Oracle® Fusion Middleware

Administrator's Guide for Oracle HTTP Server

11*g* Release 1 (11.1.1)

E10144-02

October 2009

Oracle Fusion Middleware Administrator's Guide for Oracle HTTP Server, 11*g* Release 1 (11.1.1)

E10144-02

Contributors: Jeff Schieli, Shail Goel, John Janosik, Padraic Russell, Mark Nelson, Pushkar Kapasi, Rashmi Badan, Amitabh Shukla, Rajasekhar Balannagari, Ravi Prakash Ummadi

Contents

Preface

This guide describes how to manage Oracle HTTP Server, including how to start and stop Oracle HTTP Server, how to manage network components, configure listening ports, and extend basic functionality using modules.

Audience

Oracle Fusion Middleware Administrator's Guide for Oracle HTTP Server is intended for application server administrators, security managers, and managers of databases used by application servers. This documentation is based on the assumption that readers are already familiar with the Apache software.

Documentation Accessibility

Our goal is to make Oracle products, services, and supporting documentation accessible to all users, including users that are disabled. To that end, our documentation includes features that make information available to users of assistive technology. This documentation is available in HTML format, and contains markup to facilitate access by the disabled community. Accessibility standards will continue to evolve over time, and Oracle is actively engaged with other market-leading technology vendors to address technical obstacles so that our documentation can be accessible to all of our customers. For more information, visit the Oracle Accessibility Program Web site at `http://www.oracle.com/accessibility/`.

Accessibility of Code Examples in Documentation

Screen readers may not always correctly read the code examples in this document. The conventions for writing code require that closing braces should appear on an otherwise empty line; however, some screen readers may not always read a line of text that consists solely of a bracket or brace.

Accessibility of Links to External Web Sites in Documentation

This documentation may contain links to Web sites of other companies or organizations that Oracle does not own or control. Oracle neither evaluates nor makes any representations regarding the accessibility of these Web sites.

Deaf/Hard of Hearing Access to Oracle Support Services

To reach Oracle Support Services, use a telecommunications relay service (TRS) to call Oracle Support at 1.800.223.1711. An Oracle Support Services engineer will handle technical issues and provide customer support according to the Oracle service request process. Information about TRS is available at `http://www.fcc.gov/cgb/consumerfacts/trs.html`, and a list of phone numbers is available at `http://www.fcc.gov/cgb/dro/trsphonebk.html`.

Related Documents

For more information, see the following documents in the Oracle Fusion Middleware 11*g* Release 1 (11.1.1) documentation set:

- *Oracle Fusion Middleware Concepts*
- *Oracle Fusion Middleware Administrator's Guide*
- *Oracle Fusion Middleware High Availability Guide*
- Apache documentation included in this library

Note: Readers using this guide in PDF or hard copy formats will be unable to access third-party documentation, which Oracle provides in HTML format only. To access the third-party documentation referenced in this guide, use the HTML version of this guide and click the hyperlinks.

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

Understanding Oracle HTTP Server

This part presents introductory and conceptual information about Oracle HTTP Server. It contains the following chapters:

- Chapter 1, "Introduction to Oracle HTTP Server"
- Chapter 2, "Management Tools for Oracle HTTP Server"
- Chapter 3, "Understanding Oracle HTTP Server Modules"

1

Introduction to Oracle HTTP Server

Oracle HTTP Server is the Web server component for Oracle Fusion Middleware. It provides a listener for Oracle WebLogic Server and the framework for hosting static pages, dynamic pages, and applications over the Web.

This chapter includes the following sections:

- Section 1.1, "What is Oracle HTTP Server"
- Section 1.2, "Understanding Oracle HTTP Server Directory Structure"
- Section 1.3, "Understanding Configuration Files"
- Section 1.4, "Oracle HTTP Server Support"

1.1 What is Oracle HTTP Server

Oracle HTTP Server 11*g*, Release 1 (11.1.1.2.0) is based on Apache 2.2.13 infrastructure, and includes modules developed specifically by Oracle. The features of single sign-on, clustered deployment, and high availability enhance the operation of the Oracle HTTP Server. Oracle HTTP Server has the following components to handle client requests:

- HTTP listener, to handle incoming requests and route them to the appropriate processing utility.
- Modules (mods), to implement and extend the basic functionality of Oracle HTTP Server. Many of the standard Apache modules are included with Oracle HTTP Server. Oracle also includes several modules that are specific to Oracle Fusion Middleware to support integration between Oracle HTTP Server and other Oracle Fusion Middleware components.
- Perl interpreter, a persistent Perl runtime environment embedded in Oracle HTTP Server through mod_perl.

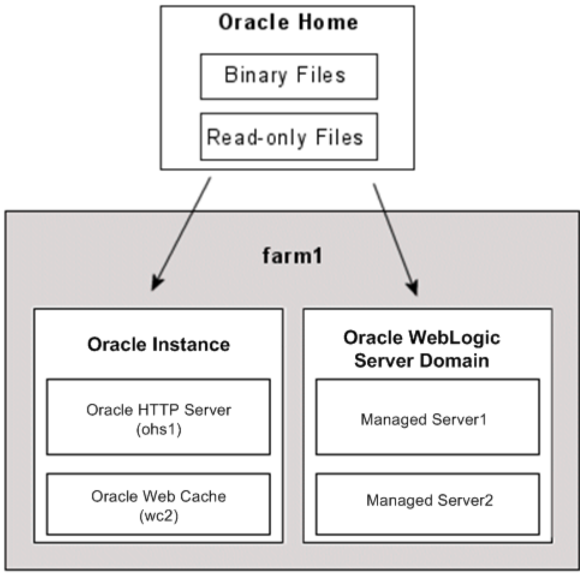
Oracle HTTP Server enables developers to program their site in a variety of languages and technologies, such as the following:

- Perl (through mod_perl and CGI)
- C (through CGI and FastCGI)
- C++ (through FastCGI)
- PHP (through mod_php)
- Oracle PL/SQL

Oracle HTTP Server can also be a proxy server, both forward and reverse. A reverse proxy enables content served by different servers to appear as if coming from one server.

Figure 1–1shows an Oracle home with an Oracle instance and an Oracle WebLogic Server domain. Throughout this book, examples will use the components in this figure. The farm (farm1) consists of an Oracle instance and a WebLogic Server domain. The Oracle instance includes Oracle HTTP Server (ohs1) and Oracle Web Cache (wc1), and the WebLogic Server domain includes two Managed Servers.

Figure 1–1 Oracle Fusion Middleware Farm



See Also: *Oracle Fusion Middleware Concepts*

1.1.1 Key Features of Oracle HTTP Server

The following sections describe some of the key features of Oracle HTTP Server:

- Section 1.1.1.1, "Security: Encryption with Secure Sockets Layer"
- Section 1.1.1.2, "Security: Single Sign-On"
- Section 1.1.1.3, "Distributed Authoring and Versioning (DAV) Support"
- Section 1.1.1.4, "URL Rewriting and Proxy Server Capabilities"
- Section 1.1.1.5, "Oracle Process Manager and Notification Server"
- Section 1.1.1.6, "Oracle Plug-Ins"
- Section 1.1.1.7, "PL/SQL Server Pages"
- Section 1.1.1.8, "Server-Side Includes"
- Section 1.1.1.9, "Perl"
- Section 1.1.1.10, "PHP"
- Section 1.1.1.11, "C / C++ (CGI and FastCGI)"
- Section 1.1.1.12, "Load Balancing"

1.1.1.1 Security: Encryption with Secure Sockets Layer

Secure Sockets Layer (SSL) is required to run any Web site securely. Oracle HTTP Server supports SSL encryption based on patented, industry standard, algorithms. SSL works seamlessly with commonly-supported Internet browsers. Security features include the following:

- SSL hardware acceleration support uses dedicated hardware for SSL with nCipher. Hardware encryption is faster than software encryption.
- Variable security per directory allows individual directories to be protected by different strength encryption.
- Oracle HTTP Server and Oracle WebLogic Server communicate using the HTTP protocol to provide both encryption and authentication. You can also enable HTTP tunneling for the T3 or IIOP protocols to provide non-browser clients access to WebLogic Server services.

See Also: *Oracle Fusion Middleware Security Guide*

1.1.1.2 Security: Single Sign-On

Basic authentication for HTTP servers uses a flat file with encrypted passwords. Oracle HTTP Server supports standard authentication as well as single sign-on. The mod_osso module is included to support single sign-on across sites and across applications. This security feature provides a better end user experience because users only have to log in once. It also helps the development cycle because most of the security is declarative.

See Also:

- *Oracle Fusion Middleware Security Guide*
- Section 3.7, "mod_osso"

1.1.1.3 Distributed Authoring and Versioning (DAV) Support

WebDAV is an HTTP based protocol that allows DAV enabled clients, such as Microsoft Office and Microsoft Windows Explorer, to edit files on a server. Oracle HTTP Server enhances DAV support with the mod_oradav module. This module enables WebDAV clients to connect to an Oracle database, read and write content, query, and lock documents in various schemas.

1.1.1.4 URL Rewriting and Proxy Server Capabilities

Active Web sites usually update their Web pages and directory contents often, and possibly their URLs as well. Oracle HTTP Server makes it easy to accommodate the changes by including an engine that supports URL rewriting so end users do not have to change their bookmarks.

Oracle HTTP Server also supports reverse proxy capabilities, making it easier to make content served by different servers to appear from one single server.

1.1.1.5 Oracle Process Manager and Notification Server

Oracle Fusion Middleware provides a high availability infrastructure integration with Oracle Process Manager and Notification Server (OPMN), for process management, failure detection, and failover for Oracle HTTP Server processes.

See Also:

- *Oracle Fusion Middleware High Availability Guide*
- *Oracle Fusion Middleware Oracle Process Manager and Notification Server Administrator's Guide*

1.1.1.6 Oracle Plug-Ins

Oracle makes the following plug-ins available to enable Microsoft Internet Information Server (IIS) to work with Oracle HTTP Server:

- Oracle Proxy Plug-In is a separately-available component that enables a third-party HTTP listener to route requests to Oracle HTTP Server. Users can benefit from Oracle HTTP Server features even when their corporate standard requires them to use, for example, Microsoft IIS. The proxy plug-in provides Oracle HTTP Server features, such as single sign-on and load balancing, to be accessed when using a third-party HTTP listener.
- Oracle SSO Plug-In is a separately-available component that enables a third-party HTTP listener to be integrated with Oracle Fusion Middleware Single Sign-on. For example, Microsoft IIS listener applications can be protected by using the Oracle Single Sign-on infrastructure, as well as basic directory structure. Users can be authenticated to these listeners using a single sign-on password. This functionality is similar to what the mod_osso module provides to Oracle HTTP Server.

1.1.1.7 PL/SQL Server Pages

PL/SQL Server Pages are similar in concept to the JavaServer Pages. The mod_plsql module enables PL/SQL to be used as the scripting language within an HTML page. PL/SQL Server Pages get translated into a stored procedure, which then uses the module to send the output to the browser.

1.1.1.8 Server-Side Includes

Server-Side Includes provide an easy way of adding dynamic or uniform static content across all pages on a site. It is typically used for header and footer information. Oracle HTTP Server supports special directives to enable these only for certain types of files, or for specified virtual hosts.

1.1.1.9 Perl

Perl is a scripting language often used to provide dynamic content. Perl scripts can either be called as a CGI program, or directly through the mod_perl module. Oracle Fusion Middleware uses Perl version 5.10.

See Also: Section 3.8, "mod_perl"

1.1.1.10 PHP

PHP (Hypertext Preprocessor) is a scripting language capable of being embedded in HTML, which makes it well-suited for Web development. Although the mod_php module does not ship with Oracle Fusion Middleware, Oracle provides detailed instructions on how to install and use PHP with Oracle HTTP Server. For more information, see Using PHP with Oracle HTTP Server 11g R1.

1.1.1.11 C / C++ (CGI and FastCGI)

CGI programs are commonly used to program Web applications. Oracle HTTP Server enhances the programs by providing a mechanism to keep them alive beyond the request lifecycle.

1.1.1.12 Load Balancing

Oracle HTTP Server includes the mod_wl_ohs module, which routes requests to Oracle WebLogic Server. The mod_wl_ohs module provides the same load balancing functionality as the Oracle WebLogic Server Plug-in for Apache HTTP Server (mod_weblogic).

For more information on the mod_wl_ohs module's load balancing capability with Oracle WebLogic Server, see "The Dynamic Server List" section of *Using Web Server Plug-Ins with Oracle WebLogic Server*.

Note: The mod_wl_ohs module provides the same functionality as the Oracle WebLogic Server Plug-in for Apache HTTP Server (mod_weblogic), except for the minor differences described in "mod_wl_ohs" on page 3-33.

1.2 Understanding Oracle HTTP Server Directory Structure

Oracle HTTP Server directories are divided between the Oracle home and the Oracle instance. The Oracle home directories are read-only, and contain the Oracle Fusion Middleware binaries. The Oracle instance directories contain the modules and content pages for Oracle HTTP Server. Table 1–1 shows the subdirectories for Oracle HTTP Server in the Oracle home directory.

Table 1–1 Oracle Home Directories

Directory	Contents
`ohs/bin`	Oracle HTTP Server binary files.
`ohs/conf`	Oracle HTTP Server template configuration files, which get provisioned to an Oracle instance when an Oracle HTTP Server component is configured. **Note:** These files should only be edited by advanced Oracle HTTP Server users.
`ohs/modules`	Oracle HTTP Server modules

Table 1–2 shows the subdirectories for Oracle Fusion Middleware in the Oracle instance directory.

Table 1–2 Oracle Instance Directories

Directory Name	Contents
`config/OHS/<ohs_name>`	Oracle HTTP Server configuration files.
`config/OHS/<ohs_name>/htdocs`	Static content and CGI scripts for Oracle HTTP Server.
`config/OHS/<ohs_name>/moduleconf`	Configuration files that are automatically included in Oracle HTTP Server configuration. Be careful not to create any files with a `.conf` extension in this directory that you do not want to be included in the configuration.
`diagnostics/logs/OHS/<ohs_name>`	Oracle HTTP Server component instance log files.

1.3 Understanding Configuration Files

Configuration for Oracle HTTP Server are specified through **directives** in configuration files in the exact same manner as Apache HTTP Server configuration files. For more information about Apache HTTP Server configuration files, see the Apache HTTP Server 2.2 Users Guide.

1.4 Oracle HTTP Server Support

Oracle provides technical support for the following Oracle HTTP Server features and conditions:

- Modules included in the Oracle distribution. Oracle does not support modules obtained from any other source, including the Apache Software Foundation. Oracle HTTP Server will still be supported when non-Oracle-provided modules are included. If it is suspected that the non-Oracle-provided modules are contributing to reported problems, customers may be requested to reproduce the problems without including those modules.
- Problems that can be reproduced within an Apache configuration consisting only of supported Oracle Apache modules.
- Use of the included Perl interpreter within the supported Apache configuration.

2 Management Tools for Oracle HTTP Server

Oracle provides the following management tools for Oracle HTTP Server:

- Fusion Middleware Control, which is a browser-based management tool
- `opmnctl`, which is a command-line management tool.

This chapter includes the following sections:

2.1 Overview of Oracle HTTP Server Management

The main tool for managing Oracle HTTP Server is Fusion Middleware Control, which is a browser-based tool for administering and monitoring the Oracle Fusion Middleware environment.

See Also: *Oracle Fusion Middleware Administrator's Guide*

2.2 Accessing Fusion Middleware Control

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

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

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.

1. Display Fusion Middleware Control by entering the URL in your Web browser. For example:

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

   The Welcome page is displayed:

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

The default user name for the administrator user is `weblogic`. This is the account you can use to log in to the Fusion Middleware Control for the first time. The `weblogic` password is the one you supplied during the installation of Fusion Middleware Control.

2.3 Accessing the Oracle HTTP Server Home Page

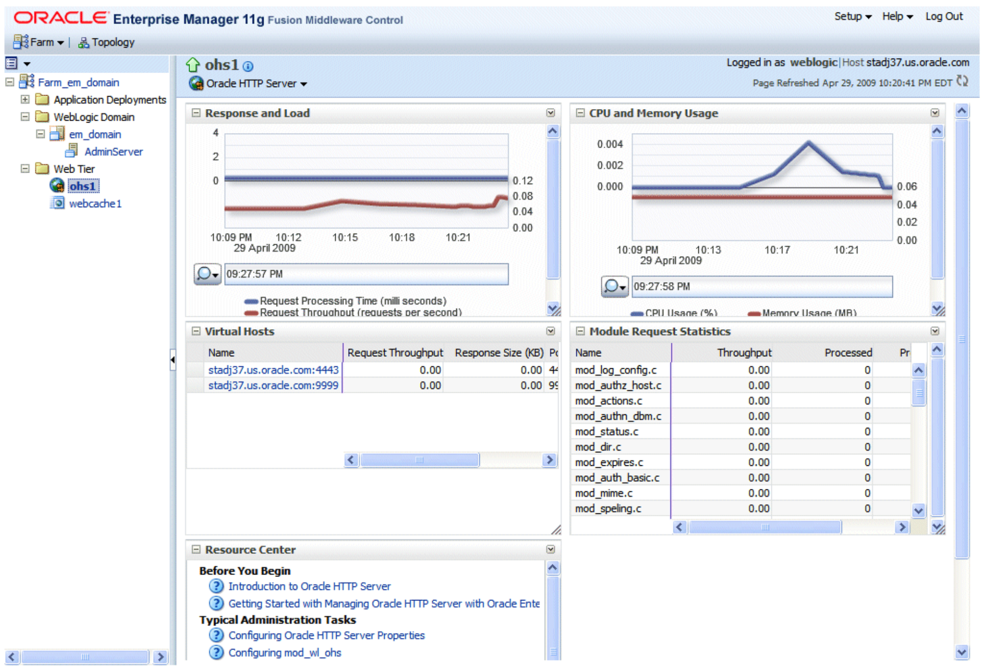
The Oracle HTTP Server Home page in Fusion Middleware Control contains menus and regions that enable you to manage the server. Use the menus for monitoring, managing, routing, and viewing general information.

2.3.1 Navigating Within Fusion Middleware Control

When you select a target, such as a WebLogic Managed Server or a component, such as Oracle HTTP Server, the target's home page is displayed in the content pane and that target's menu is displayed at the top of the page, in the context pane. For example, if you select an Oracle HTTP Server instance from the Web Tier folder, the Oracle HTTP Server menu is displayed. You can also view the menu for a target by right-clicking the target in the navigation pane.

Figure 2–1 shows the target navigation pane and the home page of Oracle HTTP Server.

Figure 2–1 Oracle HTTP Server Home Page


ORACLE Enterprise Manager 11g Fusion Middleware Control
Setup
Help
Log Out
Farm
Topology
Farm_em_domain
Application Deployments
WebLogic Domain
em_domain
AdminServer
Web Tier
ohs1
webcache1
ohs1
Oracle HTTP Server
Logged in as weblogic | Host stadj37.us.oracle.com
Page Refreshed Apr 29, 2009 10:20:41 PM EDT
Response and Load
10:09 PM 10:12 10:15 10:18 10:21
29 April 2009
09:27:57 PM
Request Processing Time (milli seconds)
Request Throughput (requests per second)
CPU and Memory Usage
10:09 PM 10:13 10:17 10:21
29 April 2009
09:27:58 PM
CPU Usage (%)
Memory Usage (MB)
Virtual Hosts
Module Request Statistics
Resource Center

Name	Request Throughput	Response Size (KB)	Po
stadj37.us.oracle.com:4443	0.00	0.00	44
stadj37.us.oracle.com:9999	0.00	0.00	99

Name	Throughput	Processed	Pr
mod_log_config.c	0.00	0	
mod_authz_host.c	0.00	0	
mod_actions.c	0.00	0	
mod_authn_dbm.c	0.00	0	
mod_status.c	0.00	0	
mod_dir.c	0.00	0	
mod_expires.c	0.00	0	
mod_auth_basic.c	0.00	0	
mod_mime.c	0.00	0	
mod_speling.c	0.00	0	

Before You Begin
- Introduction to Oracle HTTP Server
- Getting Started with Managing Oracle HTTP Server with Oracle Ente

Typical Administration Tasks
- Configuring Oracle HTTP Server Properties
- Configuring mod_wl_ohs

The Oracle HTTP Server home page contains the following regions:

- Response and Load Region: Provides information such as the number of active requests, how many requests were submitted, and how long it took for Oracle HTTP Server to respond to a request. It also provides information about the number of bytes processed with the requests.
- CPU and Memory Usage Region: Shows how much CPU (by percentage) and memory (in megabytes) are being used by an Oracle HTTP Server instance.

- Virtual Hosts Region: Shows the virtual hosts for Oracle HTTP Server.
- Module Request Statistics Region: Shows the modules for Oracle HTTP Server.
- Resource Center: Provides links to books and topics related to Oracle HTTP Server.

See Also: The *Oracle Fusion Middleware Administrator's Guide* gives detailed descriptions of all the items on the target navigation pane and the home page.

2.4 Using the opmnctl Command-line Tool

You can use the `opmnctl` command-line interface to start and stop system components, monitor system components, and perform many other tasks related to process management. Here are some of the `opmnctl` operations that you can perform with Oracle HTTP Server components:

- Create additional components.
- Delete existing components.
- Start and/or stop components.
- Check a component's status.
- Check a component's port usage.

See Also: *Oracle Process Manager and Notification Server*

`opmnctl` is located in the `ORACLE_HOME/opnm/bin` directory in an Oracle Home and in the `ORACLE_INSTANCE/bin` directory in an Oracle Instance.

Note: Oracle highly recommends using the `opmnctl` from the same `ORACLE_INSTANCE` that Oracle HTTP Server is running in, unless that instance is unavailable.

The following example demonstrates using opmnctl in a command shell to start an Oracle HTTP Server component, and to then verify the status of that component.

```
$ opmnctl startproc ias-component=ohs1
$ opmnctl status
Processes in Instance: instance1
---------------------------------+--------------------+---------+---------
ias-component                    | process-type       |     pid | status
---------------------------------+--------------------+---------+---------
webcache1                        | WebCache-admin     |   19556 | Alive
webcache1                        | WebCache           |   19555 | Alive
ohs1                             | OHS                |    7249 | Alive
```

3 Understanding Oracle HTTP Server Modules

Modules (mods) extend the basic functionality of Oracle HTTP Server, and support integration between Oracle HTTP Server and other Oracle Fusion Middleware components.

This chapter discusses the modules developed specifically by Oracle for Oracle HTTP Server. It includes the following sections:

- Section 3.1, "List of Included Modules"
- Section 3.2, "mod_certheaders"
- Section 3.3, "mod_dms"
- Section 3.4, "mod_onsint"
- Section 3.5, "mod_oradav"
- Section 3.6, "mod_ossl"
- Section 3.7, "mod_osso"
- Section 3.8, "mod_perl"
- Section 3.9, "mod_plsql"
- Section 3.10, "mod_security"
- Section 3.11, "mod_wl_ohs"

3.1 List of Included Modules

This section lists all of the modules bundled with Oracle HTTP Server.

Oracle-developed Modules for Oracle HTTP Server

The following modules have developed specifically by Oracle for Oracle HTTP Server:

- mod_certheaders
- mod_dms
- mod_onsint
- mod_oradav
- mod_ossl
- mod_osso
- mod_plsql
- mod_wl_ohs

Base Apache and Third-party Modules in Oracle HTTP Server

Oracle HTTP Server also includes the following base Apache and third-party modules out-of-the-box. These modules are *not* developed by Oracle.

> **See Also:** For information about Apache modules, refer to the Apache documentation.

- mod_access
- mod_actions
- mod_alias
- mod_asis
- mod_auth
- mod_auth_anon
- mod_auth_dbm
- mod_autoindex
- mod_cern_meta
- mod_cgi
- mod_define
- mod_digest
- mod_dir
- mod_env
- mod_example
- mod_expires
- mod_fastcgi
- mod_headers
- mod_imap
- mod_include
- mod_info
- mod_log_agent
- mod_log_config
- mod_log_referer
- mod_mime
- mod_mime_magic
- mod_mmap_static
- mod_negotiation
- mod_perl
- mod_proxy
- mod_rewrite
- mod_security

3.2 mod_certheaders

The mod_certheaders module enables reverse proxies that terminate Secure Sockets Layer (SSL) connections in front of Oracle HTTP Server to transfer information regarding the SSL connection, such as SSL client certificate information, to Oracle HTTP Server and the applications running behind Oracle HTTP Server. This information is transferred from the reverse proxy to Oracle HTTP Server using HTTP headers. The information is then transferred from the headers to the standard CGI environment variable. The mod_ossl module or the mod_ssl module populate the variable if the SSL connection is terminated by Oracle HTTP Server.

The mod_certheaders module also enables certain requests to be treated as HTTPS requests even though they are received through HTTP. This is done using the `SimulateHttps` directive.

`SimulateHttps` takes the container it is contained within, such as `<VirtualHost>` or `<Location>`, and treats all requests received for this container as if they were received through HTTPS, regardless of the real protocol used by the request.

3.3 mod_dms

The mod_dms module enables you to monitor the performance of site components using Oracle Dynamic Monitoring Service (DMS).

3.4 mod_onsint

The mod_onsint module provides integration support with Oracle Notification Service (ONS) and Oracle Process Manager and Notification Server (OPMN). It is an Oracle module and provides the following functionality.

- Provides a subscription mechanism for ONS notifications within Oracle HTTP Server. mod_insint receives notification for all modules within an Oracle HTTP Server instance.
- Publishes `PROC_READY ONS` notifications so that OPMN knows that the listener is up and ready. It also provides information such as DMS metrics and information about how the listener can be contacted. These notifications are sent periodically by mod_onsint as long as the Oracle HTTP Server instance is running.

Mod_onsint runs as a thread within a Oracle HTTP Server parent process on UNIX and within a child process on Windows. This thread is responsible for sending and receiving ONS messages.

There is an optional directive called `OpmnHostPort` that can be configured for mod_onsint. This directive enables you to specify a hostname and port that OPMN should use for pinging the Oracle HTTP Server instance that mod_onsint is running in. If `OpmnHostPort` is not specified, mod_onsint chooses an HTTP port automatically. In certain circumstances, you may want to choose a specific HTTP port and hostname that OPMN should use to ping the listener with.

`OpmnHostPort` has the following syntax that specifies the values to pass to OPMN:

```
OpmnHostPort [<http> | <https>://]<host>:<port>
```

For example, the following line would specify that OPMN should use HTTP, the localhost interface and port 7778 to ping this listener:

```
OpmnHostPort http://localhost:7778
```

3.5 mod_oradav

The mod_oradav module is an Oracle Call Interface (OCI) application written in C that extends the implementation of mod_dav. The mod_oradav directive can read and write to local files or to an Oracle database. The Oracle database must have an OraDAV driver (a stored procedure package) for the mod_oradav module to map WebDAV activity to database activity. Essentially the mod_oradav module enables WebDAV clients to connect to an Oracle database, read and write content, and query and lock documents in various schemas.

You can configure the mod_oradav module using standard Oracle HTTP Server directives. Use the Advanced Configuration page of Fusion Middleware Control to configure the mod_oradav module. The mod_oradav directive can immediately leverage other module code (such as mime_magic) to perform content management tasks. Most OraDAV processing activity involves streaming content to and from a content provider. The mod_oradav directive uses OCI streaming logic directly within Oracle HTTP Server.

See Also:

- Chapter 9, "Configuring mod_oradav"
- *Oracle Portal Administrator's Guide*
- For information about using the mod_oradav module to access database schemas for access by third-party tools, such as Adobe GoLive, Macromedia Dreamweaver, and Oracle *inter*Media, go to the OraDAV information available on OTN at:

  `http://www.oracle.com/technology/index.html`

3.6 mod_ossl

The mod_ossl module enables strong cryptography for Oracle HTTP Server. This Oracle module is a plug-in to Oracle HTTP Server that enables the server to use SSL. It is very similar to the OpenSSL module, mod_ssl. The mod_ossl module is based on the Oracle implementation of SSL, which supports SSL version 3 and TLS version 1, and is based on Certicom and RSA Security technology.

