME` subdirectory.

- Leverage the ability of Oracle Traffic Director to provide high availability as non-`root`. For more information, see Chapter 14, "Configuring Oracle Traffic Director for High Availability."
- Ensure that sufficient file descriptors are available. For more information, see Section 15.2, "Tuning the File Descriptor Limit."Chapter 15.2, "Tuning the File Descriptor Limit."
- Ensure that appropriate network level protections are taken care. For more information, see `http://www./technetwork/articles/servers-storage-admin/secure-linux-env-1841089.html.`

  In addition, you should consider hardening your system. For information about hardening an Oracle Linux system, see `http://www./technetwork/articles/servers-storage-admin/tips-harden-oracle-linux-1695888.html`.

D Oracle Fusion Middleware T2P Utility for Oracle Traffic Director

This appendix describes the Oracle Fusion Middleware T2P utility for Oracle Traffic Director, and contains the following sections:

D.1 Introduction

The Oracle Fusion Middleware T2P utility allows you to move an Oracle Fusion Middleware environment from test to production (T2P) with customization specific to the production environment. This appendix describes support for moving an Oracle Traffic Director environment.

Moving an Oracle Traffic Director installation minimizes the amount of work that would otherwise be required to reapply all the customization and configuration changes made in one environment to another. You can install, configure, customize, and validate Oracle Traffic Director in a test environment. Once the system is stable and performs as required, you can create the production environment by moving a copy of the server and its configuration from the test environment, instead of redoing all the changes that were incorporated into the test environment. If you have an existing production environment, you can move any modifications of the test environment, such as customization, to the production environment.

Moving an Oracle Traffic Director installation from a test to a production environment assumes that the production environment is on the same operating system as the test environment. In addition, the operating system architecture must be the same in both environments. For example, both environment must be running 64-bit operating systems.

For more information on Fusion Middleware Test to Production see: *Moving from a Test to a Production Environment* in *Oracle Fusion Middleware Administering Oracle Fusion Middleware*.

D.2 Overview of the T2P Process

The T2P move of an Oracle Fusion Middleware environment consists of the following two broad steps:

Moving the binaries to the production system.

Moving the configuration to the production system.

- Copy Binary (Oracle Home) - This is the process of creating an archive of the Oracle Home where Oracle Traffic Director is installed.
- Copy Configuration - This is the process of creating an archive of the domain configuration for Oracle Traffic Director.
- Edit Configuration - This is the process of creating an archive of the domain configuration for Oracle Traffic Director.
- Paste Binary (Oracle Home) - This is the process of recreating the binaries at the destination.
- Paste Configuration - This is the process of recreating the configurations at the destination.

The destination configuration of a component can be customized during the T2P move using a document generated during the Copy Configuration phase called a *move plan*.

D.3 Requirements

The following are requirements for the destination host:

- A JDK must be present on destination host in order to execute the T2P scripts.
- The archive created during the Copy Binary/Configuration phase must be accessible.

D.4 Usage of the T2P Utility

The following examples show how to execute commands from the T2P utility.

D.4.1 Copy Binary (Oracle Home)

To create an archive of the Oracle Home including all the files, libraries, and configuration, execute `copyBinary.sh`. After the copy binary operation, created jar file should be copied to the production system.

The following is an example command:

Example D–1 Copy Binary Command

```
./copyBinary.sh -javaHome ./oracle_common/jdk/jre -archiveLoc /tmp/mw.jar
-sourceOracleHomeLoc /scratch/installers/colocated_otd
```

D.4.2 Copy Configuration

To create an archive of the domain configuration (extended for Oracle Traffic Director), execute `copyConfig.sh`. This command introspects all the OTD configurations present in the domain home. Specifically, it introspects the OTD configuration file server.xml, and extracts elements such as `http-listener`, `tcp-listener`, `origin-server-pool`, and `failover-group` from server.xml. You can customize the values of these elements by editing the move plan. After the copy configuration operation, created jar file should be copied to the production system

The following is an example command:

Example D–2 Copy Configuration Command for Collocated OTD

```
./copyConfig.sh -javaHome ./oracle_common/jdk/jre -archiveLoc /tmp/otd.jar
-sourceDomainLoc /scratch/installers/otd_domain -sourceOracleHomeLoc
/scratch/installers/colocated_otd -domainHostName abc1234.example.com
-domainPortNum 7010 -domainAdminUserName weblogic -domainAdminPasswordFile
/scratch/password.txt
```

Note: Make sure that WLS administration server is running, when you perform Appendix D.4.2, "Copy Configuration".

Example D–3 Copy Configuration Command for Standalone OTD

```
./copyConfig.sh -javaHome ./oracle_common/jdk/jre -archiveLoc /tmp/otd.jar
-sourceDomainLoc /scratch/installers/otd_domain -sourceOracleHomeLoc
/scratch/installers/colocated_otd
```

D.4.3 Edit Configuration

To edit a configuration, we need to extract the move plan, execute `extractMovePlan.sh`. For details on customizing the move plan, see Extracting and Customizing an Oracle Traffic Director Move Plan.