See Also: *Oracle Fusion Middleware Security Guide*

3.7 mod_osso

The mod_osso module enables **single sign-on** for Oracle HTTP Server by examining incoming requests and determining whether the requested resource is protected. If it is, then it retrieves the Oracle HTTP Server cookie.

The module is disabled, by default. To enable the mod_osso module, follow the instructions in Section 4.4.5, "Enabling the mod_osso Module".

See also: For information about forced authentication, see *Oracle Fusion Middleware Application Developer's Guide for Oracle Identity Management*.

For information about single sign-on, see *Oracle Fusion Middleware Security Guide*

3.8 mod_perl

The mod_perl module embeds the Perl interpreter into Oracle HTTP Server. This eliminates start-up overhead and enables you to write modules in Perl. Oracle Fusion Middleware uses Perl version 5.10.

The module is disabled, by default. To enable the mod_perl module, follow the instructions in Section 4.4.3, "Configuring the mod_perl Module".

See Also: mod_perl Guide

3.8.1 Using mod_perl with a Database

This section provides information for mod_perl users working with databases. It explains how to test a local database connection and set character forms.

3.8.1.1 Using Perl to Access the Database

Perl scripts access databases using the DBI/DBD driver for Oracle. The DBI/DBD driver is part of Oracle Fusion Middleware. It calls Oracle Call Interface (OCI) to access the databases.

Once mod_perl is enabled, DBI must be enabled in the `mod_perl.conf` file to function. To enable DBI, perform the following steps:

1. Edit the `mod_perl.conf` file:
   a. In Fusion Middleware Control, navigate to the Oracle HTTP Server Advanced Configuration page.
   b. Select the `mod_perl.conf` file from the menu and click **Go**.
   c. Add the following line to the `mod_perl.conf` file:

      ```
      PerlModule Apache::DBI
      ```
2. Click **Apply** to save the file.
3. Restart Oracle HTTP Server using Fusion Middleware Control.

Place the Perl scripts that you want to run in the `*ORACLE_INSTANCE*/config/OHS/<ohs_name>/cgi-bin` directory.

Example 3–1 Using a Perl Script to Access a Database

```
#!ORACLE_HOME/perl/bin/perl -w
  use DBI;
  my $dataSource = "host=hostname.domain;sid=orclsid;port=1521";
  my $userName = "userid";
  my $password = "password";
  my $dbhandle = DBI->connect("dbi:Oracle:$dataSource", $userName, $password)
    or die "Can't connect to the Oracle Database: $DBI::errstr\n";
  print "Content-type: text/plain\n\n";
  print "Database connection successful.\n";
  ### Now disconnect from the database
  $dbhandle->disconnect
    or warn "Database disconnect failed; $DBI::errstr\n";
  exit;
```

To run the DBI scripts, the URLs would look like the following:

```
http://hostname.domain:port/cgi-bin/scriptname
http://hostname.domain:port/perl/scriptname
```

If a script specifies "`use Apache::DBI`" instead of "`use DBI`", then it can only run from the URL `http://`*`hostname.domain:port`*`/perl/`*`scriptname`*.

3.8.1.2 Testing a Database Connection

Example 3–2 shows a sample Perl script for testing a database connection. Replace the instance name, userid, and password in the `connect` statement with proper values for the target database.

Example 3–2 Sample Perl Script For Testing Connection for Local Seed Database

```
use DBI;
print "Content-type: text/plain\n\n";
$dbh = DBI->connect("dbi:Oracle:instance_name", userid/password, "") ||
            die $DBI::errstr;
$stmt = $dbh->prepare("select * from emp order by empno")|| die $DBI::errstr;
$rc = $stmt->execute() || die $DBI::errstr;
while (($empno, $name) = $stmt->fetchrow()) {
   print "$empno $name\n";
}
warn $DBI::errstr if $DBI::err;
die "fetch error: " . $DBI::errstr if $DBI::err;
$stmt->finish() || die "can't close cursor";
$dbh->disconnect() || die "cant't log off Oracle";
```

3.8.1.3 Using SQL NCHAR Data Types

SQL NCHAR data types (NCHAR, NVARCHAR2 and NCLOB) are reliable Unicode data types. SQL NCHAR data types enable you to store Unicode characters regardless of the database character set. The character set for those data types is specified by the national character set, which is either AL16UTF-16 or UTF8.

Example 3–3 shows an example of accessing SQL NCHAR data.

Example 3–3 Sample Script to Access SQL NCHAR Data

```
# declare to use the constants for character forms
use DBD::Oracle qw(:ora_forms);
# connect to the database and get the database handle
$dbh = DBI->connect( ... );

# prepare the statement and get the statement handle
$sth = $dbh->prepare( 'SELECT * FROM TABLE_N WHERE NCOL1 = :nchar1' );

# bind the parameter of a NCHAR type
$sth->bind_param( ':nchar1', $param_1 );
# set the character form to NCHAR
$sth->func( { ':nchar1' => ORA_NCHAR } , 'set_form' );

$sth->execute;
```

As shown in Example 3–3, the `set_form` function is provided as a private function that you can invoke with the standard DBI `func` method. The `set_form` function takes an anonymous hash that enables you to set the character form for parameters.

The valid values of character form are either `ORA_IMPLICIT` or `ORA_NCHAR`. Setting the character form to `ORA_IMPLICIT` causes the application's bound data to be converted to the database character set, and `ORA_NCHAR` to the national character set. The default is `ORA_IMPLICIT`.

The constants are available as `ora_forms` in `DBD::Oracle`.

`set_default_form` sets the default character form for a database handle. The following example shows its syntax:

```
# specify the default form to be NCHAR
$dbh->func( ORA_NCHAR, 'set_default_form' );
```

This syntax causes the form of all parameters to be `ORA_NCHAR`, unless otherwise specified with `set_form` calls. Unlike the `set_form` function, the `set_default_form` functions on the database handle, so every statement from the database handle has the form of your choice.

Example 3–4   Sample for set_form

```
# a declaration example for the constants ORA_IMPLICIT and ORA_NCHAR
use DBD::Oracle qw(:ora_forms);

# set the character form for the placeholder :nchar1 to NCHAR
$sth->func( { ':nchar1' => ORA_NCHAR } , 'set_form' );

# set the character form using the positional index
$sth->func( { 2 => ORA_NCHAR } , 'set_form' );

# set the character form for multiple placeholders at once
$sth->func( { 1 => ORA_NCHAR, 2 => ORA_NCHAR } , 'set_form' );
```

3.9 mod_plsql

The mod_plsql module connects Oracle HTTP Server to an Oracle database, enabling you to create Web applications using Oracle stored procedures.

To access a Web-enabled PL/SQL application, configure a PL/SQL **database access descriptor** (DAD) for the mod_plsql module. A DAD is a set of values that specifies how the module connects to a database server to fulfill an HTTP request. Besides the connection details, a DAD contains important configuration parameters for various operations in the database and for the mod_plsql module in general. Any Web-enabled PL/SQL application which makes use of the PL/SQL Web ToolKit needs to create a DAD to invoke the application.

3.9.1 Creating a DAD

Perform the following steps to create a DAD:

1. Edit the `dads.conf` configuration file.

   See Table 3–1 for mod_plsql configuration file locations.

2. Add a DAD where the DAD has the following format:

   a. The Oracle HTTP Server `<Location>` directive which defines a virtual path used to access the PL/SQL Web Application. This directive groups a set of directives that apply to the named Location.

      For example, the directive `<Location /myapp>` defines a virtual path called `/myapp` that will be used to invoke a PL/SQL Web application through a URL such as `http://host:port/myapp/`.

Note: Earlier releases of the mod_plsql module were always mounted on a virtual path with a prefix of `/pls`. This restriction is removed in later releases but might still be a restriction imposed by some of the earlier PL/SQL applications.

b. The Oracle HTTP Server `SetHandler` directive that directs Oracle HTTP Server to enable the mod_plsql module to handle the request for the virtual path defined by the named Location:

```
SetHandler pls_handler
```

c. Additional Oracle HTTP Server directives that are allowed in the context of a `<Location>` directive. Typically, the following directives are used:

```
Order deny,allow
Allow from all
AllowOverride None
```

d. One or more specific mod_plsql directives. For example:

```
PlsqlDatabaseUsername        scott
PlsqlDatabasePassword        tiger
PlsqlDatabaseConnectString   orcl
PlsqlAuthenticationMode      Basic
```

e. An Oracle HTTP Server `</Location>` directive which closes the group of directives for the named Location, and defines a single DAD.

3. Save the edits.
4. Obfuscate the DAD password by running the `dadTool.pl` script located in the `ORACLE_HOME/bin` directory.

   See Also: "PlsqlDatabasePassword" on page 3-19 for instructions on performing the obfuscation.

5. Restart Oracle HTTP Server using Fusion Middleware Control.

You can create additional DADs by defining other uniquely named Locations in `dads.conf`.

Example DADs

The following DAD connects as a specific user and has a default home page:

```
<Location /pls/mydad>
SetHandler pls_handler
Order allow,deny
Allow from All
AllowOverride None
PlsqlDatabaseUsername scott
PlsqlDatabasePassword tiger
PlsqlDatabaseConnectString prod_db
PlsqlDefaultPage scott.myapp.home
</Location>
```

The following DAD uses HTTP Basic Authentication and supports document upload/download operations:

```
<Location /pls/mydad2>
```

```
SetHandler pls_handler
Order allow,deny
Allow from All
AllowOverride None
PlsqlDatabaseConnectString prod_db2
PlsqlDefaultPage scott.myapp.my_home
PlsqlDocumentTablename scott.my_documents
PlsqlDocumentPath docs
PlsqlDocumentProcedure scott.docpkg.process_download
</Location>
```

3.9.2 Configuration Files for mod_plsql

The mod_plsql configuration parameters reside in the configuration files that are located in the *ORACLE_INSTANCE* directory, as described in Table 3–1.

Table 3–1 mod_plsql Configuration Files In an Oracle Instance

Directory Name	Contents
`config/OHS/<ohs_name>/moduleconf`	`plsql.conf` configuration file.
`config/OHS/<ohs_name>/mod_plsql`	`dads.conf` and `cache.conf` configuration files.

The mod_plsql configuration parameters are described in these sections:

- Section 3.9.2.1, "plsql.conf"
- Section 3.9.2.2, "dads.conf"
- Section 3.9.2.3, "cache.conf"

3.9.2.1 plsql.conf

The `plsql.conf` file resides in the *ORACLE_INSTANCE*`/config/OHS/<ohs_name>/moduleconf` directory and Oracle HTTP Server automatically loads all `.conf` files under this location. The `plsql.conf` file contains the `LoadModule` directive to load the mod_plsql module into Oracle HTTP Server, any global settings for the mod_plsql module, and include directives for `dads.conf` and `cache.conf`.

See Also: The `plsql.README` file, located in `ORACLE_HOME/ohs/mod_plsql`, for a detailed description of `plsql.conf`

The following parameters are used with the `plsql.conf` file:

- PlsqlDMSEnable
- PlsqlLogEnable
- PlsqlLogDirectory
- PlsqlIdleSessionCleanupInterval

PlsqlDMSEnable

Enables Dynamic Monitoring Service (DMS) for the mod_plsql module.

Category	Value
Syntax	`PlsqlDMSEnable {On	Off}`

Category	Value
Default	`On`
Example	`PlsqlDMSEnable On`

PlsqlLogEnable

Enables debug level logging for the mod_plsql module. Debug level logging is meant to be used for debugging purposes only.

When logging is enabled, Oracle HTTP Server log files are typically created in the *`ORACLE_INSTANCE`*`/diagnostics/logs/OHS/<ohs_name>` directory. However, the location specified in PlsqlLogDirectory determines the final location.

This parameter should be set to `Off` unless recommended by Oracle support to debug problems with the mod_plsql module.

To view more details about the internal processing of the mod_plsql module, set this directive to `On`. This causes the mod_plsql module to start logging every request that is processed. The log files are generated as specified by the PlsqlLogDirectory directive.

Category	Value
Syntax	`PlsqlLogEnable {On | Off}`
Default	`Off`
Example	`PlsqlLogEnable Off`

PlsqlLogDirectory

Specifies the directory where debug level logs are written.

Set the directory name of the location where log files should be generated when logging is enabled. To avoid possible confusion about the location of this directory, an absolute path is recommended.

On UNIX, this directory must have write permissions by the owner of the child `httpd` processes.

Category	Value
Syntax	`PlsqlLogDirectory` *`directory`*
Default	None
Example	`PlsqlLogDirectory ORACLE_ INSTANCE/diagnostics/logs/OHS/<ohs_name>`

PlsqlIdleSessionCleanupInterval

Specifies the time (in minutes) in which the idle database sessions should be closed and cleaned by the mod_plsql module.

This directive is used in conjunction with connection pooling of database connections and sessions in the mod_plsql module. When a session is not used for the specified amount of time, it is closed and freed. This is done so that unused sessions can be cleaned, and the memory is freed on the database side.

Setting this time to a low number helps in faster cleanup of unused database sessions. If this number is too low, then this may adversely affect the performance benefits of connection pooling in the mod_plsql module.

If the number of open database sessions is not a concern, you can increase the value of this parameter for best performance. In such a case, if the site is accessed frequently enough that the idle session cleanup interval is never reached for a session, then the DAD configuration parameter PlsqlMaxRequestsPerSession can be modified so that it is guaranteed that a pooled database session gets recycled on a regular basis.

For most installations, the default value is adequate.

Category	Value
Syntax	`PlsqlIdleSessionCleanupInterval` *`number`*
Default	15 (minutes)
Example	`PlsqlIdleSessionCleanupInterval 10`

3.9.2.2 dads.conf

The `dads.conf` file contains the configuration parameters for the PL/SQL database access descriptor. (See Table 3–1 for the file location.) A DAD is a set of values that specifies how the mod_plsql module connects to a database server to fulfill a HTTP request.

The following parameters are used with the `dads.conf` file:

- PlsqlAfterProcedure
- PlsqlAlwaysDescribeProcedure
- PlsqlAuthenticationMode
- PlsqlBeforeProcedure
- PlsqlBindBucketLengths
- PlsqlBindBucketWidths
- PlsqlCGIEnvironmentList
- PlsqlConnectionTimeout
- PlsqlConnectionValidation
- PlsqlDatabaseConnectString
- PlsqlDatabasePassword
- PlsqlDatabaseUserName
- PlsqlDefaultPage
- PlsqlDocumentPath
- PlsqlDocumentProcedure
- PlsqlDocumentTablename
- PlsqlErrorStyle
- PlsqlExclusionList
- PlsqlFetchBufferSize
- PlsqlInfoLogging
- PlsqlMaxRequestsPerSession
- PlsqlNLSLanguage

- PlsqlPathAlias
- PlsqlPathAliasProcedure
- PlsqlRequestValidationFunction
- PlsqlSessionCookieName
- PlsqlSessionStateManagement
- PlsqlTransferMode
- PlsqlUploadAsLongRaw

PlsqlAfterProcedure

Specifies the procedure to be invoked after calling the requested procedure. This enables you to put a hook point after the requested procedure is called. This is useful in doing SQL*Traces/SQL Profiles while debugging a problem with the requested procedure. This is also useful when you want to ensure that a specific call is made after running every procedure.

Category	Value
Syntax	`PlsqlAfterProcedure` *`string`*
Default	None
Example	`PlsqlAfterProcedure portal.mypkg.myafterproc`

- This parameter should only be used for debugging purposes. In addition, you could use this parameter to stop SQL trace/SQL profiling.

PlsqlAlwaysDescribeProcedure

Specifies whether the mod_plsql module should describe a procedure before trying to run it. If this is set to `On`, then the mod_plsql module will always describe a procedure before invoking it. Otherwise, the mod_plsql module will only describe a procedure when its internal heuristics have interpreted a parameter type incorrectly.

Category	Value
Syntax	`PlsqlAlwaysDescribeProcedure {On | Off}`
Default	`Off`
Example	`PlsqlAlwaysDescribeProcedure On`

- This parameter should only be used for debugging purposes.

PlsqlAuthenticationMode

Specifies the authentication mode to use for allow access through the DAD.

Category	Value
Syntax	`PlsqlAuthenticationMode {Basic | SingleSignOn | GlobalOwa | CustomOwa | PerPackageOwa}`
Default	`Basic`
Example	`PlsqlAuthenticationMode CustomOwa`

- If the DAD is not using the Basic authentication, then you must include a valid username/password in the DAD configuration. For the Basic mode, to perform dynamic authentication, the DAD username/password parameters must be omitted.
- The SingleSignOn mode is supported only for Oracle Fusion Middleware releases, and is used by Oracle Portal and Oracle Single Sign-On. Most customer applications use Basic authentication. Custom authentication modes (GlobalOwa, CustomOwa, and PerPackageOwa) are used by very few PL/SQL applications.

PlsqlBeforeProcedure

Specifies the procedure to be invoked before calling the requested procedure. This enables you to put a hook point before the requested procedure is called. This is useful in doing SQL*Traces/SQL Profiles while debugging a problem with the requested procedure. This is also useful when you want to ensure that a specific call be made before running every procedure.

Category	Value
Syntax	`PlsqlBeforeProcedure` *`string`*
Default	None
Example	`PlsqlBeforeProcedure portal.mypkg.mybeforeproc`

- This parameter should only be used for debugging purposes. In addition, you could use this parameter to start SQL Trace/SQL Profiling.

PlsqlBindBucketLengths

Note: This configuration property is rarely ever changed, and system defaults suffice in almost all cases.

Specifies the rounding size to use while binding the number of elements in a collection bind. While executing PL/SQL statements, the Oracle database maintains a cache of PL/SQL statements in the shared SQL area, and attempts to reuse the cached statement if the same statement is run again. Oracle's matching criteria requires that the statement texts be identical, and that the bind variable data types match. Unfortunately, the type match for strings is sensitive to the exact byte size specified, and for collection bindings is also sensitive to the number of elements in the collection. Since the mod_plsql module binds statements dynamically, the odds of hitting the shared cache are low, and it may fill up with near-duplicates and lead to contention for the latch on the shared area. This parameter reduces that effect by bucketing bind lengths to the nearest level.

All numbers specified should be in ascending order. After the last specified size, subsequent bucket sizes will be assumed to be twice the last one.

Category	Value
Syntax	`PlsqlBindBucketLengths` *`number multiline`*
Default	4,20,100,400

Category	Value

Example	`PlsqlBindBucketLengths 4`
	`PlsqlBindBucketLengths 25`
	`PlsqlBindBucketLengths 125`

- This parameter is relevant only if you are using procedures with array parameters, and passing varying number of parameters to the procedure.
- The default should be sufficient for most PL/SQL applications.
- To see if this parameter needs to be changed, check the number of versions of a SQL statement in the SQL area.
- After the higher configured value, mod_plsql starts auto-generating bucket sizes of larger values by doubling the last value, as needed. Therefore, after 400, the next bucket value becomes 800, then 1600, and so on.
- Consider using flexible parameter passing to reduce the problem.

PlsqlBindBucketWidths

Note: This configuration property is rarely ever changed, and system defaults suffice in almost all cases.

Specifies the rounding size to use while binding the number of elements in a collection bind. While executing PL/SQL statements, the Oracle database maintains a cache of PL/SQL statements in the shared SQL area, and attempts to reuse the cached statement if the same statement is run again. Oracle's matching criteria requires that the statement texts be identical, and that the bind variable data types match. Unfortunately, the type match for strings is sensitive to the exact byte size specified, and for collection bindings is also sensitive to the number of elements in the collection. Since the mod_plsql module binds statements dynamically, the odds of hitting the shared cache are low, and it may fill up with near-duplicates and lead to contention for the latch on the shared area. This parameter reduces that effect by bucketing bind widths to the nearest level.

All numbers specified should be in ascending order. After the last specified size, subsequent bucket sizes will be assumed to be twice the last one.

The last bucket width must be equal to or less than 4000. This is due to the restriction imposed by OCI where array bind widths cannot be greater than 4000.

Category	Value
Syntax	`PlsqlBindBucketWidths` *number multiline*
Default	`32,128,1450,2048,4000`

Example	`PlsqlBindBucketWidths 40`
	`PlsqlBindBucketWidths 400`
	`PlsqlBindBucketWidths 2000`

- This parameter is relevant only if you are using procedures with array parameters, and passing varying number of parameters to the procedure.
- The default should be sufficient for most PL/SQL applications.

- To see if this parameter needs to be changed, check the number of versions of a SQL statement in the SQL area.
- After the higher configured value, mod_plsql starts auto-generating bucket sizes of larger values by doubling the last value, as needed. Therefore, after 400, the next bucket value becomes 800, then 1600, and so on.
- Consider using flexible parameter passing to reduce the problem.

PlsqlCGIEnvironmentList

Specifies overrides and additions of CGI environment variables to the default set of environment variables passed to a PL/SQL procedure. This is a multi-line directive of name-value pairs to be added, overridden or removed. You can only specify one environment variable for each directive.

You can add CGI environment variables from the Oracle HTTP Server environment by specifying the variable name. To remove a CGI environment variable, set it equal to blank. To add your own name-value pair, use the syntax myname=myvalue.

Category	Value
Syntax	`PlsqlCGIEnvironmentList string` *`multiline`*
Default	None

Example	■ To add a new environment variable from the Oracle HTTP Server environment:
	`PlsqlCGIEnvironmentList DOCUMENT_ROOT`
	■ To remove an environment variable:
	`PlsqlCGIEnvironmentList MYENVAR2=`
	■ To override from the Oracle HTTP Server environment:
	`PlsqlCGIEnvironmentList REQUEST_PROTOCOL=HTTPS`
	■ To add your own environment variable:
	`PlsqlCGIEnvironmentList MY_VARNAME=MY_VALUE`

- Environment variables added here are available in the PL/SQL application through the function `owa_util.get_cgi_env`.

PlsqlConnectionTimeout

Specifies the timeout in milliseconds for testing a connection pool in the mod_plsql module.

When PlsqlConnectionValidation is set to `Automatic` or `AlwaysValidate`, the mod_plsql module attempts to test pooled database connections. This parameter specifies the maximum time the mod_plsql module should wait for the test request to complete before it assumes that the connection is not usable.

Category	Value
Syntax	`PlsqlConnectionTimeout` *`number`*
Default	10000 (milliseconds)
Example	`PlsqlConnectionTimeout 5000`

PlsqlConnectionValidation

Specifies the mechanism the mod_plsql module should use to detect terminated connections in its connection pool.

Note: This configuration property is rarely ever changed, and system defaults suffice in almost all cases.

For performance reasons, the mod_plsql module pools database connections. If a database instance goes down, and the mod_plsql module was maintaining a pool of connections to the instance, then each pooled database connection results in an error when it is next used to service a request. This can be a concern in high availability configurations such as RAC where even if one node goes down, other nodes servicing the database might have been able to service the request successfully. The mod_plsql module provides for a mechanism whereby it can self-correct after it detects a failure that could be caused by a database node going down. This mechanism to self-correct is controlled by the parameter PlsqlConnectionValidation.

The following are the valid values for PlsqlConnectionValidation:

- `Automatic`: The mod_plsql module tests all pooled database connections which were created prior to the detection of a failure that could mean an instance failure.
- `ThrowAwayOnFailure`: The mod_plsql module throws away all pooled database connections which were created prior to the detection of a failure that could mean an instance failure.
- `AlwaysValidate`: The mod_plsql module always tests all pooled database connections which were created prior to issuing a request. Since this option has an associated performance overhead for each request, this should be used with caution.
- `NeverValidate`: The mod_plsql module never pings any pooled database connection.

Category	Value
Syntax	`PlsqlConnectionValidation {Automatic | ThrowAwayOnFailure | AlwaysValidate | NeverValidate}`
Default	`Automatic`
Example	`PlsqlConnectionValidation ThrowAwayOnFailure`

When the mod_plsql module encounters one of the following errors, it assumes that the database may have been down.

- `00443 – background process <string> did not start`
- `00444 – background process <string> failed while starting`
- `00445 – background process did not start after <x> seconds`
- `00447 – fatal error in background processes`
- `00448 – normal completion of background process`
- `00449 – background process <string> unexpectedly terminated with error`
- `00470 – LGWR process terminated with error`

- `00471 — DBWR process terminated with error`
- `00472 — PMON process terminated with error`
- `00473 — ARCH process terminated with error`
- `00474 — SMON process terminated with error`
- `00475 — TRWR process terminated with error`
- `00476 — RECO process terminated with error`
- `00480 — LCK* process terminated with error`
- `00481 — LMON process terminated with error`
- `00482 — LMD* process terminated with error`
- `00484 — LMS* process terminated with error`
- `00485 — DIAG process terminated with error`
- `01014 — ORACLE shutdown in progress`
- `01033 — ORACLE initialization or shutdown in progress`
- `01034 — ORACLE not available`
- `01041 — internal error. hostdef extension doesn't exist`
- `01077 — background process initialization failure`
- `01089 — immediate shutdown in progress- no operations permitted`
- `01090 — shutdown in progress- connection is not permitted`
- `01091 — failure during startup force`
- `01092 — ORACLE instance terminated. Disconnection forced`
- `03106 — fatal two-task communication protocol error`
- `03113 — end-of-file on communication channel`
- `03114 — not connected to ORACLE`
- `12570 — TNS: packet reader failure`
- `12571 — TNS: packet writer failure`

PlsqlDatabaseConnectString

Specifies the connection to an Oracle database.

Category	Value
Syntax	`PlsqlDatabaseConnectString` *`string`* `{ServiceNameFormat | SIDFormat | TNSFormat | NetServiceNameFormat}` The *string* parameter depends on the second argument: ■ If the second argument is `ServiceNameFormat`, *string* is *`HOST:PORT:SERVICE_NAME`*, where *`HOST`* is the host name running the database, *`PORT`* is the port number the TNS listener is listening at, and *`SERVICE_NAME`* is the database service name. An IPv6 address can be specified using the format *[IPv6_ ADDRESS]:PORT:SERVICE_NAME*. ■ If the second argument is `SIDFormat`, *string* is *HOST:PORT:SID* where *`HOST`* is the host name running the database, *`PORT`* is the port number the TNS listener is listening at, and *`SID`* is the database SID. An IPv6 address can be specified using the format *[IPv6_ ADDRESS]:PORT:SID*. ■ If the second argument is `TNSFormat`, *string* is a valid TNS alias that can be resolved using Net8 utilities like tnsping and SQL*Plus. ■ If the second argument is `NetServiceNameFormat`, *string* is a valid net service name that can be resolved to a connect descriptor. A connect descriptor is a specially formatted description of the destination for a network connection. A connect descriptor contains destination service and network route information. If the format argument is not specified, then the mod_plsql module assumes the *string* is either in the HOST:PORT:SID format, or resolvable by Net8. The differentiation between the two is made by the presence of the colon in the specified string. It is recommended that newer DADs do not use the `SIDFormat` syntax. This exists only for backward compatibility reasons. Use the new two argument format for newly created DADs.
Default	None
Example	■ `PlsqlDatabaseConnectString example.com:1521:myhost.iasdb.inst ServiceNameFormat` ■ `PlsqlDatabaseConnectString [2001:DB8:f1ff:f1ff]:1521:myhost.iasdb.inst ServiceNameFormat` ■ `PlsqlDatabaseConnectString example.com:1521:iasdb SIDFormat` ■ `PlsqlDatabaseConnectString [2001:DB8:ff1ff:f1ff]:1521:iasdb SIDFormat` ■ `PlsqlDatabaseConnectString myhost_tns TNSFormat` ■ `PlsqlDatabaseConnectString cn=oracle,cn=iasdb NetServiceNameFormat` ■ `PlsqlDatabaseConnectString (DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(Host=`*`example.com`*`)(Port= 1521))(CONNECT_DATA=(SID=iasdb))) TNSFormat` ■ `PlsqlDatabaseConnectString myhost_tns` ■ `PlsqlDatabaseConnectString example.com:1521:iasdb`

- If the database is running in the same Oracle home, or the environment variable TWO_TASK is set, then this parameter need not be specified.
- If the database is running in a separate Oracle home, then this parameter is mandatory.