The following is an example command:

Example D–4 Extract Move Plan Command

```
./extractMovePlan.sh -javaHome ./oracle_common/jdk/jre -archiveLoc /tmp/otd.jar
-planDirLoc /scratch/moveplan/
```

D.4.4 Paste Binary (Oracle Home)

To recreate the binaries at the destination, execute `pasteBinary.sh`. The following is an example command:

Example D–5 Paste Binary Command

```
./pasteBinary.sh -javaHome ./oracle_common/jdk/jre -archiveLoc /tmp/mw.jar
-targetOracleHomeLoc /scratch/installers/cloned_colocated_otd/
```

D.4.5 Paste Configuration

To recreate the configurations at the destination, execute `pasteConfig.sh`. This command extracts the custom values from the customized move plan, updates the configuration with the value specified in the move plan, and saves the configuration.

The following is an example command:

Example D–6 Paste Configuration Command for Collocated OTD

```
./pasteConfig.sh -javaHome ./oracle_common/jdk/jre -archiveLoc /tmp/otdc.jar
-movePlanLocation /scratch/moveplan/moveplan.xml -targetDomainLoc
/scratch/installers/cloned_domain -targetOracleHomeLoc /scratch/installers/cloned_
```

```
colocated_otd/ -domainAdminPasswordFile /scratch/password.txt
```

Example D–7 Paste Configuration Command for Standalone OTD

```
./pasteConfig.sh -javaHome ./oracle_common/jdk/jre -archiveLoc /tmp/otdc.jar
-movePlanLocation /scratch/moveplan/moveplan.xml -targetDomainLoc
/scratch/installers/cloned_domain -targetOracleHomeLoc /scratch/installers/cloned_
colocated_otd
```

Additional Steps for Moving Oracle Traffic Director:

- Cert Reconfiguration : Certificate can be reconfigured using `otd_setHttpListenerSslProperties/otd_setTcpListenerSslProperties/otd_setVirtualServerSslProperties` or `otd_setOriginServerPoolSslProperties`.
- OAM Reconfiguration : If webgate is configured in the OTD instance and the user wants to re-configure OAM at the target Env in order to use a different OAM instance, then this use case must be handled manually once the pasteConfig is done.

Note: Make sure that all the paths configured for OTD instance are accessible and have proper file permissions on the target environment. For example, server log file path.

For more information on WLST, see *WebLogic Scripting Tool Command Reference for Oracle Traffic Director*

D.5 Extracting and Customizing an Oracle Traffic Director Move Plan

There are few elements in the Oracle Traffic Director server.xml that you can customize using the move plan. These elements are listed below:

Table D–1 Oracle Traffic Director Elements for Customization

Element	Editable Properties
http-listener	ip, port and server-name
tcp-listener	ip and port
failover-group	virtual-ip
origin-server-pool	origin-servers

You should also modify the following properties of the domain:

- Create a new schema for the target domain and update the schema prefix in the move plan, valid only when domain is full JRF.
- Update the `Password file` for node manager element. By default the value is `</value>` which you should modify to `<value>/scratch/T2P/password.txt</value>`. The password file you provide must exist on the file system.

The following in an example Oracle Traffic Director move plan:

```
<movableComponent>
  <componentType>OTD</componentType>
  <componentName>test</componentName>
  <Description>otd component</Description>
  <moveDescriptor>
```

```
<configGroup>
  <type>http-listeners</type>
  <configProperty id="http-listener-1">
    <configProperty>
      <name>Ip</name>
      <value>*</value>
      <itemMetadata>
        <dataType>INTEGER</dataType>
        <scope>READ_WRITE</scope>
      </itemMetadata>
    </configProperty>
    <configProperty>
      <name>Port</name>
      <value>7011</value>
      <itemMetadata>
        <dataType>INTEGER</dataType>
        <scope>READ_WRITE</scope>
      </itemMetadata>
    </configProperty>
    <configProperty>
      <name>ServerName</name>
      <value>abc1234.example.com</value>
      <itemMetadata>
        <dataType>STRING</dataType>
        <scope>READ_WRITE</scope>
      </itemMetadata>
    </configProperty>
  </configProperty>
</configGroup>
<configGroup>
  <type>tcp-listeners</type>
</configGroup>
<configGroup>
  <type>origin-server-pool</type>
  <configProperty id="origin-server-pool-1">
    <configProperty id="origin-server-1">
      <name>origin-server-1</name>
      <value>abc1234.example.com:7010</value>
      <itemMetadata>
        <dataType>STRING</dataType>
        <scope>READ_WRITE</scope>
      </itemMetadata>
    </configProperty>
    <configProperty id="origin-server-2">
      <name>origin-server-2</name>
      <value>abc1234.example.com:7015</value>
      <itemMetadata>
        <dataType>STRING</dataType>
        <scope>READ_WRITE</scope>
      </itemMetadata>
    </configProperty>
  </configProperty>
</configGroup>
<configGroup>
  <type>failover</type>
  <configProperty id="failover-group-1">
    <name>virtual-ip</name>
    <value>10.100.10.1</value>
    <itemMetadata>
      <dataType>STRING</dataType>
```

```
          <scope>READ_WRITE</scope>
        </itemMetadata>
      </configProperty>
    </configGroup>
  </moveDescriptor>
</movableComponent>
</movableComponent>
```

D.6 Logging

The T2P utility puts the logs for the copy and paste phases into the Java `temp` directory (For example: `/tmp/CLONE<Date Time>.log`). The failures and errors are logged into a separate `CLONE<Date Time>.error` file in the format `Error Message`, `Cause` and `Action`. You can specify a different location for the logs using the command line argument `-logDirLoc`.