- If you have problems connecting to the database:
  - Check the username and password information in the DAD.
  - Make sure that you run tnsping db_connect_string, and commands such as:

    ```
    sqlplus DADUsername/DADPassword@db_connect_string
    ```

  - Ensure that TNS_ADMIN is configured properly.
  - Verify that the HOST:PORT:SERVICE_NAME format works correctly.
  - Ensure that the TNS listener and database are up and running.
  - Ensure that you can ping the host from this machine.
- From a the mod_plsql module perspective, `TNSFormat` and `NetServiceNameFormat` are synonymous and denote connect descriptors that are resolved by Net8. The `TNSFormat` is provided as a convenience so that end-users use this to signify that the name resolution happens through the local tnsnames.ora. For situations where the resolution is through an LDAP lookup as configured in sqlnet.ora, it is recommended that the format specifier of `NetServiceNameFormat` be used.

  If your database supports high availability, for example, Oracle Real Application Clusters database, it is highly recommended that you use the `NetServiceNameFormat` such that the resolution for the net service name is through LDAP. This enables you to add or remove RAC nodes accessible through the mod_plsql module by changing Oracle Internet Directory with the new or deleted node information. In such situations, hard-coding database listener HOST:PORT information in dads.conf or in the local `tnsnames.ora` is not recommended.

PlsqlDatabasePassword

Specifies the password to use to log in to the database.

Category	Value
Syntax	`PlsqlDatabasePassword` *string*
Default	None
Example	`PlsqlDatabasePassword tiger`

Notes:

- This is a mandatory parameter, except for a DAD that sets PlsqlAuthenticationMode to Basic and uses dynamic authentication.
- For DADs using SingleSignOn authentication, this parameter uses the name of the schema owner.

After making manual configuration changes to DAD passwords, it is recommended that the DAD passwords are obfuscated by running the `dadTool.pl` script located in `ORACLE_HOME/bin`.

To obfuscate DAD passwords:

1. If necessary, change the user to the Oracle software owner user, typically `oracle`, using the following command:

   ```
   $ su - oracle
   ```

2. Set the *ORACLE_HOME* environment variable to specify the path to the Oracle home directory for the current release, and set the PATH environment variable to include the directory containing the Perl executable and the location of the `dadTool.pl` script.

   Bourne, Bash, or Korn shell:

   ```
   $ ORACLE_HOME=new_ORACLE_HOME_path;export ORACLE_HOME
   $ PATH=ORACLE_HOME/bin:ORACLE_HOME/perl/bin:$PATH;export PATH
   ```

   C or tcsh shell:

   ```
   % setenv ORACLE_HOME new_ORACLE_HOME_PATH
   % setenv PATH ORACLE_HOME/bin:ORACLE_HOME/perl/bin:PATH
   ```

   On Microsoft Windows, set the PATH and PERL5LIB environment variable:

   ```
   set PATH=ORACLE_HOME\bin;ORACLE_HOME\perl\bin;%PATH%
   set PERL5LIB=ORACLE_HOME\perl\lib
   ```

3. On UNIX platforms, set the shared library path environment variable.

   Include the `ORACLE_HOME/lib` or `lib32` directory in your shared library path. Table 3–2 shows the appropriate directory and environment variable for each platform.

Table 3–2 Shared Library Path Environment Variable

Platform	Environment Variable	Include Directory
AIX Based Systems	LIBPATH	ORACLE_HOME/lib
HP-UX PA-RISC	SHLIB_PATH	ORACLE_HOME/lib
Solaris Operating System	LD_LIBRARY_PATH	ORACLE_HOME/lib32
Other UNIX platforms, including Linux and HP Tru64 UNIX	LD_LIBRARY_PATH	ORACLE_HOME/lib

   For example, on HP-UX PA-RISC systems, set the SHLIB_PATH environment to include the `ORACLE_HOME/lib` directory:

   ```
   $SHLIB_PATH=$ORACLE_HOME/lib:$SHLIB_PATH;export SHLIB_PATH
   ```

4. Change directory to the mod_plsql configuration directory for the current release of Oracle HTTP Server:

   ```
   cd ORACLE_HOME/bin
   ```

5. Invoke the following Perl script to obfuscate DAD password:

   ```
   perl dadTool.pl -f dadfilename
   ```

   where *`dadfilename`* is the filename for `dads.conf`, which includes the full path to the DAD file.

   For example:

   ```
   perl dadTool.pl -f /u01/app/oracle/as11gr1/ORACLE_INSTANCE/config/OHS/<ohs_
   name>/mod_plsql/dads.conf
   ```

PlsqlDatabaseUserName

Specifies the username to use to log in to the database.

Category	Value
Syntax	`PlsqlDatabaseUsername` *`string`*
Default	None
Example	`PlsqlDatabaseUsername scott`

- This is a mandatory parameter, except for a DAD that sets PlsqlAuthenticationMode to Basic and uses dynamic authentication.
- For DADs using SingleSignOn authentication, this parameter is the name of the schema owner.

PlsqlDefaultPage

Specifies the default procedure to call if none is specified in the URL.

Category	Value
Syntax	`PlsqlDefaultPage` *`string`*
Default	None
Example	`PlsqlDefaultPage myschema.mypackage.home`

- You can also use Oracle HTTP Server Rewrite rules to achieve the same effect as you get by setting this configuration parameter.

PlsqlDocumentPath

Specifies a virtual path in the URL that initiates document download from the document table. For example, if this parameter is set to `docs`, then the following URLs will start the document downloading process for URLs of the format:

```
/pls/dad/docs
/pls/plsqlapp/docs
```

Category	Value
Syntax	`PlsqlDocumentPath` *`string`*
Default	`docs`
Example	`PlsqlDocumentPath docs`

- Omit this parameter for applications that do not perform document uploads or downloads.

PlsqlDocumentProcedure

Specifies the procedure to call when a document download is initiated. This procedure is called to process the download.

Category	Value
Syntax	`PlsqlDocumentProcedure` *`string`*
Default	None

Category	Value
Example	`PlsqlDocumentProcedure portal.wwdoc_process.process_download`

- Omit this parameter for applications that do not perform document uploads or downloads.

PlsqlDocumentTablename

Specifies the table in the database to which all documents are uploaded.

Category	Value
Syntax	`PlsqlDocumentTablename` *`string`*
Default	None
Example	`PlsqlDocumentTablename myschema.document_table`

- Omit this parameter for applications that do not perform document uploads or downloads.

PlsqlErrorStyle

Specifies the error reporting mode for mod_plsql errors.

Category	Value

Syntax	`PlsqlErrorStyle {ApacheStyle | ModplsqlStyle | DebugStyle}`
	■ `ApacheStyle`: The mod_plsql module indicates to Oracle HTTP Server the HTTP error that was encountered. Oracle HTTP Server then generates the error page. This can be used with the Oracle HTTP Server `ErrorDocument` directive to produce customized error messages.
	■ `ModplsqlStyle`: The mod_plsql module generates the error pages, usually a short message indicating the PL/SQL error encountered and PL/SQL exception stack, if any. For example:
	`scott.foo PROCEDURE NOT FOUND`
	■ `DebugStyle`: This mode provides more details than `ModplsqlStyle`. The mod_plsql module provides more details about the URL and parameters, and also produces server configuration information. This mode is for debugging purposes only. Do not use this in a production system, since displaying internal server variables could be a security risk.

Default	`ApacheStyle`
Example	`PlsqlErrorStyle ModplsqlStyle`

PlsqlExclusionList

Specifies a pattern for procedures, packages, or schema names which are forbidden to be directly run from a browser. This is a multi-line directive in which each pattern is on a separate line. The pattern is not case sensitive and can accept a wildcard such as an asterisk (*). The default patterns disallowed from direct URL access are as follows:

- sys.*

- dbms_*
- utl_*
- owa_util*
- owa.*
- htp.*
- htf.*
- wpg_docload.*

Setting this directive to `#NONE#` will disable all protection. This is strongly discouraged for an active site and should not be done. It may be used for debugging purposes.

If this parameter is overridden, the defaults still apply, which means that you do not have to explicitly add the default list to the list of excluded patterns.

Category	**Value**
Syntax	`PlsqlExclusionList {string | "#NONE#" multiline}`
Default	sys.* dbms_* utl_* owa_util* owa.* htp.* htf.* wpg_docload.*

Example	`PlsqlExclusionList myschema.private.*`
	`PlsqlExclusionList myschema.private1.*`
	will disallow access to URLs which contain one of:
	`sys.*,dbms_*,utl_*,owa_util*, owa.*, htp.*,htf.*, wpg_docload.*,myschema.private.*, myschema.private1.*`
	`PlsqlExclusionList "#NONE#"`
	will disable all protection. Its use is strongly discouraged for an active site.

- In addition to the patterns specified with this parameter, the mod_plsql module disallows any procedure name which contains the following special characters:
  - tabs
  - new lines
  - carriage-return
  - single quotation mark
  - reverse slash
  - form feed
  - left parenthesis
  - right parenthesis

- space

This cannot be changed.

PlsqlFetchBufferSize

Specifies the number of rows of content to fetch from the database for each trip, using either owa_util.get_page or owa_util.get_page_raw.

By default, the mod_plsql module attempts to fetch 200 response lines of output where each line is of 255 bytes. In situations where the response bytes are single-bytes, the response buffer is populated to the maximum and can pack 255*200=51000 bytes for each round trip. For responses containing multi-byte data, the byte packing for each row could be less than ideal resulting in lesser bytes getting transferred for each round trip. If your application generates large pages frequently and the response does not fit in one round trip, then consider setting this parameter higher. The memory usage for the mod_plsql module will increase.

Category	Value
Syntax	`PlsqlFetchBufferSize` *`number`*
Default	200
Example	`PlsqlFetchBufferSize 256`

- This parameter is changed only for performance reasons. The minimum value for this parameter is 28, but it is seldom reduced.
- Change this parameter only under the following circumstances:
  - The average response page is large and you want to reduce the number of round-trips the mod_plsql module makes to the database to fetch the response.
  - The character set in use is multi-byte, and you want to compensate for the problem of get_page or get_page_raw fetching fewer bytes for each row. Calculations in the PL/SQL Web ToolKit are character-based and in the case of multi-byte characters, OWA packages assume a worst-case character byte size and do not attempt to pack each row to its maximum.

PlsqlInfoLogging

Specifies what mode the mod_plsql module should use to do extra performance logging.

InfoDebug mode: This logs more information to the Apache's error_log. This is used in conjunction with Apache's info logging level. If the Apache's logging level is not at least set to this high, this setting will be ignored.

Category	Value
Syntax	`PlsqlInfoLogging` *`InfoDebug`*
Default	Empty
Example	`PlsqlInfoLogging InfoDebug`

The logging setting is useful for debugging problems in your PL/SQL application.

PlsqlMaxRequestsPerSession

Specifies the maximum number of requests a pooled database connection should service before it is closed and re-opened.

Category	Value
Syntax	`PlsqlMaxRequestsPerSession` *`number`*
Default	1000
Example	`PlsqlMaxRequestsPerSession 500`

- This parameter helps relieve memory and resource problems that may occur due to prolonged session reuse by a PL/SQL application.
- This parameter should not need to be changed. The default is sufficient in most cases.
- Setting this parameter to a low number can degrade performance. A case for a lower value might be an infrequently-used DAD whose performance is not a concern, and for which limiting the number of requests provides some benefit.

PlsqlNLSLanguage

Specifies the NLS_LANG variable for this DAD. This parameter overrides the NLS_ LANG environment variable. When this parameter is set, the PL/SQL Gateway uses the specified NLS_LANG to connect to the database. Once connected, an alter session command is issued to switch to the specified language and territory. If the middle tier character set matches that of the database, then no alter session call is issued by the mod_plsql module.

Category	Value
Syntax	`PlsqlNLSLanguage` *`string`*
Default	None
Example	`PlsqlNLSLanguage America_America.UTF8`

- Most applications have PlsqlTransferMode set to CHAR which means that the character set in PlsqlNLSLanguage needs to match the character set of the database. In one special case, where the database and the mod_plsql module are both using fixed-size character sets, and the character set width matches, the character set can be different. The response character set is always the mod_plsql module character set.
- If PlsqlTransferMode is set to RAW, then this parameter can be ignored.

PlsqlPathAlias

Specifies a virtual path alias to map to a procedure call. This is application-specific. This directive is used with PlsqlPathAliasProcedure.

Category	Value
Syntax	`PlsqlPathAlias` *`string`*
Default	None
Example	`PlsqlPathAlias url`

- For applications that do not use path aliasing, this parameter may be omitted.

PlsqlPathAliasProcedure

Specifies the procedure to call when the virtual path in the URL matches the path alias as configured by PlsqlPathAlias.

Category	Value
Syntax	`PlsqlPathAliasProcedure` *`string`*
Default	None
Example	`PlsqlPathAliasProcedure portal.wwpth_api_alias.process_download`

- For applications that do not use path aliasing, this parameter may be omitted.

PlsqlRequestValidationFunction

Specifies an application-defined PL/SQL function which gives you the opportunity to allow and disallow further processing of the requested procedure. This is useful in implementing tight security for your PL/SQL application by blocking out package and procedure calls that should not be allowed to run from a DAD.

The function defined by this parameter must have the following prototype:

```
boolean function_name (procedure_name IN varchar2)
```

The *procedure_name* parameter will contain the name of the procedure that the request is trying to run.

For example, if all the PL/SQL application procedures callable from a browser are inside the package mypkg, then an implementation of this function can be as follows:

```
boolean my_validation_check (procedure_name varchar2)
is
begin
  if (upper (procedure_name) like upper ('myschema.mypkg%')) then
    return TRUE
  else
    return FALSE
  end if;
end;
```

Category	Value
Syntax	`PlsqlRequestValidationFunction` *`string`*
Default	none
Example	`PlsqlRequestValidationFunction myschema.mypkg.my_validation_check`

- By default, the mod_plsql module already disallows direct URL access to certain schemas and packages. For more information, refer to PlsqlExclusionList.
- It is highly recommended that you provide an implementation for this function such that it only allows requests that belong to your application, and are callable from a browser.

- Since this function will be called for every request, be sure to make this function as optimized as possible. Suggested recommendations are:
  - Name your PL/SQL packages in a fashion such that the implementation of this function can be similar to the previous example.
  - If your implementation performs a table lookup to determine what packages and procedures should be allowed, then performance can be improved if you pin the cursor in the shared pool.

PlsqlSessionCookieName

Specifies the cookie name when PlsqlAuthenticationMode is set to SingleSignOn. This parameter is supported only for Oracle Fusion Middleware releases, and is used by Oracle Portal and Oracle Single Sign-On.

Category	Value
Syntax	`PlsqlSessionCookieName` *`cookie_name`*
Default	Same as DAD name
Example	`PlsqlSessionCookieName mycookie`

- For DADs not using SingleSignOn authentication, this parameter can be omitted. In most other cases, the session cookie name should be omitted (and this parameter automatically defaults to the DAD name).
- A session cookie name must be specified only for Oracle Portal instances that need to participate in a distributed Oracle Portal environment. For those Oracle Portal nodes you want to seamlessly participate as a federated cluster, ensure that the session cookie name for all the participating nodes is the same.
- Independent Oracle Portal nodes need to use distinct session cookie names.

PlsqlSessionStateManagement

Specifies how package and session state should be cleaned up at the end of each the mod_plsql request.

- `StatelessWithResetPackageState` causes the mod_plsql module to call `dbms_session.reset_package_state` at the end of each mod_plsql request. This is the default.
- `StatelessWithPreservePackageState` causes the mod_plsql module to call `htp.init` at the end of each mod_plsql request. This cleans up the state of session variables in the PL/SQL Web ToolKit. The PL/SQL application is responsible for cleaning up its own session state. Failure to do so causes erratic behavior, in which a request starts recognizing or manipulating state modified in previous requests.
- `StatelessWithFastResetPackageState` causes the mod_plsql module to call `dbms_session.modify_package_state(dbms_session.reinitialize)` at the end of each mod_plsql request. This API is faster than the mode of `StatelessWithResetPackageState`, and avoids some latch contention issues, but exists only in Oracle database releases 8.1.7.2 and later. This mode uses slightly more memory than the default mode.

Category	Value
Syntax	`PlsqlSessionStateManagement {StatelessWithResetPackageState | StatelessWithFastResetPackageState | StatelessWithPreservePackageState}`
Default	`StatelessWithResetPackageState`
Example	`PlsqlSessionStateManagement StatelessWithPreservePackageState`

- The earlier values of stateful=no or stateful=STATELESS_RESET corresponds to `StatelessWithResetPackageState`.
- The earlier value of stateful=STATELESS_FAST_RESET corresponds to `StatelessWithFastResetPackageState`.
- The earlier value of stateful=STATELESS_PRESERVE corresponds to `StatelessWithPreservePackageState`.

The mod_plsql module does not support stateful mode of operation. To allow PL/SQL applications stateful behavior, save the state in cookies and/or in the database.

PlsqlTransferMode

Specifies the transfer mode for data from the database back to the mod_plsql module. Most applications use the default value of `CHAR`.

Category	Value
Syntax	`PlsqlTransferMode {CHAR | RAW}`
Default	`CHAR`
Example	`PlsqlTransferMode CHAR`

- This parameter only needs to be changed to enable sending back responses in different character sets from the same DAD. In such a case, the `CHAR` mode is useless, since it always converts the response data from the database character set to the mod_plsql character set.

PlsqlUploadAsLongRaw

Specifies the file extensions to be uploaded as LONGRAW data type, as opposed to using the default BLOB data type. The default can be overridden by specifying multi-line directives of file extensions for field. A value of asterisk (`*`) in this field causes all documents to be uploaded as LONGRAW.

Category	Value
Syntax	`PlsqlUploadAsLongRaw` *`string multiline`*
Default	None

Example	`PlsqlUploadAsLongRaw jpg`
	`PlsqlUploadAsLongRaw gif`

- For applications that do not upload or download documents, this parameter may be omitted.

3.9.2.3 cache.conf

The `cache.conf` file contains the configuration settings for the file system caching functionality implemented in the mod_plsql module. This configuration file is relevant only if PL/SQL applications use the `OWA_CACHE` package to cache dynamically generated content in the file system.

The following parameters are specified in the `cache.conf` file:

- PlsqlCacheCleanupTime
- PlsqlCacheDirectory
- PlsqlCacheEnable
- PlsqlCacheMaxAge
- PlsqlCacheMaxSize
- PlsqlCacheTotalSize

PlsqlCacheCleanupTime

Specifies the time to start the cleanup of the cache storage.

This setting defines the exact day and time in which cleanup should occur. The frequency can be set as daily, weekly, and monthly.

- To define daily frequency, the keyword `Everyday` is used. The cleanup starts every day at the time defined. For example, `Everyday 2:00` causes the cleanup to happen everyday at 2:00 a.m. (local time).
- To define weekly frequency, the days of the week, (`Sunday`, `Monday`, `Tuesday`, `Wednesday`, `Thursday`, `Friday`, `Saturday`) are used. For example, `Wednesday 15:30` causes the cleanup to happen every Wednesday at 3:30 p.m. (local time).
- To define monthly frequency, the keyword `Everymonth` is used. The cleanup starts on the Saturday of the month at the time defined. For example, Saturday Everymonth 23:00 causes the cleanup to happen the first Saturday of every month at 11:00 p.m. (local time).

Category	Value
Syntax	`PlsqlCacheCleanupTime {Sunday-Saturday | Everyday | Everymonth} {hh:mm}`
Default	Saturday 23:00
Example	`PlsqlCacheCleanupTime Monday 20:00`

PlsqlCacheDirectory

Specifies the directory where cache files are written out by the mod_plsql module. This directory must exist or Oracle HTTP Server will not start.

On UNIX, this directory must have write permissions by the owner of the child httpd processes.

Category	Value
Syntax	`PlsqlCacheDirectory` *`directory`*
Default	none
Example	`PlsqlCacheDirectory` *`ORACLE_INSTANCE`*`/OHS/<ohs_name>`

PlsqlCacheEnable

Enables mod_plsql caching.

Category	Value
Syntax	`PlsqlCacheEnable {On | Off}`
Default	`Off`
Example	`PlsqlCacheEnable On`

- If an application does not make use of the `OWA_CACHE` package in the PL/SQL Web Toolkit, then you can choose to disable caching. In such situations, there will be a minor performance benefit.

PlsqlCacheMaxAge

Specifies the maximum time, in days, a cache file can reside in a file system cache, after which the cached file will be removed for cache maintenance.

This setting is to ensure that the cache system does not contain old content. This setting removes old cache files and makes space for new ones.

Category	Value
Syntax	`PlsqlCacheMaxAge` *`number`*
Default	30 (days)
Example	`PlsqlCacheMaxAge 20`

PlsqlCacheMaxSize

Specifies the maximum possible size of a cache file.

This setting prevents the case in which one file can fill up the entire cache. In general, it is recommended that this be set to about 1-3 percent of the total cache size, which is specified by PlsqlCacheTotalSize.

Category	Value
Syntax	`PlsqlCacheMaxSize` *`number`*
Default	1048576
Example	`PlsqlCacheMaxSize 1048576`

PlsqlCacheTotalSize

Specifies the total size of the cache directory. The default is 20 MB.

This setting limits the amount of space the cache is allowed to use. Both PL/SQL cache and Session Cookie cache share this cache space. This setting is not a hard limit. It might exceed the limit temporarily during normal processing. This is normal behavior.

The cleanup algorithm uses this setting to determine how much to reduce the cache files. Therefore, the real space limit is the physical storage's available size.

This parameter takes bytes as values:

- 1 megabytes = 1048576 bytes
- 10 megabytes = 10485760 bytes

Category	Value
Syntax	`PlsqlCacheTotalSize` *`number`*
Default	20971520 (bytes)
Example	`PlsqlCacheTotalSize 20971520`

3.9.3 Configuration Files and Parameters

Table 3–3 lists the mod_plsql files and their corresponding configuration parameters that were described in the preceding sections.

While specifying a value for a configuration parameter, follow Oracle HTTP Server conventions for specifying values. For instance, if a value has white spaces in it, enclose the value with double quotes. For example:

```
PlsqlNLSLanguage "TRADITIONAL CHINESE_TAIWAN.UTF8"
```

Multi-line directives enable you to specify same directive multiple times in a DAD.

Table 3–3 mod_plsql Configuration Files and Parameters

Configuration File	Parameters
plsql.conf	PlsqlDMSEnable PlsqlLogEnable PlsqlLogDirectory PlsqlIdleSessionCleanupInterval

Table 3–3 (Cont.) mod_plsql Configuration Files and Parameters

Configuration File	Parameters
dads.conf	PlsqlAfterProcedure
	PlsqlAlwaysDescribeProcedure
	PlsqlAuthenticationMode
	PlsqlBeforeProcedure
	PlsqlBindBucketLengths
	PlsqlBindBucketWidths
	PlsqlCGIEnvironmentList
	PlsqlConnectionTimeout
	PlsqlConnectionValidation
	PlsqlDatabaseConnectString
	PlsqlDatabasePassword
	PlsqlDatabaseUserName
	PlsqlDefaultPage
	PlsqlDocumentPath
	PlsqlDocumentProcedure
	PlsqlDocumentTablename
	PlsqlErrorStyle
	PlsqlExclusionList
	PlsqlFetchBufferSize
	PlsqlInfoLogging
	PlsqlMaxRequestsPerSession
	PlsqlNLSLanguage
	PlsqlPathAlias
	PlsqlPathAliasProcedure
	PlsqlRequestValidationFunction
	PlsqlSessionCookieName
	PlsqlSessionStateManagement
	PlsqlTransferMode
	PlsqlUploadAsLongRaw
cache.conf	PlsqlCacheCleanupTime
	PlsqlCacheDirectory
	PlsqlCacheEnable
	PlsqlCacheMaxAge
	PlsqlCacheMaxSize
	PlsqlCacheTotalSize

3.10 mod_security

This module is an open source Web application firewall, version 2.5.9. For more information about this module, see the bundled *ModSecurity Reference Manual* (`modsecurity2_reference.pdf`) in the `ORACLE_HOME/OHS/docs` directory.

See Also: The documentation on the ModSecurity Web site.

3.11 mod_wl_ohs

This module allows requests to be proxied from Oracle HTTP Server to Oracle WebLogic Server. The mod_wl_ohs module provides the same functionality as the Oracle WebLogic Server Plug-in for Apache HTTP Server (mod_weblogic) except for some minor differences, as follows:

- Uses Oracle's security layer (NZ) to provide SSL support for the module. A new directive, WlSSlWallet, has been added to Oracle HTTP Server through the mod_wl_ohs module that allows the use of Oracle Wallets.
- Supports two-way SSL between Oracle HTTP Server and Oracle WebLogic Server.
- Supports IPv6 for communication with WebLogic Server.

For more information on the Oracle WebLogic Server Plug-in for Apache HTTP Server (mod_weblogic), see *Using Web Server Plug-Ins with Oracle WebLogic Server*.

3.11.1 Configure the mod_wl_ohs Module on Oracle HTTP Server

The mod_wl_ohs module is installed and loaded out-of-the-box with Oracle HTTP Server, but it is not configured by default. Therefore, you must configure mod_wl_ohs to specify the application requests that the module should handle.

See Section 4.4.4, "Configuring the mod_wl_ohs Module"

3.11.2 Using SSL With mod_wl_ohs

For instructions on configuring SSL for the mod_wl_ohs module, refer to "Enable SSL for Outbound Requests from Oracle HTTP Server" in the *Oracle Fusion Middleware Administrator's Guide*.

3.11.3 Configuring IPv6 For mod_wl_ohs

The mod_wl_ohs module can be configured to communicate with Oracle WebLogic Server using IPv6.

For details on configuring IPv6 for mod_wl_ohs, refer to Configuring Oracle HTTP Server for IPv6 in the *Oracle Fusion Middleware Administrator's Guide*.

Part II

Managing Oracle HTTP Server

This part presents information about management tasks for Oracle HTTP Server. It contains the following chapters:

4

Getting Started with Oracle HTTP Server

This chapter provides information on getting started with Oracle HTTP Server. It discusses the procedures needed to configure and use Oracle HTTP Server in your environment.

This chapter includes the following sections:

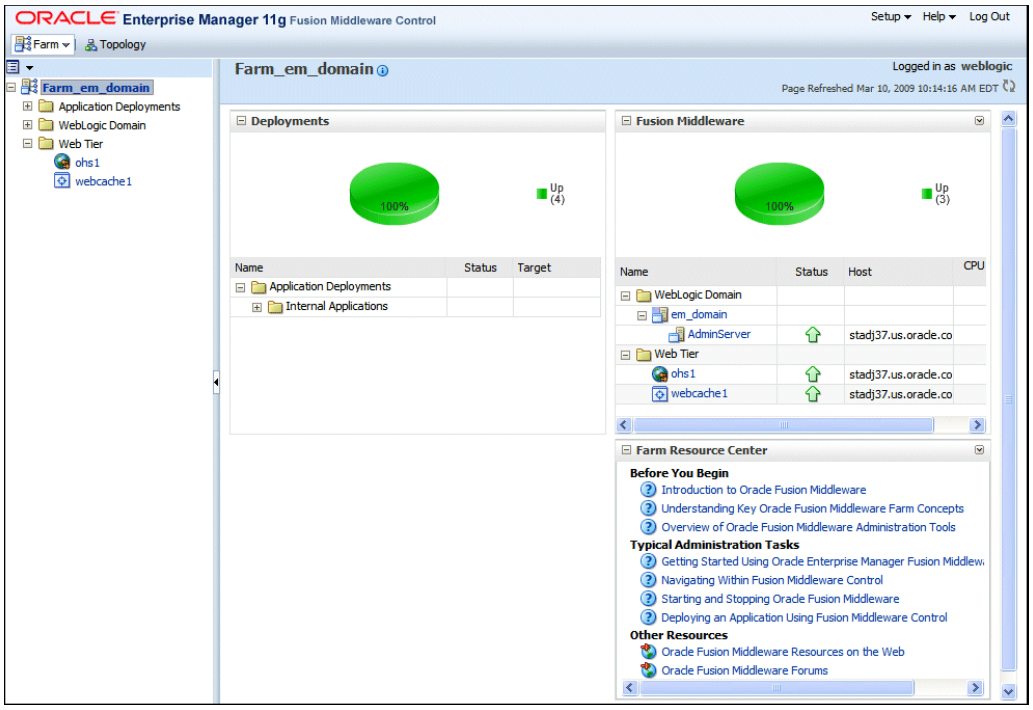
- Section 4.1, "Starting, Stopping, and Restarting Oracle HTTP Server"
- Section 4.2, "Creating a New Oracle HTTP Server Component"
- Section 4.3, "Specifying Server Properties"
- Section 4.4, "Configuring Oracle HTTP Server"
- Section 4.5, "Deleting an Oracle HTTP Server Component"

4.1 Starting, Stopping, and Restarting Oracle HTTP Server

You can use Fusion Middleware Control or the `opmnctl` command to start, stop, and restart Oracle HTTP Server.

Note: Do not use the `apachectl` utility to manage Oracle HTTP Server.

The Fusion Middleware Control home page shows the status of all installed components, including Oracle HTTP Server, as illustrated in the following figure:



Farm_em_domain

Logged in as weblogic

Page Refreshed Mar 10, 2009 10:14:16 AM EDT

- Farm_em_domain
  - Application Deployments
  - WebLogic Domain
  - Web Tier
    - ohs1
    - webcache1

Deployments

100%

Up (4)

Name	Status	Target
Application Deployments		
Internal Applications		

Fusion Middleware

100%

Up (3)

Name	Status	Host	CPU
WebLogic Domain			
em_domain			
AdminServer	⇧	stadj37.us.oracle.co	
Web Tier			
ohs1	⇧	stadj37.us.oracle.co	
webcache1	⇧	stadj37.us.oracle.co	

Farm Resource Center

Before You Begin
- Introduction to Oracle Fusion Middleware
- Understanding Key Oracle Fusion Middleware Farm Concepts
- Overview of Oracle Fusion Middleware Administration Tools

Typical Administration Tasks
- Getting Started Using Oracle Enterprise Manager Fusion Middlew
- Navigating Within Fusion Middleware Control
- Starting and Stopping Oracle Fusion Middleware
- Deploying an Application Using Fusion Middleware Control

Other Resources
- Oracle Fusion Middleware Resources on the Web
- Oracle Fusion Middleware Forums

You can determine the status of Oracle HTTP Server using `opmnctl`:

```
opmnctl status

Processes in Instance: instance1
---------------------------------+--------------------+---------+---------
ias-component                    | process-type       |     pid | status
---------------------------------+--------------------+---------+---------
webcache1                        | WebCache-admin     |   19556 | Alive
webcache1                        | WebCache           |   19555 | Alive
ohs1                             | OHS                |    7249 | Alive
```

4.1.1 Understanding the PID File

When Oracle HTTP Server starts up, it writes the process ID (PID) of the parent `httpd` process to the `httpd.pid` file located, by default, in the following directory:

```
ORACLE_INSTANCE/diagnostics/logs/OHS/<ohs_name>/
```

The process ID can be used by the administrator when restarting and terminating the daemon. If a process stops abnormally, it is necessary to stop the httpd child processes using the kill command.

The `PidFile` directive in `httpd.conf` specifies the location of the PID file.

Note: On UNIX/Linux platforms, if you edit the `PidFile` directive, you also have to edit the `ORACLE_HOME/ohs/bin/apachectl` file to specify the new location of the PID file.

See Also: PidFile directive in the Apache Server documentation.

4.1.2 Starting Oracle HTTP Server

This section describes how to start Oracle HTTP Server using Fusion Middleware Control and `opmnctl`.

4.1.2.1 Using Fusion Middleware Control to Start Oracle HTTP Server

To start Oracle HTTP Server using Fusion Middleware Control:

1. Navigate to the Oracle HTTP Server home page.
2. Select **Control** from the Oracle HTTP Server menu.
3. Select **Start Up** from the Control menu.

4.1.2.2 Using opmnctl to Start Oracle HTTP Server

To start all Oracle HTTP Server components in an Oracle instance using `opmnctl`:

```
opmnctl startproc process-type=OHS
```

To start a specific Oracle HTTP Server component, such as `<ohs_name>/`, using `opmnctl`:

```
opmnctl startproc ias-component=<ohs_name>
```

To get detailed information about the start process, include the verbose option with the command, as follows:

```
opmnctl verbose startproc ias-component=<ohs_name>
```

The following is an example of the information provided using the verbose option:

```
HTTP/1.1 200 OK
Content-Length: 656
Content-Type: text/html
Response: 1 of 1 processes started.

<?xml version='1.0' encoding='WINDOWS-1252'?>
<response>
<opmn id="stadk58:6701" http-status="200" http-response="1 of 1 processes
started.">
  <ias-instance id="inst1">
    <ias-component id="ohs1">
      <process-type id="OHS">
        <process-set id="OHS">
          <process id="699033550" pid="11366" status="Alive" index="1"
log="/scratch/oracle/product/11110/test090306/instances/inst1/diagnostics/logs/OHS
/ohs1/console~OHS~1.log" operation="request" result="success">
            <msg code="0" text="">
            </msg>
          </process>
        </process-set>
      </process-type>
    </ias-component>
  </ias-instance>
</opmn>
</response>
```

4.1.2.3 Starting Oracle HTTP Server on a Privileged Port

On a UNIX system the TCP/IP port numbers below 1024 are special in that only processes with root privileges are allowed to listen on those ports.

By default, Oracle HTTP Server runs as a non-root user (the user that installed Oracle Fusion Middleware). Therefore, on UNIX systems, if you plan on running Oracle HTTP Server on a privileged port (for example, port 80), you must enable Oracle HTTP Server to run as root, as follows:

1. Stop Oracle HTTP Server using Fusion Middleware Control, or with the following `opmnctl` command:

   ```
   opmnctl stopproc ias-component=<ohs_name>
   ```

2. Change to the root user.
3. Navigate to `ORACLE_HOME/ohs/bin` and run the following commands:

   ```
   chown root .apachectl
   chmod 6750 .apachectl
   ```

4. Exit as the root user.
5. Add or uncomment the User and Group directives in the `httpd.conf` file and set them to the user and group that were used to install and configure Oracle Fusion Middleware.
6. Start Oracle HTTP Server using Fusion Middleware Control, or with the following command:

   ```
   opmnctl startproc ias-component=<ohs_name>
   ```

4.1.3 Stopping Oracle HTTP Server

This section describes how to stop Oracle HTTP Server using Fusion Middleware Control and `opmnctl`. Other services may be impacted when Oracle HTTP Server is stopped.

4.1.3.1 Using Fusion Middleware Control to Stop Oracle HTTP Server

To stop Oracle HTTP Server using Fusion Middleware Control:

1. Navigate to the Oracle HTTP Server home page.
2. Select **Control** from the Oracle HTTP Server menu.
3. Select **Shut Down** from the Control menu.

4.1.3.2 Using opmnctl to Stop Oracle HTTP Server

To stop all Oracle HTTP Server components in an Oracle instance using `opmnctl`:

```
opmnctl stopproc process-type=OHS
```

To stop a specific Oracle HTTP Server component, such as `<ohs_name>`, using `opmnctl`:

```
opmnctl stopproc ias-component=<ohs_name>
```

To get detailed information about the stop process, include the verbose option with the command, as follows:

```
opmnctl verbose stopproc ias-component=<ohs_name>
```

4.1.4 Restarting Oracle HTTP Server

Restarting Oracle HTTP Server causes the Apache parent process to advise its child processes to exit after their current request (or to exit immediately if they are not serving any requests). Upon restarting, the parent process re-reads its configuration files and reopens its log files. As each child process exits, the parent replaces it with a child process from the new generation of the configuration file, which begins serving new requests immediately.

Note: If your configuration file contains errors, when you issue a restart command, the Apache parent process will not restart; it will exit with an error but it will also leave child processes running when it exits. (These are the children that are gracefully exiting by handling their last request.) This will cause problems if you attempt to restart Oracle HTTP Server because it will not be able to bind to its listening ports. Therefore, before restarting a server, make sure there are no syntax or semantics errors in the configuration file(s).

The following sections describe how to restart Oracle HTTP Server using Fusion Middleware Control and `opmnctl`.

4.1.4.1 Using Fusion Middleware Control to Restart Oracle HTTP Server

To restart Oracle HTTP Server using Fusion Middleware Control:

1. Navigate to the Oracle HTTP Server home page.
2. Select the **Control** from the Oracle HTTP Server menu.
3. Select **Restart** from the Control menu.

4.1.4.2 Using opmnctl to Restart Oracle HTTP Server

To restart all Oracle HTTP Server components in an Oracle instance using `opmnctl`:

```
opmnctl restartproc process-type=OHS
```

To restart a specific Oracle HTTP Server component, such as `<ohs_name>`, using `opmnctl`:

```
opmnctl restartproc ias-component=<ohs_name>
```

To get detailed information about the start process, include the verbose option with the command, as follows:

```
opmnctl verbose restartproc ias-component=<ohs_name>
```

4.2 Creating a New Oracle HTTP Server Component

Oracle HTTP Server is not automatically created during installation unless it was selected as a component to be installed. This section describes how to create an Oracle HTTP Server component using `opmnctl`. You cannot create an Oracle HTTP Server component using Fusion Middleware Control.

To create a new Oracle HTTP Server component using `opmnctl`, use the following command:

```
opmnctl createcomponent -componentType OHS -componentName name
```

For example, to create an Oracle HTTP Server instance named `ohs2`, use the following command:

```
opmnctl createcomponent -componentType OHS -componentName ohs2
```

When you create the Oracle HTTP Server component ports are automatically assigned. However, you can use the following parameters to specify the ports of your choice:

- `-listenPort` – HTTP listening port
- `-sslPort` – HTTPS (SSL) listening port
- `-proxyPort` – Proxy MBean port internally used by Oracle HTTP Server to communicate with Fusion Middleware Control

4.3 Specifying Server Properties

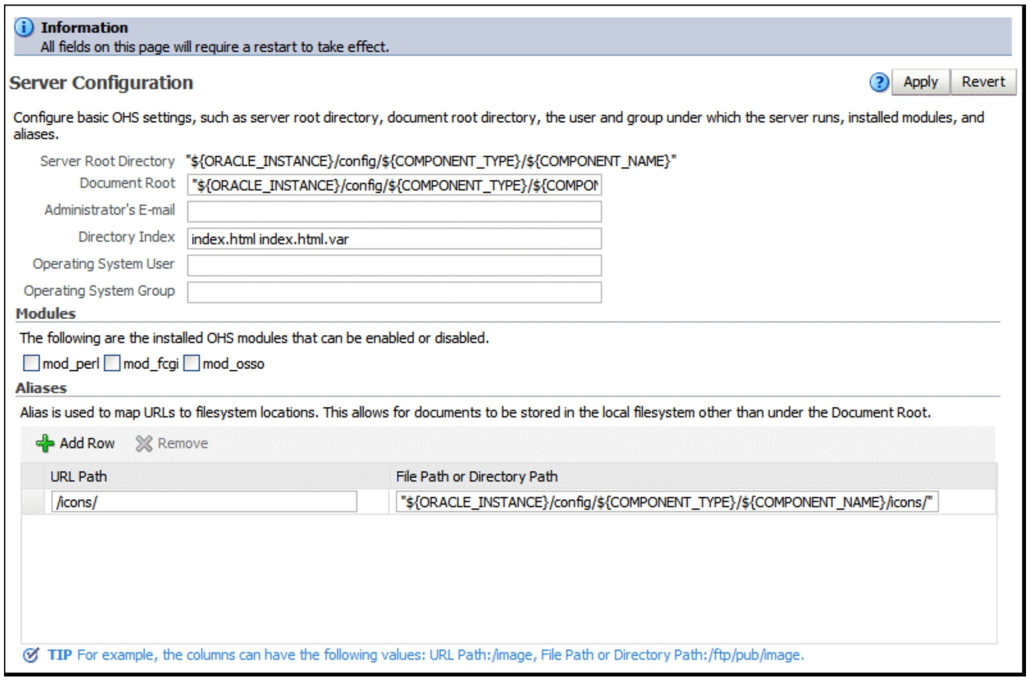
Server properties for Oracle HTTP Server can be set using Fusion Middleware Control or direct editing of the Oracle HTTP Server configuration files. You cannot specify the server properties using `opmnctl` commands.

- Using Fusion Middleware Control to Specify Server Properties
- Editing the httpd.conf File to Specify Server Properties

4.3.1 Using Fusion Middleware Control to Specify Server Properties

To specify the server properties using the Fusion Middleware Control:

1. Select **Administration** from the Oracle HTTP Server menu.
2. Select **Server Configuration** from the Administration menu. The Server Configuration page appears.



Information
All fields on this page will require a restart to take effect.

Server Configuration

Configure basic OHS settings, such as server root directory, document root directory, the user and group under which the server runs, installed modules, and aliases.

Server Root Directory	"${ORACLE_INSTANCE}/config/${COMPONENT_TYPE}/${COMPONENT_NAME}"
Document Root	"${ORACLE_INSTANCE}/config/${COMPONENT_TYPE}/${COMPOI
Administrator's E-mail	
Directory Index	index.html index.html.var
Operating System User	
Operating System Group	

Modules

The following are the installed OHS modules that can be enabled or disabled.

☐ mod_perl ☐ mod_fcgi ☐ mod_osso

Aliases

Alias is used to map URLs to filesystem locations. This allows for documents to be stored in the local filesystem other than under the Document Root.

URL Path	File Path or Directory Path
/icons/	"${ORACLE_INSTANCE}/config/${COMPONENT_TYPE}/${COMPONENT_NAME}/icons/"

TIP For example, the columns can have the following values: URL Path:/image, File Path or Directory Path:/ftp/pub/image.

3. Enter the documentation root directory in the Document Root field that forms the main document tree visible from the Web site.
4. Enter the e-mail address in the Administrator's E-mail Address field that the server will includes in error messages sent to the client.
5. Enter the directory index in the Directory Index field. The is the main (index) page that will be displayed when a client first accesses the Web site.
6. Enter the user name in the Operating System User field.

   This is the user name for the server, when sending and responding to requests. The user should not have privileges that allow it to access files or run programs that are for internal-use only. For example, when a request comes from Oracle Portal, Oracle HTTP Server will respond as the user defined in this field, and should have privileges to access the content in Oracle Portal. However, the user should not have privileges to access company-confidential content.

   Oracle recommends that you set up a user specifically for running the server. Oracle also recommends that you do **not** set the user to root.
7. Enter the group name in the Operating System Group field. This is the group for the server, when sending and responding to requests. The user defined for Oracle HTTP Server must be a member of this group.

   Oracle recommends that you set up a group specifically for running the server. Oracle also recommends that you do **not** set the group as root.
8. The Modules region is used to enable or disable modules. There are three modules that you can enable or disable: mod_perl, mod_fcgi, and mod_osso.

   For instructions on configuring the mod_perl module, see "Configuring the mod_perl Module" on page 4-11.
9. Create an alias, if necessary in the Aliases table. An alias maps to a specified directory. For example, to use a specific set of content pages for a group you can create an alias to the directory that has the content pages.
10. Review the settings. If the settings are correct, then click **Apply** to apply the changes. If the settings are incorrect, or you decide to not apply the changes, then click **Revert** to return to the original settings.
11. Restart Oracle HTTP Server. See Section 4.1.4.

The server properties are saved, and shown on the Server Configuration page.

4.3.2 Editing the httpd.conf File to Specify Server Properties

To specify the server properties using the `httpd.conf` file:

1. Open the `httpd.conf` file using either a text editor or the Advanced Server Configuration page in Fusion Middleware Control. (See Section 4.4.6, "Modifying an Oracle HTTP Server Configuration File.")
2. In the `DocumentRoot` section of the file, enter the directory that stores the main content for the Web site. The following is an example of the syntax:

   ```
   DocumentRoot "${ORACLE_INSTANCE}/config/${COMPONENT_TYPE}/${COMPOENT_
   NAME}/htdocs"
   ```
3. In the `ServerAdmin` section of the file, enter the administrator's e-mail address. This is the e-mail address that will appear on client pages. The following is an example of the syntax:

```
ServerAdmin WebMaster@example.com
```

4. In the `DirectoryIndex` section of the file, enter the directory index. This is the main (index) page that will be displayed when a client first accesses the Web site. The following is an example of the syntax:

   ```
   DirectoryIndex index.html index.html.var
   ```

5. In the `User` and `Group` section of the file, enter the user name and group. The following is an example of the syntax:

   ```
   User nobody
   Group nobody
   ```

   The user name is for the server, when sending and responding to requests. The user should not have privileges that allow it to access files or run programs that are for internal-use only. For example, when a request comes from Oracle Portal, Oracle HTTP Server will respond as the user defined in this field, and should have privileges to access the content in Oracle Portal. However, the user should not have privileges to access company-confidential content.

   Oracle recommends that you set up a group specifically for running the server. Oracle also recommends that you do **not** set the group as root. The user defined for Oracle HTTP Server must be a member of this group.

6. Create aliases, if needed. An alias maps to a specified directory. For example, to use a specific set of icons, you can create an alias to the directory that has the icons for the Web pages. The following is an example of the syntax:

   ```
   Alias /icons/ "${ORACLE_HOME}/config/${COMPONENT_TYPE}/${COMPONENT_
   NAME}/icons/"

   <Directory "${ORACLE_HOME}/content/${COMPONENT_TYPE}/${COMPONENT_NAME}/icons/">
       Options MultiViews
       AllowOverride None
       Order allow, deny
       Allow from all
   </Directory>
   ```

7. Save the file.

8. Restart Oracle HTTP Server. See Section 4.1.4.

4.4 Configuring Oracle HTTP Server

This section includes the following sections:

- Section 4.4.1, "Configuring Secure Sockets Layer"
- Section 4.4.2, "Configuring MIME Settings"
- Section 4.4.3, "Configuring the mod_perl Module"
- Section 4.4.4, "Configuring the mod_wl_ohs Module"
- Section 4.4.5, "Enabling the mod_osso Module"
- Section 4.4.6, "Modifying an Oracle HTTP Server Configuration File"
- Section 4.4.7, "Disabling the Options Method"

4.4.1 Configuring Secure Sockets Layer

Secure Sockets Layer (SSL) is an encrypted communication protocol that is designed to securely send messages across the Internet. It resides between Oracle HTTP Server on the application layer and the TCP/IP layer, transparently handling encryption and decryption when a secure connection is made by a client.

One common use of SSL is to secure Web HTTP communication between a browser and a Web server. This case does not preclude the use of non-secured HTTP. The secure version is simply HTTP over SSL (HTTPS). The differences are that HTTPS uses the URL scheme `https://` rather than `http://`.

By default, an SSL listen port is configured and enabled using a default **wallet** during Oracle HTTP Server installation. Wallets store your credentials, such as certificate requests, certificates, and private keys.

The default wallet that is automatically installed with Oracle HTTP Server is for testing purposes only. A real wallet must be created for your production server. The default wallet is located in the *`ORACLE_INSTANCE`*`/config/OHS/<ohs_name>/keystores/default` directory. You can either place the new wallet in this location, or change the SSLWallet directive in *`ORACLE_INSTANCE`*`/config/OHS/<ohs_name>/ssl.conf` to point to the location of your real wallet.

4.4.1.1 Using Fusion Middleware Control to Configure a Wallet and SSL

For instructions on configuring wallets and SSL using Fusion Middleware Control, see "Enabling SSL for Oracle HTTP Server Virtual Hosts" in the Oracle Fusion Middleware Administrator's Guide.

4.4.2 Configuring MIME Settings

Multipurpose Internet Mail Extension (MIME) settings are used by Oracle HTTP Server to interpret file types, encodings, and languages. MIME settings for Oracle HTTP Server can only be set using Fusion Middleware Control. You cannot specify the MIME settings using `opmnctl` commands.

The following tasks can be completed on the MIME Configuration page:

- Configuring MIME Types
- Configuring MIME Encoding
- Configuring MIME Languages

4.4.2.1 Configuring MIME Types

MIME type maps a given file extension to a specified content type. The MIME type is used for filenames containing an extension.

4.4.2.1.1 Using Fusion Middleware Control to Configure MIME Types

To configure a MIME type using Fusion Middleware Control, do the following:

1. Select **Administration** from the Oracle HTTP Server menu.
2. Select **MIME Configuration** from the Administration menu. The MIME configuration page appears.
3. Click **Add Row** in MIME Configuration region. A new, blank row is added to the list.
4. Enter the MIME type.

5. Enter the file extension.
6. Review the settings. If the settings are correct, click **Apply** to apply the changes. If the settings are incorrect, or you decide to not apply the changes, click **Revert** to return to the original settings.
7. Restart Oracle HTTP Server. See Section 4.1.4.

The MIME configuration is saved, and shown on the MIME Configuration page.

4.4.2.2 Configuring MIME Encoding

MIME encoding enables Oracle HTTP Server to determine the file type based on the file extension. You can add and remove MIME encodings. The encoding directive maps the file extension to a specified encoding type.

1. Select **Administration** from the Oracle HTTP Server menu.
2. Select **MIME Configuration** from the Administration menu. The MIME configuration page appears.
3. Expand the MIME Encoding region by clicking the plus sign (+) next to MIME Encoding.
4. Click **Add Row** in MIME Encoding region. A new, blank row is added to the list.
5. Enter the MIME encoding, such as x-gzip.
6. Enter the file extension, such as .gx.
7. Review the settings. If the settings are correct, click **Apply** to apply the changes. If the settings are incorrect, or you decide to not apply the changes, click **Revert** to return to the original settings.
8. Restart Oracle HTTP Server. See Section 4.1.4.

4.4.2.3 Configuring MIME Languages

The MIME language setting maps file extensions to a particular language. This directive is commonly used for content negotiation, in which Oracle HTTP Server returns the document that most closely matched the preferences set by the client.

1. Select **Administration** from the Oracle HTTP Server menu.
2. Select **MIME Configuration** from the Administration menu. The MIME configuration page appears.
3. Expand the MIME Languages region by clicking the plus sign (+) next to MIME Languages.
4. Click **Add Row** in MIME Languages region. A new, blank row is added to the list.
5. Enter the MIME language, such as en-US.
6. Enter the file extension, such as en-us.
7. Review the settings. If the settings are correct, click **Apply** to apply the changes. If the settings are incorrect, or you decide to not apply the changes, click **Revert** to return to the original settings.
8. Restart Oracle HTTP Server. See Section 4.1.4.

4.4.3 Configuring the mod_perl Module

The mod_perl module embeds the Perl interpreter into Oracle HTTP Server. This eliminates start-up overhead and enables you to write modules in Perl. The module is disabled, by default.

To enable the mod_perl module using Fusion Middleware Control, do the following:

1. Select **Administration** from the Oracle HTTP Server menu.
2. Select **mod_perl Configuration** from the Administration menu. The mod_perl configuration page appears.

   Note: If mod_perl has not been enabled, then you will be redirected to the Server Configuration page. Select mod_perl and click **Apply** to enable mod_perl. After the confirmation page has been displayed, restart Oracle HTTP Server, and then return to the mod_perl Configuration page.

3. Enter the switch information in the Switches field.
4. Enter the environment variables to be passed to the scripts in the Environment field.
5. Enter the required script names in the Require field.
6. Click **Add Row** to create a new row.
7. Configure mod_perl directives for a Location in the Perl Locations table. The Location assigns a number of rules that the server should follow when the request's URI matches the Location.

   a. Enter the base URI for the Perl scripts in the Locations field. Just as it is the widely accepted convention to use `/cgi-bin` for your mod_cgi scripts, it is also conventional to use `/perl` as the base URI of the Perl scripts that are running under mod_perl.

   b. Enter options in the Options field. The `PerlOptions` directive provides fine-grained configuration by providing control over which class of Perl interpreter pool to be used. Options are enabled by prepending them with a plus sign (+) and are disabled by prepending them with a minus sign (-).

   c. If you want to send headers, then click the **Send Header** check box. The `PerlSendHeader` directive is for mod_perl 1.0 backwards-compatibility. When enabled, the server sends an HTTP header to the browser on every script invocation. You should disable this option for NPH (non-parsed-headers) scripts.

   d. Enter the environment in the Environment field. The `PerlSetEnv` directive allows you to specify system environment variables and pass them into your mod_perl handlers.

   e. Enter the response handler in the Response Handler field. The `PerlResponseHandler` directive tells mod_perl which callback is going to do the job.

   f. Enter the authentication handler in the Authentication Handler field. The `PerlAuthenHandler` directive is used to set the handler to verify a user's identification credentials.

8. Review the settings. If the settings are correct, click **Apply** to apply the changes. If the settings are incorrect, or you decide to not apply the changes, click **Revert** to return to the original settings.

9. Restart Oracle HTTP Server. See Section 4.1.4.

The mod_perl module configuration is saved and shown on the mod_perl Configuration page.

Note: If you are manually editing the mod_perl configuration instead of using Fusion Middleware Control, then all directives must be defined within the `<IfModule mod_perl.c>` block of the `mod_perl.conf` file. Any mod_perl related directive defined outside of this block might be ignored.

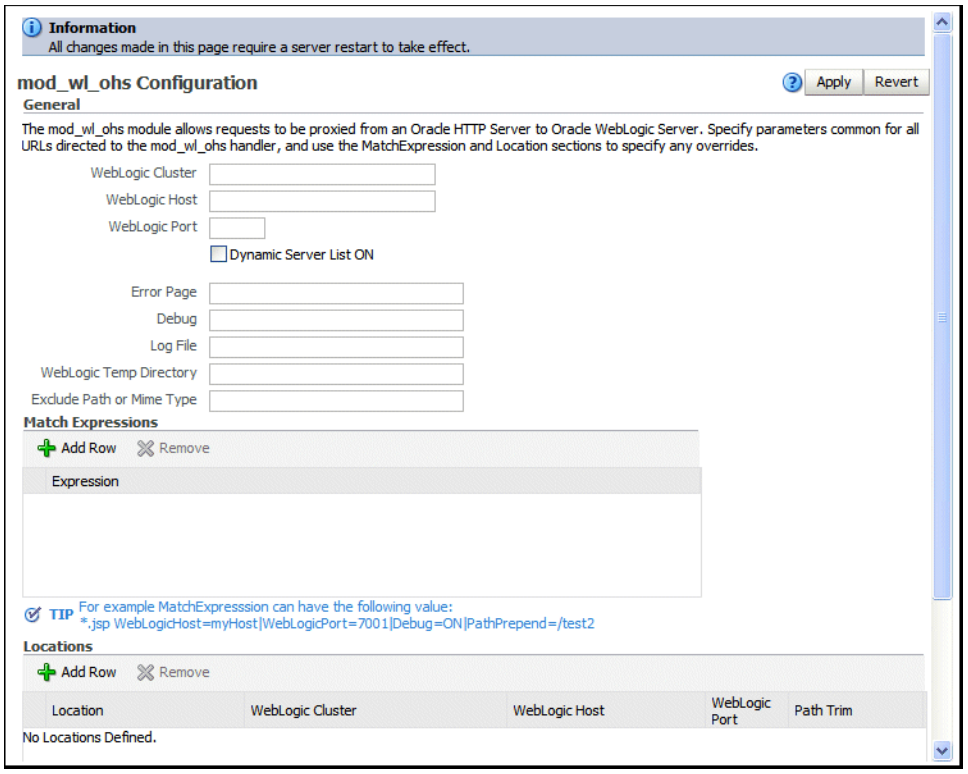
4.4.4 Configuring the mod_wl_ohs Module

The mod_wl_ohs module allows requests to be proxied from an Oracle HTTP Server to Oracle WebLogic Server.

To configure the mod_wl_ohs module using Fusion Middleware Control, do the following:

1. Select **Administration** from the Oracle HTTP Server menu.

2. Select **mod_wl_ohs Configuration** from the Administration menu. The mod_wl_ohs configuration page appears.


Information
All changes made in this page require a server restart to take effect.
mod_wl_ohs Configuration
Apply
Revert
General
The mod_wl_ohs module allows requests to be proxied from an Oracle HTTP Server to Oracle WebLogic Server. Specify parameters common for all URLs directed to the mod_wl_ohs handler, and use the MatchExpression and Location sections to specify any overrides.

WebLogic Cluster	
WebLogic Host	
WebLogic Port	
	☐ Dynamic Server List ON
Error Page	
Debug	
Log File	
WebLogic Temp Directory	
Exclude Path or Mime Type	

TIP For example MatchExpresssion can have the following value:
*.jsp WebLogicHost=myHost|WebLogicPort=7001|Debug=ON|PathPrepend=/test2

Location	WebLogic Cluster	WebLogic Host	WebLogic Port	Path Trim
No Locations Defined.				

3. If you are using a WebLogic cluster, enter the WebLogic Servers that can be used for load balancing in the WebLogic Cluster field. The server or cluster list is a list of `host:port` entries. If a mixed set of clusters and single servers is specified, the dynamic list returned for this parameter will return only the clustered servers.

The module does a simple round-robin between all available servers. The server list specified in this property is a starting point for the dynamic server list that the server and module maintain. WebLogic Server and the module work together to update the server list automatically with new, failed, and recovered cluster members.

You can disable the use of the dynamic cluster list by disabling the Dynamic Server List ON field. The module directs HTTP requests containing a cookie, URL-encoded session, or a session stored in the POST data to the server in the cluster that originally created the cookie.

4. Use the WebLogic Host field to enter the WebLogic Server host (or virtual host name as defined in WebLogic Server) to which HTTP requests should be forwarded. If you are using a WebLogic cluster, use the WebLogic Cluster field instead of WebLogic Host.
5. Use the WebLogic Port field to enter the port on which the WebLogic Server host is listening for connection requests from the module (or from other servers). (If you are using SSL between the module and WebLogic Server, set this parameter to the SSL listen port.
6. If you want to use the dynamic cluster list for load balancing requests proxied from the module, then select the **Dynamic Server List ON** check box. When set to OFF, the module ignores the dynamic cluster list and only uses the static list specified with the WebLogic Cluster parameter. Normally this parameter should be set to ON.
7. You can use the Error Page field to create your own error page that is displayed when your Web server is unable to forward requests to WebLogic Server.
8. Use the Debug field to specify the type of logging performed for debugging operations. The debugging information is written to the `/tmp/wlproxy.log` file on UNIX systems and `c:\TEMP\wlproxy.log` on Windows systems. Override this location and filename by setting the Log File parameter to a different directory and file. Ensure that the `tmp` or `TEMP` directory has write permission assigned to the user who is logged in to the server.

   The Debug parameter can be set any of the following logging options. Additionally, the HFC, HTW, HFW, and HTC options can be set in combination by entering them separated by commas; for example: `HFC,HTW`.

   - **ON** – The module logs informational and error messages.
   - **OFF** – No debugging information is logged.
   - **HFC** – The module logs headers from the client, informational, and error messages.
   - **HTW** – The module logs headers sent to WebLogic Server, and informational and error messages.
   - **HFW** – The module logs headers sent from WebLogic Server, and informational and error messages.
   - **HTC** – The module logs headers sent to the client, informational messages, and error messages.
   - **ERR** – Prints only the Error messages in the module.
   - **ALL** – The module logs headers sent to and from the client, headers sent to and from WebLogic Server, information messages, and error messages.

9. Use the Log File field to specify the path and file name for the log file that is generated when the Debug parameter is set to ON. You must create this directory before setting this parameter.

10. Use the WebLogic Temp Directory field to specify the directory where a `wlproxy.log` will be created. If the location fails, the module resorts to creating the log file under `c:/temp` in Windows and `/tmp` in all UNIX platforms

    This also specifies the location of the `_wl_proxy` directory for post data files. When both WebLogic Temp Directory and Log File are set, Log File will override as to the location of `wlproxy.log`. WebLogic Temp Directory will still determine the location of the `_wl_proxy` directory.

11. Use the Exclude Path or Mime Type field to exclude certain requests from proxying. This parameter can be defined locally at the Location tag level as well as globally. When the property is defined locally, it does not override the global property but defines a union of the two parameters.

12. The Match Expression region is used to specify any Expression overrides.

    Example when proxying by MIME type:

    ```
    *.jsp WebLogicHost=myHost|paramName=value
    ```

    It is possible to define a new parameter for Match Expression using the following syntax:

    ```
    *.jsp PathPrepend=/test PathTrim=/foo
    ```

13. The Location region is used to specify any Location overrides.

    a. Click **Add Row** to create a new row.

    b. Enter the base URI for which following directives become effective.

    c. Complete the WebLogic Cluster, WebLogic Host, and WebLogic Port fields using the definitions supplied earlier in this section.

    d. For the Path Trim field, as per the RFC specification, generic syntax for URL is:

       ```
       [PROTOCOL]://[HOSTNAME]:{PORT}/{PATH}/{FILENAME};{PATH_PARAMS}/{QUERY_
       STRING}...
       ```

       Path Trim specifies the string trimmed by the module from the `{PATH}/{FILENAME}` portion of the original URL, before the request is forwarded to WebLogic Server. For example, if the URL:

       ```
       http://myWeb.server.com/weblogic/foo
       ```

       is passed to the module for parsing and if Path Trim has been set to strip off `/weblogic` before handing the URL to WebLogic Server, the URL forwarded to WebLogic Server is:

       ```
       http://myWeb.server.com:7002/foo
       ```

       Note that if you are newly converting an existing third-party server to proxy requests to WebLogic Server using the module, you will need to change application paths to `/foo` to include `weblogic/foo`. You can use Path Trim and Path Prepend in combination to change this path.

    e. For the Path Prepend field, as per the RFC specification, generic syntax for URL is:

       ```
       [PROTOCOL]://[HOSTNAME]:{PORT}/{PATH}/{FILENAME};{PATH_PARAMS}/{QUERY_
       ```

```
STRING}...
```

Path Prepend specifies the path that the module prepends to the `{PATH}` portion of the original URL, after Path Trim is trimmed and before the request is forwarded to WebLogic Server.

Note that if you need to append File Name, use the DefaultFileName module parameter instead of Path Prepend.

f. Complete the Log File and Debug fields using the definitions supplied earlier in this section.

g. Click **Add Row** again to save the new row.

14. Review the settings. If the settings are correct, click **Apply** to apply the changes. If the settings are incorrect, or you decide to not apply the changes, click **Revert** to return to the original settings.

15. Restart Oracle HTTP Server. See Section 4.1.4.

The mod_wl_ohs module configuration is saved and shown on the mod_wl_ohs Configuration page.

Note: If you are manually editing the mod_wl_ohs configuration settings instead of using Fusion Middleware Control, then all directives should be defined within the defined within the `<IfModule weblogic_module>` block of the `mod_wl_ohs.conf` file. Mod_wl_ohs will continue to work if directives are defined outside of this block, but this could put the mod_wl_ohs Configuration page in Fusion Middleware Controlin an inconsistent state.

4.4.5 Enabling the mod_osso Module

The mod_osso module is enabled on the Server Configuration Properties page. For more information see "Using Fusion Middleware Control to Specify Server Properties" on page 4-6.

4.4.6 Modifying an Oracle HTTP Server Configuration File

To modify an Oracle HTTP Server configuration file using Fusion Middleware Control, do the following:

1. Select **Administration** from the HTTP Server menu.
2. Select **Advanced Configuration** from the Administration menu. The Advanced Configuration page appears.
3. Select the configuration file from the list, such as the `httpd.conf` file.
4. Edit the file, as needed.
5. Review the settings. If the settings are correct, click **Apply** to apply the changes. If the settings are incorrect, or you decide to not apply the changes, click **Revert** to return to the original settings.
6. Restart Oracle HTTP Server. See Section 4.1.4.

The file is saved and shown on the Advanced Configuration page.

4.4.7 Disabling the Options Method

The Options method enables clients to determine which methods are supported by a web server. If enabled, it appears in the `Allow` line of HTTP response headers.

For example, if you send a request such as:

```
---- Request -------
OPTIONS / HTTP/1.0
Content-Length: 0
Accept: */*
Accept-Language: en-US
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)
Host: host123:80
```

you might get the following response from the web server:

```
---- Response --------
HTTP/1.1 200 OK
Date: Wed, 23 Apr 2008 20:20:49 GMT
Server: Oracle-Application-Server-11g/11.1.1.0.0 Oracle-HTTP-Server
Allow: GET,HEAD,POST,OPTIONS
Content-Length: 0
Connection: close
Content-Type: text/html
```

Some sources consider exposing the Options method a low security risk because malicious clients could use it to determine the methods supported by a web server. However, because web servers support only a limited number of methods, disabling this method will just slow down malicious clients, not stop them. In addition, the Options method may be used by legitimate clients.

If your Oracle Fusion Middleware environment does not have clients that require the Options method, you can disable it by including the following lines in the `httpd.conf` file:

```
<IfModule mod_rewrite.c>
RewriteEngine on
RewriteCond %{REQUEST_METHOD} ^OPTIONS
RewriteRule .* - [F]
</IfModule>
```

4.5 Deleting an Oracle HTTP Server Component

This section describes how to delete an Oracle HTTP Server component using `opmnctl`. You cannot delete an Oracle HTTP Server component using Fusion Middleware Control.

To delete an Oracle HTTP Server component using `opmnctl`:

```
opmnctl deletecomponent -componentName name
```

For example, to delete an Oracle HTTP Server component named `ohs2` use the following command:

```
opmnctl deletecomponent -componentName ohs2
```

5 Managing and Monitoring Server Processes

This chapter describes how to manage and monitor Oracle HTTP Server. It discusses the procedures and tools to manage Oracle HTTP Server in your environment.

This chapter includes the following sections:

5.1 Oracle HTTP Server Processing Model

The following sections explain the processing model for Oracle HTTP Server.

5.1.1 Request Process Model

After Oracle HTTP Server is started, it is ready to listen for and respond to HTTP(S) requests. The request processing model on Microsoft Windows systems differs from that on UNIX systems.

- On Microsoft Windows, there is a single parent process and a single child process. The child process creates threads that are responsible for handling client requests. The number of created threads is static and can be configured for performance.
- On UNIX, there is a single parent process that manages multiple child processes. The child processes are responsible for handling requests. The parent process brings up additional child processes as necessary, based on configuration. Although the server has the ability to dynamically bring up additional child processes, it is best to configure the server to start enough child processes initially so that requests can be handled without having to spawn more child processes.

5.1.2 Single Unit Process Model

Oracle HTTP Server provides functionality that allows it to terminate as a single unit if the parent process fails. The parent process is responsible for starting and stopping all the child processes for an Oracle HTTP Server instance. The failure of the parent process without first shutting down the child processes leaves Oracle HTTP Server in an inconsistent state that can only be fixed by manually shutting down all the orphaned child processes. Until all the child processes are closed, a new Oracle HTTP Server instance cannot be started because the orphaned child processes still occupy the ports the new Oracle HTTP Server instance needs to access.

To prevent this from occurring, the DMS instrumentation layer in child processes on UNIX and monitor functionality within WinNT MPM on Windows monitor the parent process. If they detect that the parent process has failed, then all of the remaining child processes are shut down.

When this functionality is combined with OPMN, it means that Oracle HTTP Server is easily restarted in case of a parent process failure. The DMS instrumentation layer on UNIX and a monitor within WinNT MPM on Windows ensures that all of the Oracle HTTP Server child processes are shut down, leaving the ports open for a new Oracle HTTP Server instance. OPMN ensures that a new instance is started once the failure of the original instance is detected.

5.2 Monitoring Oracle HTTP Server Performance

Oracle Fusion Middleware automatically and continuously measures run-time performance for Oracle HTTP Server. The performance metrics are automatically enabled; you do not need to set options or perform any extra configuration to collect them. If you encounter a problem, such as an application that is running slowly or is hanging, you can view particular metrics to find out more information about the problem.

Note that Fusion Middleware Control provides real-time data. If you are interested in viewing historical data, consider using Grid Control.

5.2.1 Viewing Oracle HTTP Server Performance Metrics

You can view metrics from the Oracle HTTP Server home menu of Fusion Middleware Control:

1. Select the Oracle HTTP Server that you want to monitor.

   The Oracle HTTP Server home page is displayed.

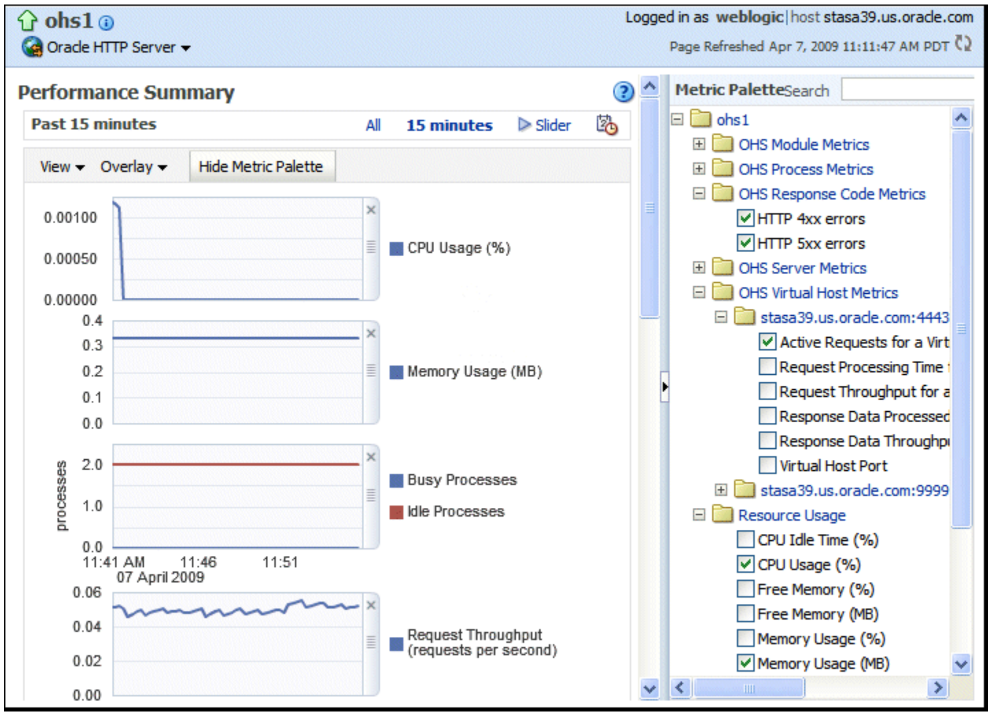
2. From the Oracle HTTP Server menu, choose **Monitoring**, and then select **Performance Summary**.

   The Performance Summary page is displayed. It shows performance metrics, as well as information about response time and request processing time for the Oracle HTTP Server instance.

3. To see additional metrics, click **Show Metric Palette** and expand the metric categories.

   Tip: Oracle HTTP Server port usage information is also available from the Oracle HTTP Server home menu.

   The following figure shows the Oracle HTTP Server Performance Summary page with the Metric Palette displayed:



4. Select additional metrics to add them to the Performance Summary.

5.2.2 Understanding Oracle HTTP Server Performance Metrics

This section lists some of the most commonly-used metrics that can help you analyze Oracle HTTP Server performance.

OHS Server Metrics

The OHS Server Metrics folder contains performance metric options for Oracle HTTP Server. The following table describes the metrics in the OHS Server Metrics folder:

Element	Description
CPU Usage	CPU usage and idle times
Memory Usage	Memory usage and free memory, in MB
Processes	Busy and idle process metrics
Request Throughput	Request throughput, as measured by requests per second
Request Processing Time	Request processing time, in seconds
Response Data Throughput	Response data throughput, in KB per second
Response Data Processed	Response data processed, in KB per second
Active HTTP Connections	Number of active HTTP connections
Connection Duration	Length of time for connections
HTTP Errors	Number of HTTP 4xx and 5xx errors

OHS Virtual Host Metrics

The OHS Virtual Host Metrics folder contains performance metric options for virtual hosts, also known as access points. The following table describes the metrics in the OHS Virtual Host Metrics folder:

Element	Description
Request Throughput for a Virtual Host	Number of requests per second for each virtual host
Request Processing Time for a Virtual Host	Time to process each request for each virtual host
Response Data Throughput for a Virtual Host	Amount of data being sent for each virtual host
Response Data Processed for a Virtual Host	Amount of data being processed for each virtual host

OHS Module Metrics

The OHS Module Metrics folder contains performance metric option for modules. The following table describes the metrics in the OHS Module Metrics folder.

Element	Description
Request Handling Throughput	Request handling throughput for a module, in requests per second
Request Handling Time	Request handling time for a module, in seconds
Module Metrics	Modules including active requests, throughput, and time for each module

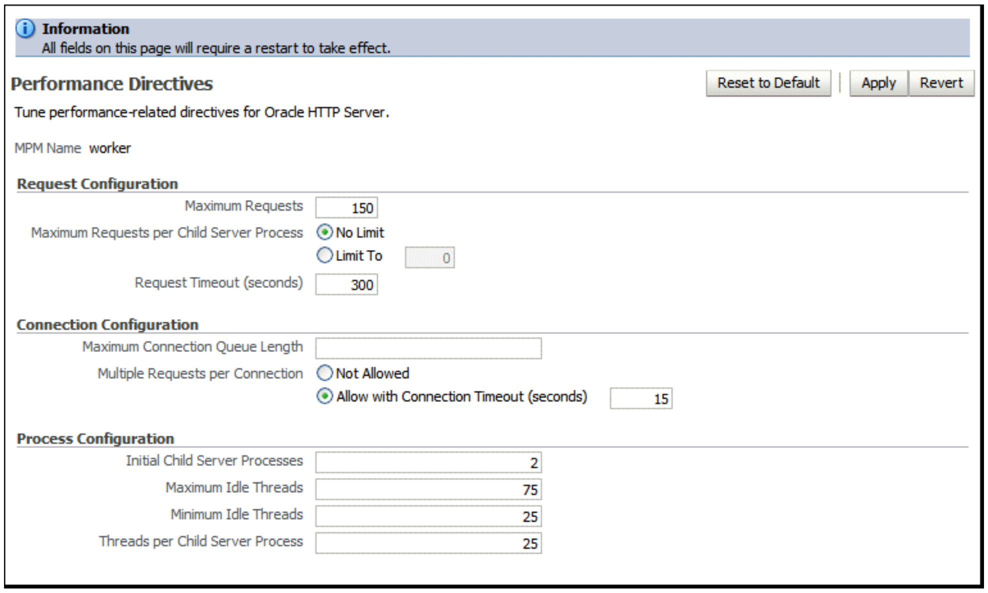
5.3 Configuring Oracle HTTP Server Performance Directives

Oracle HTTP Server uses directives in `httpd.conf`. This configuration file specifies the maximum number of HTTP requests that can be processed simultaneously, logging details, and certain limits and timeouts. Oracle HTTP Server supports and ships with the following three Multi-Processing Modules (MPMs) which are responsible for binding to network ports on the machine, accepting requests, and dispatching children to handle the requests:

- Worker - This is the default MPM for Oracle HTTP Server on UNIX/Linux platforms. This MPM implements a hybrid multi-process multi-threaded server. By using threads to serve requests, it is able to serve a large number of requests with fewer system resources than a process-based server. However, it retains much of the stability of a process-based server by keeping multiple processes available, each with many threads.
- WinNT - This is the default MPM for Oracle HTTP Server on Windows platforms. It uses a single control process which launches a single child process which in turn creates threads to handle requests.
- Prefork - This MPM implements a non-threaded, pre-forking server that handles requests in a manner similar to Apache 1.3. It is appropriate for sites that need to avoid threading for compatibility with non-thread-safe libraries. It is also the best MPM for isolating each request, so that a problem with a single request will not affect any other.

The discussion and recommendations in this section are based on the use of Worker or WinNT MPM, which uses threads. The directives listed below may not be applicable if you are using the Prefork MPM. Refer to the Oracle Application Server 10g Release 3 documentation if you are using Oracle HTTP Server based on Apache 1.3 or Apache 2.2 with Prefork MPM.

The Performance Directives page allows you to tune performance-related directives for Oracle HTTP Server, as illustrated in the following figure:



Information
All fields on this page will require a restart to take effect.

Performance Directives

Tune performance-related directives for Oracle HTTP Server.

MPM Name worker

Request Configuration

Maximum Requests	150
Maximum Requests per Child Server Process	[✓ No Limit] Limit To 0
Request Timeout (seconds)	300

Connection Configuration

Maximum Connection Queue Length	
Multiple Requests per Connection	Not Allowed [✓ Allow with Connection Timeout (seconds)] 15

Process Configuration

Initial Child Server Processes	2
Maximum Idle Threads	75
Minimum Idle Threads	25
Threads per Child Server Process	25

Performance directives management consists of three areas: request configuration, connection configuration, and process configuration. You can set these configurations using the Performance Directive page of Fusion Middleware Control and by following the instructions in the following sections:

- Using Fusion Middleware Control to Set the Request Configuration
- "Using Fusion Middleware Control to Set the Connection Configuration"
- Using Fusion Middleware Control to Set the Process Configuration

5.3.1 Using Fusion Middleware Control to Set the Request Configuration

To specify the Oracle HTTP Server request configuration using Fusion Middleware Control, do the following:

1. Select **Administration** from the Oracle HTTP Server menu.
2. Select **Performance Directives** from the Administration menu. The Performance Directives page appears.
3. Enter the maximum number of requests in the Maximum Requests field (`MaxClients` directive). This setting limits the number of requests that can be dealt with at one time. The default and recommended value is 150. This is for Linux only.
4. Set the maximum requests per child process in the Maximum Request per Child Process field (`MaxRequestPerChild` directive). You can choose to have no limit, or a maximum number. If you choose to have a limit, enter the maximum number in the field.
5. Enter the request timeout value in the Request Timeout (seconds) field (`Timeout` directive). This value sets the maximum time, in seconds, Oracle HTTP Server waits to receive a GET request, the amount of time between receipt of TCP packets

on a POST or PUT request, and the amount of time between ACKs on transmissions of TCP packets in responses.

6. Review the settings. If the settings are correct, click **Apply** to apply the changes. If the settings are incorrect, or you decide to not apply the changes, click **Revert** to return to the original settings.
7. Restart Oracle HTTP Server. See Section 4.1.4.

The request configuration settings are saved, and shown on the Performance Directives page.

5.3.2 Using Fusion Middleware Control to Set the Connection Configuration

To specify the connection configuration using Fusion Middleware Control, do the following:

1. Select **Administration** from the Oracle HTTP Server menu.
2. Select **Performance Directives** from the Administration menu. The Performance Directives page appears.
3. Enter the maximum connection queue length in the Maximum Connection Queue Length field (`ListenBacklog` directive). This is the queue for pending connections. This is useful if the server is experiencing a TCP SYN overload, which causes numerous new connections to open up, but without completing the pending task.
4. Set the Multiple Requests per Connection field (`KeepAlive` directive) to indicate whether or not to allow multiple connections. If you choose to allow multiple connections, enter the number of seconds for timeout in the Allow With Connection Timeout field.

   The Allow With Connection Timeout value sets the number of seconds the server waits for a subsequent request before closing the connection. Once a request has been received, the specified value applies. The default is 15 seconds.
5. Review the settings. If the settings are correct, click **Apply** to apply the changes. If the settings are incorrect, or you decide to not apply the changes, click **Revert** to return to the original settings.
6. Restart Oracle HTTP Server. See Section 4.1.4.

The connection configuration settings are saved, and shown on the Performance Directives page.

5.3.3 Using Fusion Middleware Control to Set the Process Configuration

The child process and configuration settings impact the ability of the server to process requests. You may need to modify the settings as the number of requests increase or decrease to maintain a well-performing server.

For UNIX, the default number of child server processes is 2. For Microsoft Windows, the maximum number of threads to handle requests is 250.

To specify the process configuration using Fusion Middleware Control, do the following:

1. Select **Administration** from the Oracle HTTP Server menu.
2. Select **Performance Directives** from the Administration menu. The Performance Directives page appears.

3. Enter the number for the initial child server processes in the Initial Child Server Processes field (`StartServers` directive). This is the number of child server processes created when Oracle HTTP Server is started. The default is 2. This is for UNIX only.
4. Enter the number for the maximum idle threads in the Maximum Idle Threads field (`MaxSpareThreads` directive). An idle thread is a process that is running, but not handling a request.
5. Enter the number for the minimum idle threads in the Minimum Idle Threads field (`MinSpareThreads` directive).
6. Enter the number for the threads per child server process in the Threads per Child Server Process field (`ThreadsPerChild` directive).
7. Review the settings. If the settings are correct, click **Apply** to apply the changes. If the settings are incorrect, or you decide to not apply the changes, click **Revert** to return to the original settings.
8. Restart Oracle HTTP Server. See Section 4.1.4.

The process configuration settings are saved, and shown on the Performance Directives page.

5.4 Understanding Process Security

By default, Oracle HTTP Server runs as a non-root user (the user that installed Oracle Fusion Middleware). Therefore, on UNIX systems, if you plan on running Oracle HTTP Server on a privileged port (for example, port 80), you must enable Oracle HTTP Server to run as root. See "Starting Oracle HTTP Server on a Privileged Port" on page 4-4.

For additional security on UNIX, you can change the `User` directive in the `httpd.conf` configuration file to nobody. Be sure that the child processes can accomplish their tasks as the user nobody.

If your PL/SQL application is using the file system caching functionality in mod_plsql, then the httpd processes should have read and write privileges to the cache directory, specified through the parameter PlsqlCacheDirectory in *`ORACLE_INSTANCE`*`/config/OHS/<ohs_name>/mod_plsql/cache.conf`. By default, this parameter points to *`ORACLE_INSTANCE`*`/OHS/<ohs_name>`.

Finally, given that the cached content might contain sensitive data, the contents of the file system cache should be protected. So, although Oracle HTTP Server might run as nobody, access to the system as this user should be well-protected.

See Also: Section 3.9, "mod_plsql"

6 Managing Connectivity

Oracle HTTP Server comes configured with two listen ports: a non-SSL port (http) and an SSL port (https). The default, non-SSL port is 7777. If port 7777 is occupied, the next available port number, within a range of 7777-7877, is assigned. The default SSL port is 4443. Similarly, if port 4443 is occupied, the next available port number, within a range of 4443-4543, is assigned.

You can set these ports during installation or when creating a new Oracle HTTP Server component. For specifying ports at installation time, see the *Oracle Fusion Middleware Installation Guide* for Oracle WebTier. For specifying ports when creating a new Oracle HTTP Server component using opmnctl, refer to Section 4.2, "Creating a New Oracle HTTP Server Component".

Note: An additional SSL port (9999) is configured to run out-of-the-box in the `admin.conf` file. It is called *Proxy MBean* or *Admin port* and is used internally by Oracle HTTP Server to communicate with Fusion Middleware Control.

This chapter includes the following sections:

- Section 6.1, "Viewing Port Number Usage"
- Section 6.2, "Managing Ports"
- Section 6.3, "Configuring Virtual Hosts"

6.1 Viewing Port Number Usage

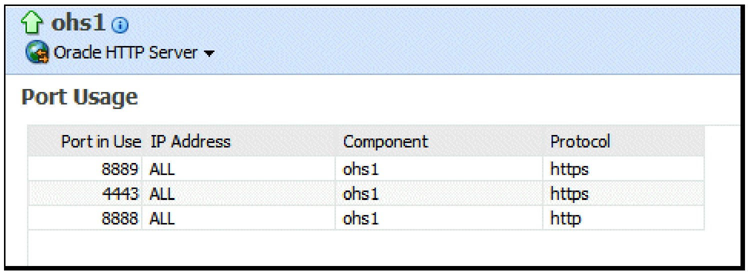
This section describes how to view ports using Fusion Middleware Control.

6.1.1 Using the Fusion Middleware Control to View Port Number Usage

To view the port number usage using Fusion Middleware Control, do the following:

1. Navigate to the Oracle HTTP Server home page.
2. Select **Port Usage** from the Oracle HTTP Server menu.

   The Port Usage detail page shows the component, the ports that are in use, the IP address the ports are bound to, and the protocol being used, as illustrated in the following figure:



Port Usage

Port in Use	IP Address	Component	Protocol
8889	ALL	ohs1	https
4443	ALL	ohs1	https
8888	ALL	ohs1	http

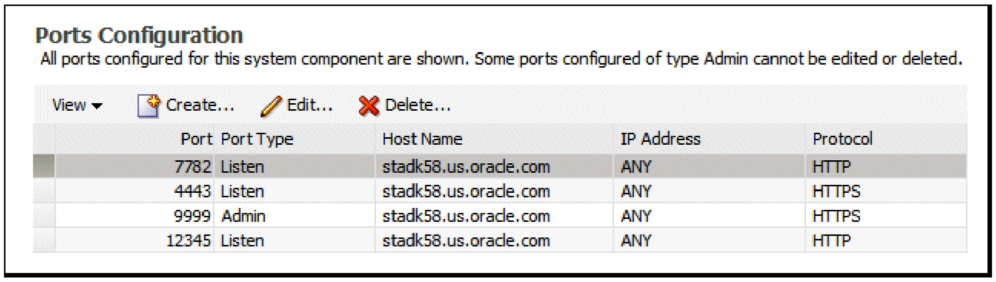
6.2 Managing Ports

The ports used by Oracle HTTP Server can be set during and after installation. In addition, you can change the port numbers, as needed. This section describes how to create, edit, and delete ports using Fusion Middleware Control.

Caution: The Oracle HTTP Server administration (proxy MBean) virtual host and its configuration, defined in the `admin.conf` file, must not be edited with the WebLogic Scripting Tool (WLST).

See Also: "Changing the Oracle HTTP Server Listen Ports" in the *Oracle Fusion Middleware Administrator's Guide*.

- Using Fusion Middleware Control to Create Ports
- Using Fusion Middleware Control to Edit Ports
- Updating the Registration of Oracle HTTP Server with a WebLogic Domain After Changing the Administration Port



Ports Configuration

All ports configured for this system component are shown. Some ports configured of type Admin cannot be edited or deleted.

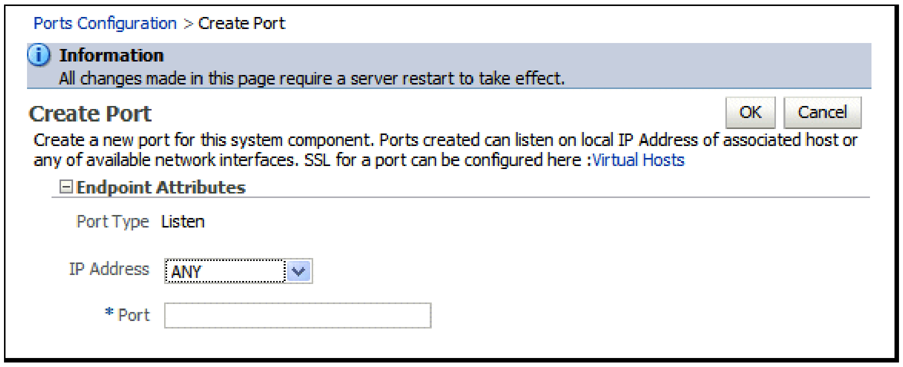
Port	Port Type	Host Name	IP Address	Protocol
7782	Listen	stadk58.us.oracle.com	ANY	HTTP
4443	Listen	stadk58.us.oracle.com	ANY	HTTPS
9999	Admin	stadk58.us.oracle.com	ANY	HTTPS
12345	Listen	stadk58.us.oracle.com	ANY	HTTP

Note: When deleting a port, if there is a virtual host configured to use the port you want to delete, you must first delete that virtual host before deleting the port.

6.2.1 Using Fusion Middleware Control to Create Ports

To create ports using Fusion Middleware Control, do the following:

1. Navigate to the Oracle HTTP Server home page.
2. Select **Administration** from the Oracle HTTP Server menu.
3. Select **Ports Configuration** from the Administration menu.
4. Click **Create**.



Ports Configuration > Create Port

Information
All changes made in this page require a server restart to take effect.

Create Port

Create a new port for this system component. Ports created can listen on local IP Address of associated host or any of available network interfaces. SSL for a port can be configured here :Virtual Hosts

Endpoint Attributes

Port Type Listen

IP Address ANY

* Port

5. Use the IP Address menu to select an IP address for the new port. Ports can listen on a local IP Address of an associated host or on any available network interfaces.

   SSL for a port can be configured on the Virtual Hosts page, as described in Section 6.3.2, "Using Fusion Middleware Control to Configure Virtual Hosts".

6. Use the **Port** field to enter the port number.
7. Click **OK**.
8. Restart Oracle HTTP Server. See Section 4.1.4.

Note: If you change the port or make other changes that affect the URL, such as changing the hostname, enabling or disabling SSL, you need to re-register partner applications with the SSO server using the new URL.

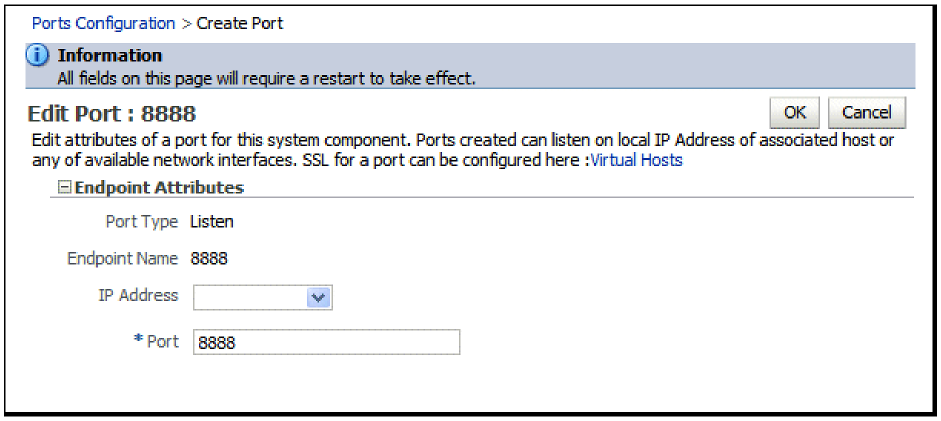
6.2.2 Using Fusion Middleware Control to Edit Ports

To create the ports using Fusion Middleware Control, do the following:

1. Navigate to the Oracle HTTP Server home page.
2. Select **Administration** from the Oracle HTTP Server menu.
3. Select **Ports Configuration** from the Administration menu.
4. Select the port for which you want to change the port number.

   When editing a port number, the Admin port cannot be edited using Fusion Middleware Control. Although this is a port Oracle HTTP Server uses for its internal communication with Fusion Middleware Control, in most of the cases it does not need to be changed. If you really want to change it, manually edit the `*ORACLE_INSTANCE*/config/OHS/<ohs_name>/admin.conf` file. And also refer to Section 6.2.3, "Updating the Registration of Oracle HTTP Server with a WebLogic Domain After Changing the Administration Port" for additional necessary steps after the Admin port is changed.

5. Click **Edit**.



Ports Configuration > Create Port

Information
All fields on this page will require a restart to take effect.

Edit Port : 8888

Edit attributes of a port for this system component. Ports created can listen on local IP Address of associated host or any of available network interfaces. SSL for a port can be configured here :Virtual Hosts

Endpoint Attributes

Port Type	Listen
Endpoint Name	8888
IP Address	
* Port	8888

6. Edit the IP Address and/or Port number for the port.

   SSL for a port can be configured on the Virtual Hosts page, as described in Section 6.3.2, "Using Fusion Middleware Control to Configure Virtual Hosts".

7. Click **OK**.

8. Restart Oracle HTTP Server. See Section 4.1.4.

Note: If you change the port or make other changes that affect the URL, such as changing the hostname, enabling or disabling SSL, you need to re-register partner applications with the SSO server using the new URL.

6.2.3 Updating the Registration of Oracle HTTP Server with a WebLogic Domain After Changing the Administration Port

In an Oracle Instance that is registered with a WebLogic domain, if the Oracle HTTP Server administration port (proxy MBean port in the `admin.conf` file) is changed after creating the component, then you must update the component registration with the WebLogic domain using the `opmnctl updatecomponentregistration` command, as follows:

```
opmnctl updatecomponentregistration -componentType OHS -componentName name
-proxyPort port
```

For example, if the proxy port of an Oracle HTTP Server component named `ohs2` has been changed to 8787, then use the following command:

```
opmnctl updatecomponentregistration -componentType OHS -componentName ohs2
-proxyPort 8787
```

6.3 Configuring Virtual Hosts

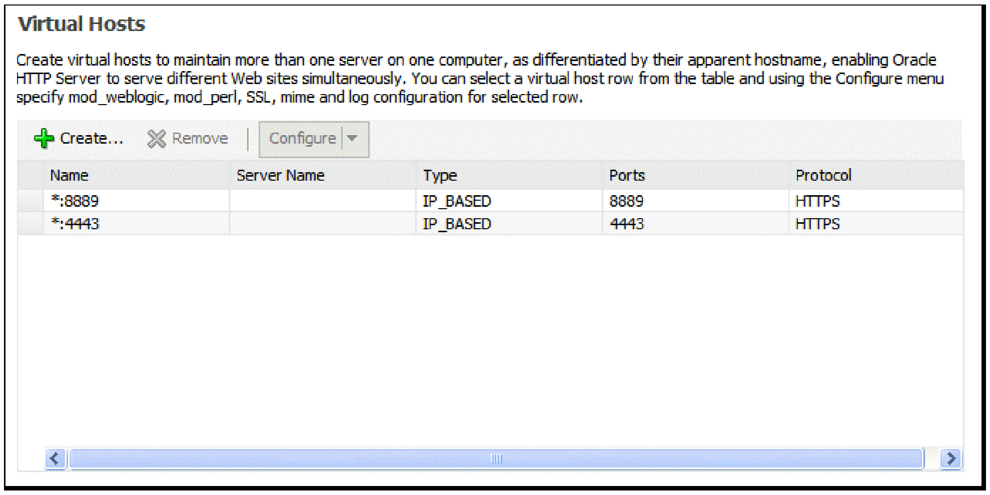
You can create virtual hosts to run more than one Web site (such as `www.company1.com` and `www.company2.com`) on a single machine. Virtual hosts can be *IP-based*, meaning that you have a different IP address for every Web site, or *name-based*, meaning that you have multiple names running on each IP address. The fact that they are running on the same physical server is not apparent to the end user.

Caution: The Oracle HTTP Server administration (proxy MBean) virtual host and its configuration, defined in the `admin.conf` file, must not be edited with the WebLogic Scripting Tool (WLST).

This section describes how to create and edit virtual hosts using Fusion Middleware Control.

- Using Fusion Middleware Control to Create Virtual Hosts
- Using Fusion Middleware Control to Configure Virtual Hosts

See Also: For more information about virtual hosts, refer to the Apache documentation.



Virtual Hosts

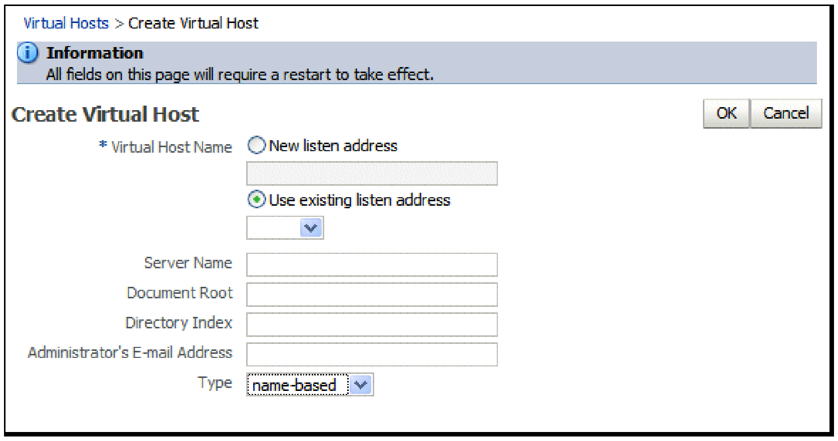
Create virtual hosts to maintain more than one server on one computer, as differentiated by their apparent hostname, enabling Oracle HTTP Server to serve different Web sites simultaneously. You can select a virtual host row from the table and using the Configure menu specify mod_weblogic, mod_perl, SSL, mime and log configuration for selected row.

Name	Server Name	Type	Ports	Protocol
*:8889		IP_BASED	8889	HTTPS
*:4443		IP_BASED	4443	HTTPS

6.3.1 Using Fusion Middleware Control to Create Virtual Hosts

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

1. Navigate to the Oracle HTTP Server home page.
2. Select **Administration** from the Oracle HTTP Server menu.
3. Select **Virtual Hosts** from the Administration menu.
4. Click **Create**.



Virtual Hosts > Create Virtual Host

Information
All fields on this page will require a restart to take effect.

Create Virtual Host

* Virtual Host Name	New listen address
	[✓ Use existing listen address]
Server Name	
Document Root	
Directory Index	
Administrator's E-mail Address	
Type	name-based

5. Enter a name for the virtual host field and then choose whether to enter a new listen address or to use an existing listen address.
   - New listen address - use this option when you want to create a virtual host that maps to a specific hostname or IP address, for example mymachine.com:8080. This will create following type NameVirtualHost and VirtualHost directives:

     ```
     NameVirtualHost mymachine.com:8080
     <VirtualHost mymachine.com:8080>
     ```

   - Use existing listen address - use this option when you want to create a virtual host using an existing listen port and the one that maps to all IP addresses. This will create following type `VirtualHost` directive:

     ```
     <VirtualHost *:8080>
     ```

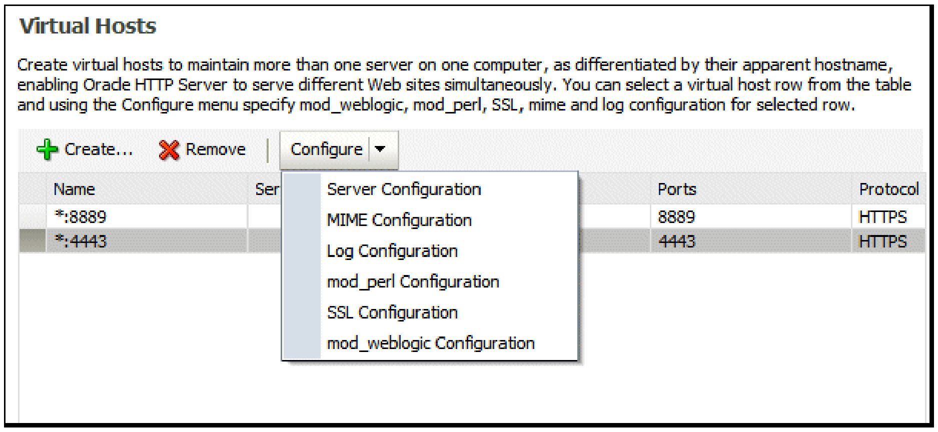
6. Enter the remaining attributes for the new virtual host.
7. Use the Type field to select whether the virtual host will be IP-based or name-based.
8. Click **OK**.
9. Restart Oracle HTTP Server. See Section 4.1.4.

6.3.2 Using Fusion Middleware Control to Configure Virtual Hosts

You can use the options on the Configure menu to specify Server, MIME, Log, mod_perl, SSL, and mod_wl_ohs configuration for a selected virtual host.

To configure a virtual host using Fusion Middleware Control, do the following:

1. Navigate to the Oracle HTTP Server home page.
2. Select **Administration** from the Oracle HTTP Server menu.
3. Select **Virtual Hosts** from the Administration menu.
4. Highlight an existing virtual host in the table.
5. Click **Configure**.


Virtual Hosts

Create virtual hosts to maintain more than one server on one computer, as differentiated by their apparent hostname, enabling Oracle HTTP Server to serve different Web sites simultaneously. You can select a virtual host row from the table and using the Configure menu specify mod_weblogic, mod_perl, SSL, mime and log configuration for selected row.

Name	Ser	Ports	Protocol
*:8889		8889	HTTPS
*:4443		4443	HTTPS

6. Select one of the following options from Configure menu to open its corresponding configuration page. The values on these pages apply only to the virtual host. If the fields are blank, the virtual host uses the values configured at the server level.

- Server Configuration – Configure basic virtual host properties, such as document root directory, installed modules, and aliases. See Section 4.3.1, "Using Fusion Middleware Control to Specify Server Properties."
- MIME Configuration – Configure MIME settings, which are used by Oracle HTTP Server to interpret file types, encodings, and languages. Section 4.4.2, "Configuring MIME Settings."
- Log Configuration – Configure access logs that will record all requests processed by the virtual host. The logs contain basic information about every HTTP transaction handled by the virtual host. See Section 7.2, "Configuring Oracle HTTP Server Logs."
- mod_perl Configuration – Configure the mod_perl module to embed the Perl interpreter into the virtual host, thereby eliminating startup overhead and enabling you to write modules in Perl. This module is disabled, by default. See Section 4.4.3, "Configuring the mod_perl Module."
- SSL Configuration – For instructions on configuring SSL using Fusion Middleware Control, see "Enabling SSL for Oracle HTTP Server Virtual Hosts" in the Oracle Fusion Middleware Administrator's Guide.
- mod_wl_ohs Configuration – Configure the mod_wl_ohs module to allow requests to be proxied from an Oracle HTTP Server to Oracle WebLogic Server. See Section 4.4.4, "Configuring the mod_wl_ohs Module."

7. Review the settings on each configuration page. If the settings are correct, click **OK** to apply the changes. If the settings are incorrect, or you decide to not apply the changes, click **Cancel** to return to the original settings.
8. Restart Oracle HTTP Server. See Section 4.1.4.

7 Managing Oracle HTTP Server Logs

Oracle HTTP Server generate log files containing messages that record all types of events, including startup and shutdown information, errors, warning messages, access information on HTTP requests, and additional information. This chapter describes how to find information about the cause of an error and its corrective action, to view and manage log files to assist in monitoring system activity and to diagnose problems.

This chapter includes the following sections:

- Section 7.1, "Introducing Server Logs"
- Section 7.2, "Configuring Oracle HTTP Server Logs"
- Section 7.3, "Log Directives for Oracle HTTP Server"
- Section 7.4, "Viewing Oracle HTTP Server Logs"

7.1 Introducing Server Logs

There are two types of logs for Oracle HTTP Server:

- Error logs, which record server problems.
- Access logs, which record which components and applications are being accessed and by whom.

You can view Oracle Fusion Middleware log files using either Fusion Middleware Control or a text editor. The log files for Oracle HTTP Server are located in the following directory:

```
ORACLE_INSTANCE/diagnostics/logs/OHS/<ohs_name>
```

See Also: For information about searching and viewing log files, see *Oracle Fusion Middleware Administrator's Guide*

7.1.1 About Error Logs

Oracle HTTP Server enables you to choose the format in which you want to generate log messages. You can choose to generate log messages in the legacy Apache message format, or use Oracle Diagnostic Logging (ODL) to generate log messages in text or XML-formatted logs, which complies with Oracle standards for generating error log messages.

By default, Oracle HTTP Server error logs use ODL for generating diagnostic messages. It provides a common format for all diagnostic messages and log files, and a mechanism for correlating the diagnostic messages from various components across Oracle Fusion Middleware.

The default name of the error log file is `<ohs_name>.log`.

7.1.2 About Access Logs

Access logs record all requests processed by the server. The logs contain basic information about every HTTP transaction handled by the server. The access log contains the following information:

- Host name
- Remote log name
- Remote user and time
- Request
- Response code
- Number of transferred bytes

The default name of the access log file is `access_log`.

7.1.2.1 Log Format

You can specify the information to include in the access log, and the manner in which it is written. The default format is the Common Log Format (CLF).

The CLF format contains the following fields:

```
host ident authuser date request status bytes
```

- host: This is the client domain name or its IP number. Use `%h` to specify the host field in the log.
- ident: If IdentityCheck is enabled and the client system runs identd, this is the client identity information. Use `%i` to specify the client identity field in the log.
- authuser: This is the user ID for the authorized user. Use `%a` to specify the authorized user field in the log.
- date: This is the date and time of the request in the day/month/year:hour:minute:second format. Use `%t` to specify date and time in the log.
- request: This is the request line, in double quotes, from the client. Use `%r` to specify request in the log.
- status: This is the three-digit status code returned to the client. Use `%s` to specify the status in the log. If the request will be forwarded from another server, use `%>s` to specify the last server in the log.
- bytes: This is the number of bytes, excluding headers, returned to the client. Use `%b` to specify number of bytes in the log. Use `%i` to include the header in the log.

See Also: Access Log in the Apache Server documentation.

7.1.3 About Log Rotation

It is important to have log files periodically rotated on a moderately busy server. Oracle HTTP Server supports two types of log rotation policies: size-based and time-based. You can configure both error log and access log to use either one of these two rotation polices.

In addition to the `rotatelogs` binary from Apache, Oracle HTTP Server comes with another rotation binary called `odl_rotatelogs`, which provides all the functionality of `rotatelogs` binary plus the extra functionality of log retention. By default, Oracle HTTP Server uses `odl_rotatelogs` for both error and access logs. `odl_rotatelogs` takes the following arguments:

```
odl_rotatelogs
    [-u:<utc offset in seconds>]
    LOGFILENAME
    {size based rotation options OR time based rotation options}
```

- Size-based rotation options: `<maxFileSize>M [<allFilesSize>M]`

  For example, when configured as `10M 70M`, the rotation will happen whenever log file reaches 10MB in size, and a total of 70MB is allowed for all log files (a maximum of 70/10=7 log files will be retained).

- Time-based rotation options: `<frequency in sec> [<retentionTime in sec>] [<startTime in YYYY-MM-DDThh:mm:ss>]`

  For example, when configured as `43200 604800 2009-05-08T10:53:29`, the rotation will happen every 43200 seconds (that is, 12 hours), rotated log files will be retained for maximum of 604800 seconds (7 days), starting from May 5, 2009 at 10:53:29

7.2 Configuring Oracle HTTP Server Logs

You can use Fusion Middleware Control to configure error and access logs. The following logging tasks can be set from the Log Configuration page:

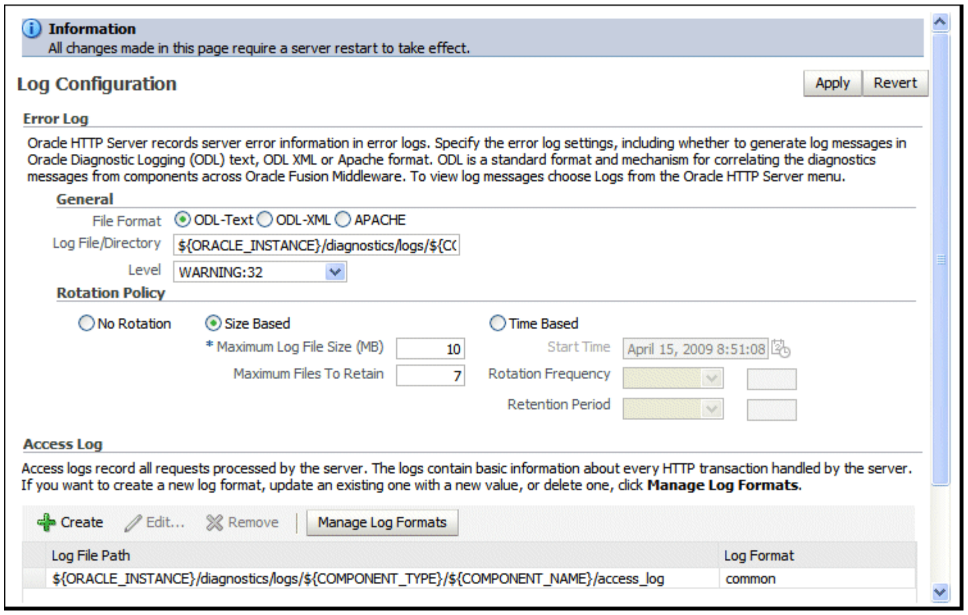
- Using Fusion Middleware Control to Configure Error Logs
- Using Fusion Middleware Control to Configure Access Logs

7.2.1 Using Fusion Middleware Control to Configure Error Logs

To configure an error log for Oracle HTTP Server using Fusion Middleware Control, do the following:

1. Navigate to the Oracle HTTP Server home page.
2. Select **Log Configuration** from the Administration menu.

   The Log Configuration page is displayed, as shown in the following figure.


Information
All changes made in this page require a server restart to take effect.

Log Configuration

Error Log

Oracle HTTP Server records server error information in error logs. Specify the error log settings, including whether to generate log messages in Oracle Diagnostic Logging (ODL) text, ODL XML or Apache format. ODL is a standard format and mechanism for correlating the diagnostics messages from components across Oracle Fusion Middleware. To view log messages choose Logs from the Oracle HTTP Server menu.

General

File Format [✓ ODL-Text] ODL-XML APACHE
Log File/Directory ${ORACLE_INSTANCE}/diagnostics/logs/${C(
Level WARNING:32

Rotation Policy

No Rotation [✓ Size Based] Time Based
* Maximum Log File Size (MB) 10
Maximum Files To Retain 7
Start Time April 15, 2009 8:51:08
Rotation Frequency
Retention Period

Access Log

Access logs record all requests processed by the server. The logs contain basic information about every HTTP transaction handled by the server. If you want to create a new log format, update an existing one with a new value, or delete one, click **Manage Log Formats.**

Log File Path	Log Format
${ORACLE_INSTANCE}/diagnostics/logs/${COMPONENT_TYPE}/${COMPONENT_NAME}/access_log	common

3. The following error log configuration tasks can be set from this page:
   - Configuring the Error Log Format and Location
   - Configuring the Error Log Level
   - Configuring Error Log Rotation Policy

7.2.1.1 Configuring the Error Log Format and Location

Oracle HTTP Server by default uses ODL-Text as the error log format and creates the log file with the name `<ohs_name>.log` under *ORACLE_INSTANCE*`/diagnostics/logs/OHS/ <ohs_name>` directory. To use a different format or log location, do the following:

1. From the Log Configuration page, navigate to the General section under the Error Log section.
2. Select the desired file format. Although both ODL-Text and ODL-XML formats provide the same information, the ODL-XML file includes XML elements and wrappers, and so may be easier to read.
   - ODL-Text – the format of the diagnostic messages conform to an Oracle standard and are written in text format.
   - ODL-XML – the format of the diagnostic messages conform to an Oracle standard and are written in XML format.
   - Apache – the format of the diagnostic messages conform to the legacy Apache message format.
3. Enter a path for the error log in the Log File/Directory field. This directory must exist before you enter it here.
4. Review the settings. If the settings are correct, click **Apply** to apply the changes. If the settings are incorrect, or you decide to not apply the changes, click **Revert** to return to the original settings.
5. Restart Oracle HTTP Server. See Section 4.1.4.

7.2.1.2 Configuring the Error Log Level

You can configure the amount and type of information written to log files by specifying the message type and level. Error log level for Oracle HTTP Server by default is configured to WARNING:32. To use a different error log level do the following:

1. From the Log Configuration page, navigate to the General section under the Error Log section.
2. Select a level for the logging from the Level menu. The higher the log level, the more information that is included in the log.
3. Review the settings. If the settings are correct, click **Apply** to apply the changes. If the settings are incorrect, or you decide to not apply the changes, click **Revert** to return to the original settings.
4. Restart Oracle HTTP Server. See Section 4.1.4.

Note: The log levels are different for the Apache log format from ODL-Text and the ODL-XML log format.

- For details on ODL log levels, refer to "Setting the Level of Information Written to Log Files" in the *Oracle Fusion Middleware Administrator's Guide*.
- For details on Apache log levels, refer to the LogLevel Directive in the Apache Server documentation.

7.2.1.3 Configuring Error Log Rotation Policy

Log rotation policy for error logs can either be time-based, such as once a week, or sized-based, such as 120MB. By default, the error log file is rotated when it reaches 10 MB in size and a maximum of 7 error log files will be retained. To use a different rotation policy for error log file do, the following:

1. From the Log Configuration page, navigate to the General section under the Error Log section.
2. Select a rotation policy.
   - No Rotation – if you do not want to have the log file rotated ever.
   - Size Based – rotate the log file whenever it reaches a configured size. Set the maximum size for the log file in Maximum Log File Size (MB) field and the maximum number of error log files to retain in Maximum Files to Retain field.
   - Time Based – rotate the log file whenever configured time is reached. Set the start time, rotation frequency, and retention period.
3. Review the settings. If the settings are correct, click **Apply** to apply the changes. If the settings are incorrect, or you decide to not apply the changes, click **Revert** to return to the original settings.
4. Restart Oracle HTTP Server. See Section 4.1.4.

7.2.2 Using Fusion Middleware Control to Configure Access Logs

To configure an access log for Oracle HTTP Server using Fusion Middleware Control, do the following:

1. Navigate to the Oracle HTTP Server home page.

2. Select **Log Configuration** from the Administration menu.

The following access log configuration tasks can be set from this page:

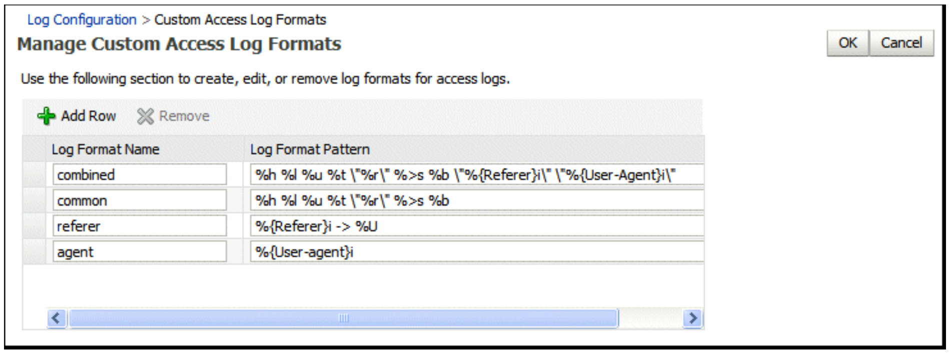
- Configuring the Access Log Format
- Configuring the Access Log File

7.2.2.1 Configuring the Access Log Format

Log format specifies the information included in the access log file and the manner in which it is written. To add a new access log format or to edit or remove an existing format, do the following:

1. From the Log Configuration page, navigate to the Access Log section.
2. Click **Manage Log Formats**.

   The Manage Custom Access Log Formats page is displayed, as shown in the following figure.



Manage Custom Access Log Formats

Use the following section to create, edit, or remove log formats for access logs.

Log Format Name	Log Format Pattern
combined	%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\"
common	%h %l %u %t \"%r\" %>s %b
referer	%{Referer}i -> %U
agent	%{User-agent}i

3. Select an existing format to change or remove, or click **Add Row** to create a new format.
4. If you choose to create a new format, then enter the new log format in the Log Format Name field and the log format in the Log Format Pattern field.

   See Also: Refer to the Apache documentation for information about log format directives.

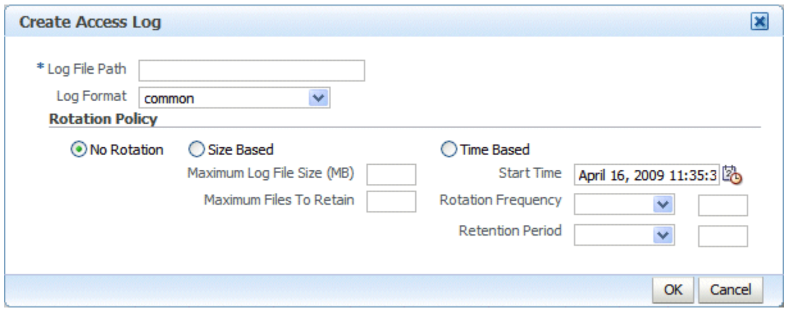
5. Click **OK** to save the new format.

7.2.2.2 Configuring the Access Log File

To configure an access log for file Oracle HTTP Server, do the following:

1. From the Log Configuration page, navigate to the Access Log section.
2. Click **Create** to create a new access log, or select a row from the table and click **Edit** button to edit an existing access log file.

   The Create or Edit Access Log page is displayed.



Log Format: common

Rotation Policy: [✓ No Rotation] Size Based Time Based

Start Time: April 16, 2009 11:35:3

3. Enter the path for the access log in the Log File Path field. This directory must exist before you enter it.
4. Select an existing access log format from the Log Format menu.
5. Select a rotation policy.
   - No Rotation – if you do not want to have the log file rotated ever.
   - Size Based – rotate the log file whenever it reaches a configured size. Set the maximum size for the log file in Maximum Log File Size (MB) field and the maximum number of error log files to retain in Maximum Files to Retain field.
   - Time Based – rotate the log file whenever configured time is reached. Set the start time, rotation frequency, and retention period.
6. Click **OK** to continue.

   Note that you can create multiple access log files.

7.3 Log Directives for Oracle HTTP Server

This section discuss Oracle HTTP Server error and access log-related directives in the `httpd.conf` file. The directives discussed are:

- Oracle Diagnostic Logging Directives
- Apache Log Directives

7.3.1 Oracle Diagnostic Logging Directives

Oracle HTTP Server by default uses Oracle Diagnostic Logging (ODL) for generating diagnostic messages. The following directives are used to setup logging using ODL:

- OraLogMode
- OraLogDir
- OraLogSeverity
- OraLogRotationParams

7.3.1.1 OraLogMode

Enables you to choose the format in which you want to generate log messages. You can choose to generate log messages in the legacy Apache, ODL text, or ODL XML format.

```
OraLogMode Apache | ODL-Text | ODL-XML
```

Default value: `ODL-Text`

For example: `OraLogMode ODL-XML`

Note: The Apache log directives `ErrorLog` and `LogLevel` are only effective when `OraLogMode` is set to `Apache`. When `OraLogMode` is set to either `ODL-Text` or `ODL-XML`, the `ErrorLog` and `LogLevel` directives are ignored.

7.3.1.2 OraLogDir

Specifies the path to the directory that contains all log files. This directory must exist.

This directive is used only when `OraLogMode` is set to either `ODL-Text` or `ODL-XML`. When `OraLogMode` is set to `Apache`, `OraLogDir` is ignored and `ErrorLog` is used instead.

```
OraLogDir <path>
```

Default value: *`ORACLE_INSTANCE`*`/diagnostics/logs/OHS/<ohs_name>`

For example: `OraLogDir /tmp/logs`

7.3.1.3 OraLogSeverity

Enables you to set message severity. The message severity specified with this directive is interpreted as the lowest desired message severity, and all messages of that severity level and higher are logged.

This directive is used only when `OraLogMode` is set to either `ODL-Text` or `ODL-XML`. When `OraLogMode` is set to `Apache`, `OraLogSeverity` is ignored and `LogLevel` is used instead.

```
OraLogSeverity <msg_type>[:msg_level]
```

Default value: `WARNING:32`

For example: `OraLogSeverity NOTIFICATION:16`

msg_type

Message types can be specified in upper or lower case, but appear in the message output in upper case. This parameter must be of one of the following values:

- INCIDENT_ERROR
- ERROR
- WARNING
- NOTIFICATION
- TRACE

msg_level

This parameter must be an integer in the range of 1–32, where 1 is the most severe, and 32 is the least severe. Using level 1 will result in fewer messages than using level 32.

7.3.1.4 OraLogRotationParams

Enables you to choose the rotation policy for an error log file. This directive is used only when `OraLogMode` is set to either `ODL-Text` or `ODL-XML`. When `OraLogMode` is set to `Apache`, `OraLogRotationParams` is ignored.

```
OraLogRotationParams <rotation_type> <rotation_policy>
```

Default value: `S 10:70`

For example: `OraLogRotationParams T  43200:604800 2009-05-08T10:53:29`

rotation_type

This parameter can either be `S` (for sized-based rotation) or `T` (for time-based rotation).

rotation_policy

When rotation_type is set to `S` (sized-based), set the rotation_policy parameter to:

`maxFileSize:allFilesSize` (in MB)

For example, when configured as `10:70`, the error log file is rotated whenever it reaches 10MB in size and a total of 70MB is allowed for all error log files (a maximum of 70/10=7 error log files will be retained).

When rotation_type is set to `T` (time-based), set the rotation_policy parameter to:

```
frequency(in sec) retentionTime(in sec) startTime(in
YYYY-MM-DDThh:mm:ss)
```

For example, when configured as `43200:604800 2009-05-08T10:53:29`, the error log is rotated every 43200 seconds (that is, 12 hours), rotated log files are retained for maximum of 604800 seconds (7 days) starting from May 5, 2009 at 10:53:29.

7.3.2 Apache Log Directives

Although Oracle HTTP Server uses ODL by default for error logs, you can configure the `OraLogMode` directive to `Apache` to generate error log messages in the legacy Apache message format. The following directives are discussed in this section:

- ErrorLog
- LogLevel
- LogFormat
- CustomLog

7.3.2.1 ErrorLog

The `ErrorLog` directive sets the name of the file where the server logs any errors it encounters. If the file-path is not absolute then it is assumed to be relative to the ServerRoot.

This directive is used only when `OraLogMode` is set to `Apache`. When `OraLogMode` is set to either `ODL-Text` or `ODL-XML`, `ErrorLog` is ignored and `OraLogDir` is used instead.

See Also: Refer to the Apache documentation for information about the `ErrorLog` directive.

7.3.2.2 LogLevel

The `LogLevel` directive adjusts the verbosity of the messages recorded in the error logs.

This directive is used only when `OraLogMode` is set to `Apache`. When `OraLogMode` is set to either `ODL-Text` or `ODL-XML`, `LogLevel` is ignored and `OraLogSeverity` is used instead.

See Also: Refer to the Apache documentation for information about the `LogLevel` directive.

7.3.2.3 LogFormat

The `LogFormat` directive specifies the format of the access log file. By default, Oracle HTTP Server comes with the following four access log formats defined:

```
LogFormat "%h %l %u %t \"%r\" %>s %b" common
LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\"" combined
LogFormat "%{Referer}i -> %U" referer
LogFormat "%{User-agent}i" agent
```

See Also: Refer to the Apache documentation for information about the `LogFormat` directive.

7.3.2.4 CustomLog

The `CustomLog` directive is used to log requests to the server. A log format is specified and the logging can optionally be made conditional on request characteristics using environment variables. By default, the access log file is configured to use the common log format.

See Also: Refer to the Apache documentation for information about the `CustomLog` directive.

7.4 Viewing Oracle HTTP Server Logs

You can search, view, and list Oracle HTTP Server log files using Fusion Middleware Control, or you can download a log file to your local client and view the log files using another tool.

You can also use the text editor of your choice to view Oracle HTTP Server log files directly from the *`ORACLE_INSTANCE`* directory. By default, Oracle HTTP Server log files are located in the *`ORACLE_INSTANCE`*`/diagnostics/logs/OHS/<ohs_name>` directory.

As discussed in Section 7.1, "Introducing Server Logs", there are mainly two types of log files for Oracle HTTP Server: error logs and access logs. The error log file is an important source of information for maintaining a well-performing server. The error log records all of the information about problem situations so that the system administrator can easily diagnose and fix the problems. The access log file contains basic information about every HTTP transaction that the server handles. This information can be used to generate statistical reports about the server's usage patterns.

See Also: For information about searching and viewing log files, see *Oracle Fusion Middleware Administrator's Guide*

8

Managing Application Security

This chapter contains an overview of Oracle HTTP Server security features, and provides configuration information for setting up a secure Web site.

This chapter includes the following sections:

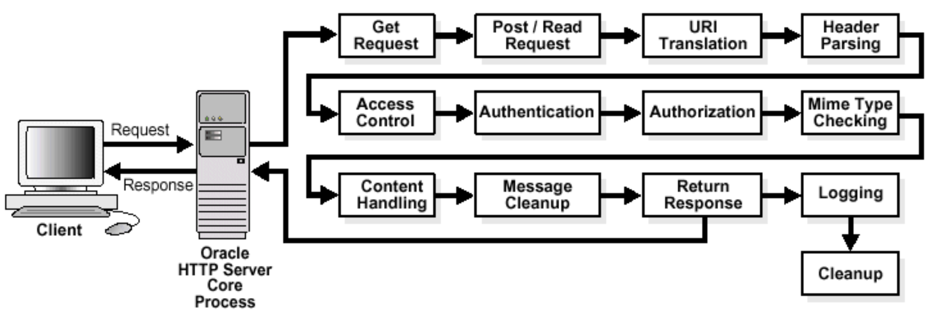
8.1 About Oracle HTTP Server Security

Security can be organized into the three categories of authentication, authorization, and confidentiality. Oracle HTTP Server provides support for all three of these categories. It is based on the Apache Web server, and its security infrastructure is primarily provided by the Apache modules, mod_auth_basic, mod_authn_file, mod_auth_user, and mod_authz_groupfile, and the Oracle modules, mod_ossl and mod_osso. The mod_auth_basic, mod_authn_file, mod_auth_user, and mod_authz_groupfile modules provide authentication based on user name and password pairs, while mod_authz_host controls access to the server based on the characteristics of a request, such as hostname or IP address, mod_ossl provides confidentiality and authentication with X.509 client certificates over SSL, and mod_osso enables single sign-on authentication for Web applications.

Based on the Apache model, Oracle HTTP Server provides access control, authentication, and authorization methods that can be configured with access control directives in the `httpd.conf` file. When URL requests arrive at Oracle HTTP Server, they are processed in a sequence of steps determined by server defaults and configuration parameters. The steps for handling URL requests are implemented through a module or plug-in architecture that is common to many Web listeners.

Figure 8–1 shows how URL requests are handled by the server. Each step in this process is handled by a server module depending on how the server is configured. For example, if basic authentication is used, then the steps labeled "Authentication" and "Authorization" in Figure 8–1 represent the processing of the Apache mod_auth_basic, mod_authn_file, mod_auth_user, and mod_authz_groupfile modules.

Figure 8–1 Steps for Handling URL Requests in Oracle HTTP Server



8.2 Classes of Users and Their Privileges

Oracle HTTP Server authorizes and authenticates users before allowing them to access, or modify resources on the server. The following are three classes of users that access the server using Oracle HTTP Server, and their privileges:

- Users that access the server without providing any authentication. They have access to unprotected resources only.
- Users that have been authenticated and potentially authorized by modules within Oracle HTTP Server. This includes users authenticated by Apache's mod_auth_basic, mod_authn_file, mod_auth_user, and mod_authz_groupfile modules and Oracle's mod_ossl. Such users have access to URLs defined in `http.conf` file.

  See Also: Section 8.4, "Authentication, Authorization and Access Control".

- Users that have been authenticated through mod_osso and Single Sign-On server. These users have access to resources allowed by Single Sign-On.

  See Also: *Oracle Fusion Middleware Security Guide*

8.3 Resources Protected

Oracle HTTP Server is configured to protect resources such as:

- Static content such as static HTML pages, graphics interchange format, `.gif`, files, and other static files that Oracle HTTP Server provides directly.
- CGI/FastCGI scripts, simple scripts or programs that Oracle HTTP Server invokes directly.
- Content generated by modules within Oracle HTTP Server. Modules such as mod_perl, mod_dms generate responses that are returned to the client.
- Oracle Application Server components that exist behind Oracle HTTP Server, including servlets and JSPs running with Oracle WebLogic Server that are accessed through mod_wl_ohs. Oracle HTTP Server forms the first line of authentication and authorization for these components, although further authentication may occur at the component level.

8.4 Authentication, Authorization and Access Control

Oracle HTTP Server provides user authentication and authorization at two stages:

- Access Control (stage one): This is based on the details of the incoming HTTP request and its headers, such as IP addresses or host names.
- User Authentication and Authorization (stage two): This is based on different criteria depending on the HTTP server configuration. The server can be configured to authenticate users with user name and password pairs that are checked against a list of known users and passwords. You can also configure the server to use single sign-on authentication for Web applications or X.509 client certificates over SSL.

8.4.1 Access Control

Access control refers to any means of controlling access to any resource.

See Also: Refer to the Apache documentation for more information on how to configure access control to resources.

8.4.2 User Authentication and Authorization

Authentication is any process by which you verify that someone is who they claim they are. Authorization is any process by which someone is allowed to be where they want to go, or to have information that they want to have.

8.4.2.1 Using Apache to Authenticate Users

Access control refers to any means of controlling access to any resource.

See Also: Refer to the Apache documentation for more information on how to authenticate users.

8.4.2.2 Using mod_osso to Authenticate Users

mod_osso enables single-sign on for Oracle HTTP Server. mod_osso examines incoming requests and determines whether the resource requested is protected, and if so, retrieves the Oracle HTTP Server cookie for the user.

Through mod_osso, Oracle HTTP Server becomes a single sign-on (SSO) partner application enabled to use SSO to authenticate users and obtain their identity using Oracle Single Sign-On, and to make user identities available to Web applications as an Apache header variable.

Using mod_osso, Web applications can register URLs that require SSO authentication. When Oracle HTTP Server receives URL requests, mod_osso detects which requests require SSO authentication and redirects them to the SSO server. Once SSO server authenticates the users, it passes the user's authenticated identity back to mod_osso in a secure token, or cookie. mod_osso retrieves the user's identity from the cookie and propagates the user's identity information to applications running in Oracle HTTP Server instance.mod_osso can propagate the user's identity information to applications running in CGI, and those running in Oracle WebLogic Server, and it can also authenticate users for access to static files.

See Also: *Oracle Fusion Middleware Security Guide*

9 Configuring mod_oradav

This chapter describes distributed authoring and versioning (DAV) concepts, and explains how to configure OraDAV using the mod_oradav module. The mod_oradav module enables you to use OraDAV to access content in files from a Web browser or a WebDAV client.

This chapter includes the following sections:

- Section 9.1, "Introduction to the mod_oradav Module"
- Section 9.2, "Configuring mod_oradav"
- Section 9.3, "WebDAV Security Considerations"
- Section 9.4, "OraDAV Performance Considerations"
- Section 9.5, "Globalization Support Considerations with OraDAV"
- Section 9.6, "Location of DAV Files"

9.1 Introduction to the mod_oradav Module

The mod_oradav module is an extended implementation of the Apache implementation of the WebDAV specification. The mod_oradav module is an OCI application written in C and is integrated with Oracle HTTP Server. The mod_oradav module enables WebDAV clients to connect to files, read and write content, and query and lock documents.

This section includes the following subsections:

- Section 9.1.1, "WebDAV"
- Section 9.1.2, "OraDAV"
- Section 9.1.3, "OraDAV Architecture"
- Section 9.1.4, "OraDAV Usage Model"
- Section 9.1.5, "PROPFIND Security"

9.1.1 WebDAV

WebDAV is a protocol extension to HTTP that supports distributed authoring and versioning. With WebDAV, the Internet becomes a transparent read and write medium, where content can be checked out, edited, and checked in to a URL address.

WebDAV enables collaboration among authors building Web sites. WebDAV also serves as universal read and write access protocol to arbitrary hierarchies of content, not necessarily Web sites. With WebDAV, you can save content to a URL provided by an Internet Service Provider (ISP), and then access and optionally change that content from various devices.

Note: When a WebDAV client first connects to Oracle HTTP Server, it must use the full ServerName string in the URL for the connection. Do not use an abbreviated form of the server name.

For example, if the server name value is `server1.example.com`, then connect to Oracle HTTP Server using the string `http://server1.example.com:7778`, not an abbreviated form such as `http://server1:7778`.

If you use an abbreviated form, the connection might succeed, but COPY and MOVE operations will fail to run, and generate BAD_GATEWAY errors.

9.1.2 OraDAV

OraDAV refers to the set of capabilities available through the mod_oradav module to Oracle Fusion Middleware users. Some OraDAV-specific terms include:

- OraDAV: Code in the Oracle HTTP Server that supports file-based DAV access. Apache DAV directives can be used with OraDAV.

  See Also: For more information about DAV directives, see the article written by Greg Stein (`gstein@lyra.org`) available at

  `http://www.webdav.org/mod_dav/install.html#apache`

- OraDAV API: Stored procedure calls that are used by the OraDAV driver to provide support for the following WebDAV functions over the Internet:
  - Reading and writing documents
  - Locking and unlocking documents
  - Managing, such as creating, populating, and deleting, hierarchies of information
  - Retrieving properties associated with documents
  - Associating properties with specific documents
- OraDAV driver: Stored procedure implementation of the OraDAV driver API that runs in Oracle and manages a repository.

The primary users of OraDAV are Oracle HTTP Server Web administrators and content editors. End users interact only indirectly with OraDAV through Web browsers or WebDAV client tools. OraDAV interaction requires the following proficiency:

- The Web administrator needs to know how to start and stop Oracle HTTP Server, and how to configure Oracle HTTP Server to direct URL traffic to an OraDAV driver.
- The content editors need to know how to connect to the server, and upload and retrieve files.

9.1.3 OraDAV Architecture

The mod_oradav module, which includes OraDAV, is part of the Oracle HTTP Server architecture. A simple form of the architecture is illustrated in Figure 9–1.

Figure 9–1 OraDAV Architecture

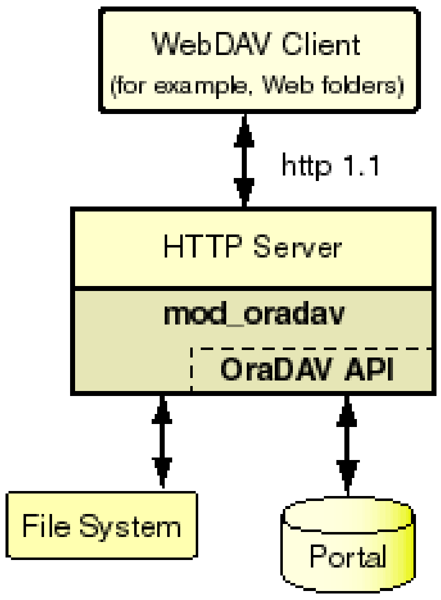


Figure 9–1 shows a WebDAV client, such as Web folders, passing HTTP requests to Oracle HTTP Server. If the request is for content stored in the file system, the mod_oradav module handles the access. If the request is for content stored in Oracle Portal, the OraDAV API handles the access.

The OraDAV API capabilities are equivalent to using the mod_oradav module running with a file system. The following HTTP methods are supported by the OraDAV API:

- COPY: Copies files within a Web site folder.
- DELETE: Deletes files within a Web site folder.
- MOVE: Moves files within a Web site folder.
- MKCOL: Makes a new directory.
- GET: Retrieves a file from the server. This method is not supported by Oracle Web Cache.
- PUT: Puts a file back to the server. This method is not supported by Oracle Web Cache.
- HEAD: Gets the header content of a file without retrieving the file.
- LOCK: Locks a file when the file is checked out. This method is not supported by Oracle Web Cache.
- UNLOCK: Unlocks a file after check in. This method is not supported by Oracle Web Cache.
- PROPFIND: Gets properties defined for a file.

- PROPPATCH: Sets the properties for a file.

The OraDAV API supports shared and exclusive locking, retrieving basic DAV properties, and defining and retrieving server-defined properties or client-defined properties. Set-based operations such as COPY, MOVE, DELETE can be done completely by a single call to an OraDAV driver.

9.1.4 OraDAV Usage Model

OraDAV usage can involve any combination of the following activities:

- Browsing: Read-only activity which uses WebDAV to access content on the file server. Its usage model is typical of a read-only Web site.

  The DAVOraReadOnly directive specifies whether or not WebDAV should be used in a read-only mode by WebDAV clients. A value of Off specifies that WebDAV clients function normally. A value of On prevents WebDAV clients from performing write operations while using WebDAV. It does allow read-only activity by Web browsers and WebDAV clients. The default is Off.

- Restructuring: Deleting, moving, and copying content. Restructuring is usually done infrequently by a restricted set of individuals who have write access to the WebDAV content.

  Restructuring has the same limitations and complications that one encounters when restructuring a file directory. In some cases, this directory hierarchy is owned and managed by one user. If the directory is shared, then the client doing restructuring is given sole access to the hierarchy through WebDAV exclusive locks.

- Editing: Modifying one or a small subset of resources in a hierarchy. Properly designed WebDAV clients use shared or exclusive locks on such resources to coordinate these activities.

- Property Management: Associating properties and attributes (for example, author) with documents for ease of lookup and for categorization. WebDAV clients assign properties to documents using the PROPPATCH directive and retrieve properties using the PROPFIND directive.

9.1.5 PROPFIND Security

The PROPFIND method can be used to list all the files in the DAV-enabled directory. For security reasons, it is probably best to protect the list of files from general read access.

An alternative is to limit the PROPFIND to a group of people, a set of domains, or a set of hosts, while the methods that modify content are limited to just a few authors. This scenario allows, for example, your company's employees to browse the files on the server, yet only a few people can change them. Anonymous (non-authenticated) visitors cannot browse or modify.

Finally, you can simply omit PROPFIND from the limits if your Web server is intended as a general, read-only repository of files. This allows anybody to arbitrarily browse the directories and to fetch the files.

9.2 Configuring mod_oradav

Use the Advanced Server Configuration page of Fusion Middleware Control to configure the mod_oradav module.

This section includes the following subsections:

- Section 9.2.1, "OraDAV Configuration Parameters"
- Section 9.2.2, "Using Fusion Middleware Control to Configure mod_oradav"

9.2.1 OraDAV Configuration Parameters

When Oracle Fusion Middleware is installed, all required OraDAV parameters are set to their default values. If the default values do not meet your needs, you can modify the values for required parameters and specify values for optional parameters. The OraDAV parameters in the `mod_oradav.conf` file start with "DAV" and "DAVParam".

Note: To configure the parameters use Fusion Middleware Control. Do not edit the `mod_oradav.conf` file directly. Doing so may harm your installation.

The `DAV` parameter indicates that a URL location is DAV-enabled. The `DAV` keyword is followed by one of the following values:

- `On` – indicates that mod_oradav is to use the local file system for content.
- `Oracle` – indicates that mod_oradav is to use OraDAV for all content.

The DAVParam parameters are used to specify name-value pairs. The required pairs are those that enable Oracle HTTP Server to connect to an Oracle database. These include the names OraService, OraUser, and OraPassword or OraAltPassword.

Each OraDAV driver can use the DAVParam mechanism to create its own driver-specific settings. All DAVParam name-value pairs are passed to the OraDAV driver. In addition to the OraDAV parameters, you should consider whether to specify additional DAV parameters, such as `DavMinTimeout`.

Example 9–1 shows the syntax to configure access to files on the local system. It specifies that the directory dav_portal under the Web server documents directory is to be DAV-enabled, along with all directories under dav_portal in the hierarchy. There must not be any symlinks defined on the dav_portal directory or any of its subdirectories.

Example 9–1 Configuring File System Access

```
<Location /dav_portal>
  DAV On
</Location>
```

The following recommendations should be considered when mapping containers under the root location:

- Do not map the root itself. For example, do not specify `<Location / >` in the `mod_oradav.conf` file.
- Do not map a container as a subelement in the hierarchy to another container. For example, do not specify the containers `<Location /project1>` and `<Location /project1/project2>`. It is acceptable to specify `<Location /project1>` and `<Location /project2>`.
- Do not create any symbolic links to the container or any location under the container in the hierarchy.

The OraDAV parameters are described in the following sections:

- ORAAllowIndexDetails
- ORAAltPassword
- ORACacheDirectory
- ORACacheMaxResourceSize
- ORACachePrunePercent
- ORACacheTotalSize
- ORAConnect
- ORAConnectSN
- ORAContainerName
- ORAException
- ORAGetSource
- ORALockExpirationPad
- ORAPackageName
- ORAPassword
- ORARootPrefix
- ORAService
- ORATraceEvents
- ORATraceLevel
- ORAUser

9.2.1.1 ORAAllowIndexDetails

In an Oracle HTTP Server environment that is not OraDAV-enabled, mod_dav does not respond to HTTP GET requests. Instead, normal Oracle HTTP Server mechanisms are used to respond to GET requests.

The `ORAAllowIndexDetails` parameter controls how OraDAV responds when a GET request is performed on a DAV collection and no `index.html` file is found in the directory. In a typical Oracle HTTP Server environment, a separate module takes control, automatically generating and returning to the client HTML that represents an "index" of the resources (files) in that collection.

An OraDAV-enabled Oracle HTTP Server performs similar actions when responding to a GET request on a collection. A description column containing links to more detailed information about each resource is included in the generated index when `ORAAllowIndexDetails` is set to `TRUE`.

The default is `FALSE`, in which case no description column appears in the generated index. If `?details` is used in a URL, it is ignored and the URL contents are returned.

9.2.1.2 ORAAltPassword

Specifies the password for the user specified by the `ORAUser` parameter. The `OraAltPassword` uses a base-64 encoded character string. This parameter provides an alternative if you do not want the password to appear in unencoded plain text with the `ORAUser` parameter.

If the ORAPassword parameter is not specified, `ORAAltPassword` parameter is used for the password.

9.2.1.3 ORACacheDirectory

Specifies the directory to use for disk caching operations. If you do not use this parameter, disk caching is not performed for OraDAV operations.

The specified directory must exist and be readable by Oracle HTTP Server, but cannot be visible to normal GET requests. If the directory is visible to normal GET requests, security measures could be bypassed by users accessing the cache directory.

The directory should be located on a file system that supports a last accessed time. On Microsoft Windows systems, this means using NTFS, not FAT, formatted partitions.

Do not use the cache directory for anything other than caching. Any files in the cache directory are subject to deletion.

If you use the `ORACacheDirectory` parameter, you must also use the ORACacheTotalSize parameter.

See Also: Section 9.4.1, "Using Disk Caching with OraDAV"

9.2.1.4 ORACacheMaxResourceSize

Specifies the maximum cacheable resource size for disk caching operations. You can specify KB (for kilobytes) or MB (for megabytes) after an integer. If you do not specify a unit after the integer, then the default unit is bytes.

This parameter enables Web administrators to prevent large media files from dominating the cache. The performance benefit of caching a large file is greater than from caching a small file.

Example 9–2 shows an example for `ORACacheMaxResourceSize`.

Example 9–2 ORACacheMaxResourceSize Parameter

```
DAVParam ORACacheMaxResourceSize 1024KB
```

The setting in Example 9–2 prevents OraDAV from caching any resource larger than 1 MB.

See Also: Section 9.4.1, "Using Disk Caching with OraDAV"

9.2.1.5 ORACachePrunePercent

Specifies the percentage of disk cache usage to be freed when the cache is full. When the disk cache is full, the oldest files in the cache are deleted until the cache disk usage is reduced by the `ORACachePrunePercent` value. The default value is 25.

See Also: Section 9.4.1, "Using Disk Caching with OraDAV"

9.2.1.6 ORACacheTotalSize

Specifies the size of the cache to use for disk caching operations. You can specify MB (for megabytes) or GB (for gigabytes) after an integer. If you do not specify a unit after the integer, the default unit is bytes.

Example 9–3 ORACacheTotalSize Parameter

```
DAVParam ORACacheTotalSize 1GB
```

If you use the ORACacheDirectory parameter, you must also use the `ORACacheTotalSize` parameter.

The `ORACacheTotalSize` value should be large enough to hold either a significant amount of your Web site, or all of the most frequently accessed files plus 25 percent more space. If the value is too small, overall performance degrades because of the extra work of writing BLOB data to the file system and deleting files to make room for newer cache requests.

The actual space utilized by the disk cache might sometimes exceed the `ORACacheTotalSize` value, possibly by as much as the ORACacheMaxResourceSize value. You should also be aware of file system block size issues that could cause the cache to use more disk space than the `ORACacheTotalSize` value.

See Also: Section 9.4.1, "Using Disk Caching with OraDAV"

9.2.1.7 ORAConnect

Specifies the Oracle database to connect to. The `ORAConnect` parameter lets you connect to a database that is not included in the `tnsnames.ora` file. The value must use the following format:

```
database_host:database_port:database_sid
```

Example 9–4 shows an example:

Example 9–4 ORAConnect Parameter

```
DAVParam ORAConnect my-pc.example.com:1521:mysid
```

To connect to an Oracle database, you must specify one, and no more than one, of the parameters ORAConnect, ORAConnectSN, or ORAService. To connect to a database included in the `tnsnames.ora` file, use the ORAService parameter.

9.2.1.8 ORAConnectSN

Specifies the Oracle database to connect to. The `ORAConnectSN` parameter lets you connect to a database that is not included in the `tnsnames.ora` file. The value for this parameter is a character string.The value must use the following format:

```
database-host:database-port:database-service-name
```

To connect to an Oracle database, you must specify one, and no more than one, of the parameters ORAConnect, or ORAService. To connect to a database included in the `tnsnames.ora` file, use the ORAService parameter.

9.2.1.9 ORAContainerName

Within the database user (schema) specified by the ORAUser parameter, there must exist a container, which is a set of PL/SQ packages and database tables that allow the storage of files in the database within a hierarchical structure. The `ORAContainerName` parameter specifies the name of the container to use for the location. The value for this parameter is a character string, up to 20 characters. For example, `<Location/project1>`.

9.2.1.10 ORAException

Writes PL/SQL stack dumps in the Oracle HTTP Server log file, error_log, in the event of an exception in the PL/SQL package. The values are `RAISE` or `NORAISE`. Default value is `NORAISE`.

Note: Although this parameter is useful for debugging purposes, it can use a large amount of disk space and can slow the performance of your system.

9.2.1.11 ORAGetSource

Applies only to file system access. It specifies one or more file extensions to identify types of files that are not to be run, but rather opened for editing. Include periods (.) with the file extension and use a comma to separate file extensions. The value for this parameter is enclosed within double quotation marks. For example:

```
".htm, .html, .jsp1, .jsp2"
```

The `ORAGetSource` parameter lets you open files for editing that are usually run as a result of a GET operation.

Note: The `.jsp` and `.sqljsp` files are by default opened for editing; you do not need to specify them in the `ORAGetSource` parameter.

9.2.1.12 ORALockExpirationPad

Intended to be used in high-latency network environments to adjust the refresh lock behavior in Microsoft Office. Microsoft Office attempts to refresh locks on DAV resources just before the lock is set to expire. If there is network congestion between the Microsoft Office client and the DAV server, the refresh request might arrive after the lock has expired. The value is the number of seconds. The default value is 0.

OraDAV periodically looks for locks on resources that have expired and deletes those locks. The `ORALockExpirationPad` parameter can be used to provide some additional time between when a lock expires and when that lock is deleted. For example, if `ORALockExpirationPad` is set to 120, OraDAV does not actually delete locks for at least two minutes after the expiration time.

9.2.1.13 ORAPackageName

Identifies the OraDAV driver implementation that is to be called when issuing OraDAV commands. The default is the OraDAV driver, which is the ORDSYS.DAV_API_DRIVER package.

9.2.1.14 ORAPassword

Specifies the password for the user specified by the ORAUser parameter.

If you do not want to specify the password as an unencoded text string with the `ORAPassword` parameter, you can specify the password as a base-64 encoded string with the ORAAltPassword parameter.

9.2.1.15 ORARootPrefix

Specifies the directory within the database repository to use as the root. If this parameter is specified, WebDAV clients see this directory as the root and are not be able to see the repository directories that lead up to it. Do not include a trailing slash (/) in the value.

In Example 9–5, assume that the database repository contains the directory /first/second/third/fourth.

Example 9–5 ORARootPrefix Parameter

```
DAVParam ORARootPrefix /first/second
```

WebDAV clients can view the /third directory and can navigate to the /third/fourth directory, but will not be able to view or navigate to the /first or /first/second directories.

9.2.1.16 ORAService

Specifies the Oracle database to connect to. The specified value must match a SID value in the `tnsnames.ora` file.

To connect to an Oracle database, you must specify one, and no more than one, of the parameters ORAConnect, ORAConnectSN, or ORAService. To connect to a database that is not included in the tnsnames.ora file, use the ORAConnect or ORAConnectSN parameter.

9.2.1.17 ORATraceEvents

Specifies the types of events to be recorded in the Oracle HTTP Server error log for debugging. The value for this parameter is one of the following:

- `getsource`: traces GET activity against the file system
- `hreftoutf8`: traces the HREF conversion from the native character set to UTF-8
- `request`: traces DAV requests, responses, and status values handled by mod_oradav

Note: Although this parameter is useful for debugging purposes, it can use a large amount of disk space and can slow the performance of your system.

9.2.1.18 ORATraceLevel

Specifies the level of debugging (trace statements) that will be entered in the Oracle HTTP Server error log. The lowest level is `0` (the default), which performs no tracing. The highest level is `4`, which performs maximum tracing. The value for this parameter is an integer between `0` and `4`.

The higher the number for the debugging level, the more information is written to the error log file.

Note: Although setting this parameter to a high number is useful for debugging purposes, it can use a large amount of disk space and can slow the performance of your system.

9.2.1.19 ORAUser

Specifies the database user (schema) to use when connecting to the service specified by the ORAConnect, ORAConnectSN, or ORAService parameter.

This user must have the following privileges:

- CONNECT
- RESOURCE
- CREATE TABLESPACE

- DROP TABLESPACE
- CREATE ANY TRIGGER

9.2.2 Using Fusion Middleware Control to Configure mod_oradav

On the Advanced Server Configuration page of Fusion Middleware Control, you can enter parameters within a `<Location>` container directive in the `mod_oradav.conf` file. The `<Location>` container directive specifies the DAV-enabled URL. The `DAV` keyword is followed by the parameter `On`, which instructs mod_dav to use the local file system for content.

The following example specifies that the directory `myfiles` under the Web server documents directory (`htdocs` by default) to be DAV-enabled, along with all directories under `myfiles` in the hierarchy. There must not be any symbolic links defined on the `myfiles` directory or any of its subdirectories.

```
<Location /myfiles>
   DAV On
</Location>
```

9.3 WebDAV Security Considerations

Because WebDAV enables read/write capabilities, Internet users can write to your Web site or to an Oracle repository. A major concern is preventing users from placing an inappropriate file, such as a Trojan horse, that can run on the Web server system. If the WebDAV configuration and authorization is not set up properly, an inappropriate file from the file system can be run. However, mod_oradav is disabled by default in new installations of Oracle HTTP Server so that your system is secure out-of-the-box.

See Also: Apache Module mod_dav Security Issues in the Apache Server documentation.

Be sure to apply the standard Basic or Digest authentication and authorization mechanisms supported by Oracle HTTP Server. Generally, you do this for the default location, such as dav_public, in the supplied `mod_oradav.conf` file. This restricts who can use your system for remote storage, preventing unauthorized users from filling up your disks.

In addition, you should always apply Oracle HTTP Server authentication and authorization to authors of the Web site. You should also provide both an execution context and an editing context, so that Web authors, after being properly authenticated and authorized, can edit a JSP file or other executable file and then see how it runs. To do this, create an alias for the directory associated with the execution context, and then DAV-enable the aliased location.

9.4 OraDAV Performance Considerations

This section provides information that can help you optimize the performance of various operations. It contains the following subsections:

- Section 9.4.1, "Using Disk Caching with OraDAV"
- Section 9.4.2, "Bypassing Oracle Web Cache for WebDAV Activities"

9.4.1 Using Disk Caching with OraDAV

The performance benefit from disk caching is greatest with medium to large-size files (approximately 50 KB and larger). With smaller files, the performance benefit is less, and with very small files the performance can be worse with disk caching than without disk caching. For example, if the file myfile.dat is requested and if the file size is only 24 bytes, the time required for copying the file from the server to the local system is very small compared to the time required for accessing the server to check if the file has changed. If disk caching is not used, there is no check of the server to see if the file has changed, and the file is copied in all cases.

You can set the following OraDAV parameters to control disk caching for OraDAV operations:

- `ORACacheDirectory`
- `ORACacheTotalSize`
- `ORACacheMaxResourceSize`
- `ORACachePrunePercent`

If you specify `ORACacheDirectory`, disk caching for OraDAV operations is enabled. You must also specify a value for `ORACacheTotalSize`, and you can specify values for `ORACacheMaxResourceSize` and `ORACachePrunePercent` parameters. If you do not specify `ORACacheDirectory`, disk caching for OraDAV operations is not enabled, and other disk cache-related parameters are not relevant.

9.4.2 Bypassing Oracle Web Cache for WebDAV Activities

Oracle Web Cache enhances performance for most Web activity that involves client read-only operations of data on the Web server system. Oracle Web Cache does not cache OraDAV operations for GET, PUT, LOCK and UNLOCK, which are designed for read/write capability. For better performance, WebDAV clients can connect directly to Oracle HTTP Server.

To bypass Oracle Web Cache for WebDAV clients, you can send requests directly to the Oracle HTTP Server listen port, which is set in the `httpd.conf` file. By doing this, WebDAV clients will connect directly to Oracle HTTP Server, resulting in better performance than if Oracle Web Cache is used.

See Also: *Oracle Fusion Middleware Administrator's Guide for Oracle Web Cache*

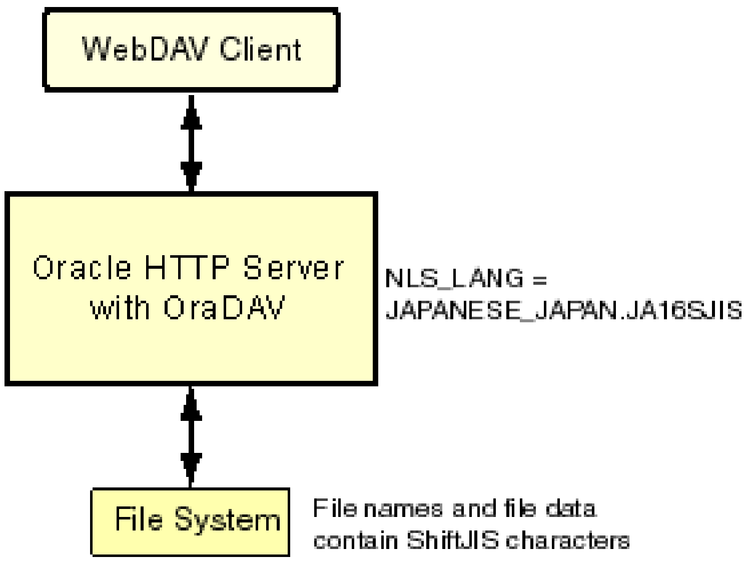
9.5 Globalization Support Considerations with OraDAV

The `DAVOraUseNLSLang` directive provides globalization support for access to the local file systems. This directive specifies whether or not the file names in the file system need to go through conversion using the NLS_LANG setting. A value of `Off` specifies that no conversion is needed. A value of `On` specifies that the character set for the file system provides for conversion of all possible characters in client requests. The default is `Off`.

For access to the local file system, the character set for the file system must be the same as, or compatible with, the character set for URLs embedded in client requests. The character set for the file system must provide for conversion of all possible characters in client requests. The NLS_LANG parameter value must represent the character set of both the client and the OraDAV server. You must also specify a value of `On` for the parameter `DAVOraUseNLSLang`.

For example, assume that you are using Web folders on a system where the files have ShiftJIS characters and that the file system under dav_public is represented by the operating system in the JAPANESE_JAPAN.JA16SJIS character sets shown in Figure 9–2.

Figure 9–2 OraDAV Access to File System with ShiftJIS Characters


WebDAV Client
Oracle HTTP Server with OraDAV
NLS_LANG = JAPANESE_JAPAN.JA16SJIS
File System
File names and file data contain ShiftJIS characters

You must do the following:

1. Set the NLS_LANG value to JAPANESE_JAPAN.JA16SJIS.
2. Include the following in the `mod_oradav.conf` file:

```
<Location /dav_public>
  DAV On
  DAVOraUseNLSLang On
</Location>
```

Note: If you use Microsoft Internet Explorer with OraDAV and a multibyte character set, you must disable the Internet option Always send URLs as UTF-8, located under the Advanced tab in the Internet Options section. By default, this option is enabled. The requirement to disable this option applies to both database access and file system access.

9.6 Location of DAV Files

When the `ORACLE_HOME/ohs/cas/templates/default/moduleconf/mod_oradav.conf` file is configured to use file storage, it places the files by default in:

```
ORACLE_INSTANCE/config/OHS/<ohs_name>/moduleconf
```

Oracle Fusion Middleware Backup and Recovery Service backs up this default location. If you change the location where the files are stored, and you want Oracle Fusion Middleware Backup and Recovery Service to backup the files, then you must register the new location.

Part III

Appendixes and Glossary

This part contains the following appendixes and a glossary:

A

Using Oracle Plug-ins for Third-Party Web Servers

This appendix explains how the Oracle Proxy Plug-In and Oracle SSO Plug-In enable you to use Oracle Fusion Middleware components in conjunction with a third-party HTTP listener. This appendix includes the following sections:

- Section A.1, "Using Oracle Proxy Plug-In"
- Section A.2, "Using Oracle SSO Plug-In"

Documentation from the Apache Software Foundation is referenced when applicable.

A.1 Using Oracle Proxy Plug-In

Oracle Proxy Plug-In enables you to proxy/send requests from a third-party HTTP listener to Oracle Fusion Middleware. The Oracle Proxy Plug-In is provided and certified to work with Sun Java System Web Server Enterprise Edition on UNIX and Microsoft Windows systems, or Microsoft Internet Information Server (IIS) v6.0 and 7.0.

For other third-party HTTP listeners, you can use the respective listener's native proxy functionality.

> **See Also:** `http://www.oracle.com/technology/software/products/ias/files/fusion_certification.html` for complete certification information.

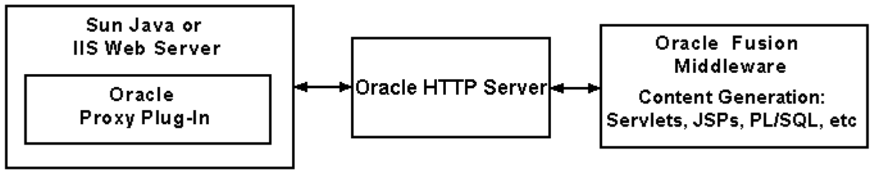
This section includes the following topics:

- Section A.1.1, "Overview of Oracle Proxy Plug-In"
- Section A.1.2, "Installing Oracle Proxy Plug-In"
- Section A.1.3, "Configuring Oracle Proxy Plug-In"
- Section A.1.4, "Configuring Sun Java System Listener to Use Oracle Proxy Plug-In"
- Section A.1.5, "Configuring Microsoft IIS 6.0 Listener to Use Oracle Proxy Plug-In"
- Section A.1.6, "Configuring Microsoft IIS 7.0 Listener to Use Oracle Proxy Plug-In"
- Section A.1.7, "Oracle Proxy Plug-In Usage Information"
- Section A.1.8, "Troubleshooting Oracle Proxy Plug-In"

A.1.1 Overview of Oracle Proxy Plug-In

Oracle Proxy Plug-In is a reverse HTTP proxy. The **plug-in** forwards incoming HTTP requests from a Sun Java System or Microsoft IIS server HTTP listener to an Oracle Fusion Middleware instance, as shown in Figure A–1.

Figure A–1 Oracle Proxy Plug-In


Sun Java or IIS Web Server
Oracle Proxy Plug-In
Oracle HTTP Server
Oracle Fusion Middleware
Content Generation: Servlets, JSPs, PL/SQL, etc

The proxy logic is provided as a plug-in, shared library that is loaded by the third-party HTTP listeners. The plug-in uses APIs provided with the third-party listeners to directly handle HTTP requests, in much the same way that modules are used with Oracle HTTP Server.

Oracle HTTP Server can mimic the address and port used by the third-party listener. When sending a request to Oracle HTTP Server, the proxy can be configured to send a different Host: HTTP header than the actual host name and port that the request is being sent to, so that other applications are safeguarded from the reverse proxy.

Oracle Proxy Plug-In does not perform load balancing; it only forwards requests to a single back-end Oracle HTTP Server for a given URL. Also, Oracle Proxy Plug-In does not support SSL between a third-party listener and Oracle HTTP Server.

A.1.2 Installing Oracle Proxy Plug-In

Oracle Proxy Plug-In for Sun Java System or Microsoft IIS is available on the Oracle Fusion Middleware WebTier and Utilities CD for Microsoft Windows, which is included in your Oracle Fusion Middleware CD Pack. Place the appropriate definition file and shared library in directories that are accessible to the third-party listener.

Files on the Oracle Fusion Middleware Web Tier CD-ROM are in the following locations, based on platform:

- UNIX: /plugins/solaris/ for UNIX
- Microsoft Windows: /plugins/win32/

Table A–1 contains information about the shared libraries for the OracleAS Proxy Plug-In.

Table A–1 Oracle Proxy Plug-In Shared Libraries

Listener	Location and Description
Sun Java System Web Listener	Location on CD-ROM: ▪ UNIX: `/Disk1/plugins/sjsws/oracle_proxy.so` ▪ Microsoft Windows: `\Disk1\plugins\sjsws\oracle_proxy.dll` To install the plug-in for the listener, copy the `.dll` file to a directory that the listener has read and execute privileges for UNIX, or a directory that the listener can access for Microsoft Windows.

Table A–1 (Cont.) Oracle Proxy Plug-In Shared Libraries

Listener	Location and Description
Microsoft IIS	Location on CD-ROM: ■ `\Disk1\plugins\iis\oracle_proxy.dll` To install the plug-in for the listener, copy `oracle_proxy.dll` to a directory the listener can access.

A.1.3 Configuring Oracle Proxy Plug-In

There is one definition file for the Oracle Proxy Plug-In that controls the proxy functionality. The presence of the file in the Web server file system makes the proxy functionality active. You also need to modify the configuration file(s) specific to the third-party listener, such as the Sun Java System Listener or Microsoft IIS configuration file, to enable the plug-in on these listeners. The definition file for the Oracle Proxy Plug-In can have any name.

A.1.3.1 Proxy Server Definition File

The **proxy server** definition file must reside in a directory that is readable by the third-party listener. For example, you could create a directory called `proxy` in a directory on your system, and place the proxy server definition file, the proxy shared library file, and proxy log files in it.

The proxy server definition file contains the following parameters:

- Name-value pairs that describe the servers that will be used to proxy requests to Oracle Fusion Middleware
- Options for communicating with the servers
- A set of rules that map URLs to the servers

You can create this file with the text editor of your choice.

A.1.3.2 Proxy Definition File Parameters

The following proxy parameters are used in the proxy definition file:

- oproxy.serverlist
- oproxy.servername.hostname
- oproxy.servername.port
- oproxy.servername.alias
- oproxy.servername.resolveall
- oproxy.servername.urlrule

A.1.3.2.1 oproxy.serverlist Lists all the server names recognizable to the plug-in.

Category	Value
Allowable Values	Comma-separated list of server names, one for each Oracle HTTP Server to which requests will be sent. All servers in the server list must also be defined in the file. **Note:** Oracle Proxy Plug-In does not do load balancing; therefore, if you define two servers with same `urlrule`, it will forward requests to only one of those servers.

Category	Value
Default Value	None. At least one server name must be provided for the proxy to be functional.
Example	`oproxy.serverlist=ohs1,ohs2`

A.1.3.2.2 oproxy.servername.hostname

Defines the host name to use when communicating with a specific server.

Category	Value
Allowable Values	A valid host name
Default Value	None
Example	`oproxy.ohs1.hostname=host1.acme.com`

A.1.3.2.3 oproxy.servername.port

Defines the port to use when communicating with a specific server.

Category	Value
Allowable Values	A port value
Default Value	80
Example	`oproxy.ohs1.port=7777`

A.1.3.2.4 oproxy.servername.alias

Supports the alias feature of the proxy by defining the host name and port that clients use to access the third-party HTTP listener. If defined, then this value will be passed as the Host: HTTP header. If not defined, then the host name and port of the system actually being communicated with will be sent.

Category	Value
Allowable Values	host:port
Example	`oproxy.ohs1.alias=www.example.com:80`

A.1.3.2.5 oproxy.servername.resolveall

Directs the proxy plug-in to resolve the host name to the backend server on every request. This enables DNS based failover or routing between the proxy plug-in and backend servers. The use of this parameter requires going to the DNS server for every incoming request, and should only be used if the mapping from host name to IP address will change dynamically.

Category	Value
Allowable Values	`true` | `false`
Default Value	`false`
Example	`oproxy.ohs1.resolveall=true`

A.1.3.2.6 oproxy.servername.urlrule

Describes a URL or set of URLs that are redirected to this server. A given server can have any number of urlrule properties assigned to it.

Category	Value
Example	`oproxy.ohs1.urlrule=/foo/*`

The following types of rules can be used:

- Exact matches: One URL is mapped to a server. For example:

  `oproxy.ohs1.urlrule=/foo/bar/foo.html`

  maps only the URL `/foo/bar/foo.html`. This would be the proxy for the server with the name `ohs1`.

- Context matches: A set of URLs with a common prefix or context are mapped to a server. For example:

  `oproxy.ohs1.urlrule=/foo/*`

  maps URLs beginning with `/foo` to the server with the name `ohs1`.

  For context matches, you can use the `stripcontext` option with the `urlrule` parameter to send only the portion of the URL following the wildcard to the server. The default for the stripcontext option is `false`, so you do not need to include it unless you are setting it to `true`. It is shown for completeness of the example.

Parameters	URL Request	Result

oproxy.ohs1.urlrule=/foo/*
oproxy.ohs1.stripcontext=true
	http://*hostname*/foo/header1.gif	*ORACLE_INSTANCE*/config/OHS/<*ohs_name*>/htdocs/header1.gif

oproxy.ohs1.urlrule=/foo/*
oproxy.ohs1.stripcontext=false
	http://*hostname*/foo/header1.gif	ORACLE_INSTANCE/config/OHS/<*ohs_name*>/htdocs/foo/header1.gif

- Suffix matches: All files with a common file extension are mapped to a server.

  For example:

  `oproxy.ohs1.urlrule=/*.jsp`

  maps all the URLs that end in `.jsp` to the server `ohs1`. This can be combined with the context rule to have `/foo/bar/*.jsp` so that only URLs that start with `/foo/bar` and end in `.jsp` would be proxied.

Note: For `oproxy.`*`servername`*`.urlrule`, when multiple rules apply to the same URL, the following precedence applies:

1. Exact matches
2. Longest context match plus suffix match
3. Longest context match

Some examples of the precedence are as follows:

- /foo/bar/index.html would take precedence over /foo/bar/*
- /foo/bar/*.jsp would take precedence over /foo/bar/*
- /foo/bar/* would take precedence over /foo/*

A.1.3.3 Defining Oracle Proxy Plug-In Behavior

In the proxy server definition file, you define which servers and URLs to use as proxy to the plug-in.

1. In the first line of the file, specify the list of all the servers that can be used by the plug-ins, such as the following:

   ```
   oproxy.serverlist=ohs1,ohs2
   ```

2. View the Oracle HTTP Server ports using Fusion Middleware Control or the opmnctl command-line utility. The port number will be used in the next step.
3. Set the relevant properties (host name, port, and server alias) for each server. For example:

   ```
   oproxy.ohs1.hostname=host1.acme.com.us.example.com
   oproxy.ohs1.port=7777
   oproxy.ohs1.alias=www.example.com
   ```

   The host name must be provided. If an alias value is not given, the combination of the host name and port is used. The alias enables the backend server to receive requests that have an HTTP Host header that looks exactly like the one the client delivers to the third-party listener.
4. Set the `urlrule` parameter to specify redirection between servers. For example, this rule maps all incoming requests to proxy to the Web server on the server `ohs1`:

   ```
   oproxy.ohs1.urlrule=/*
   ```

Example A–1 is a sample proxy server definition file.

Example A–1 Sample Proxy Server Definition File

```
# This file defines proxy server behavior.
#
# Server names that the proxy plug-in will recognize.
oproxy.serverlist=ohs1

# Hostname to use when communicating with a specific server.
oproxy.ohs1.hostname=host1.acme.com

# Port to use when communicating with a specific server.
oproxy.ohs1.port=7777

# URL(s) that will be redirected to this server.
oproxy.ohs1.urlrule=/*
```

A.1.4 Configuring Sun Java System Listener to Use Oracle Proxy Plug-In

This section provides proxy plug-in configuration information for Sun Java System listener on UNIX and Microsoft Windows platforms.

The default configuration files for Sun Java System route all incoming requests for the URI /servlet to the Sun Java System servlet handler. The Oracle Proxy Plug-In does not override the Sun Java System server configuration settings. You must ensure that the URL mappings to the Oracle Proxy Plug-In are distinct from the URL mappings to the Sun Java System servlet engine.

If you are configuring the Sun Java System listener on Microsoft Windows, use forward slashes (/) in all paths.

1. Open the `magnus.conf` file located in the `/config` directory for the Sun Java System listener version 6 or 7.
2. Add the following load-modules line, depending on the operating system:
   - For UNIX:

     ```
     Init fn="load-modules" shlib="/path/oracle_proxy.so" funcs=op_init,op_
     objecttype,op_service
     ```

   - For Microsoft Windows:

     ```
     Init fn="load-modules" shlib="/path/oracle_proxy.dll" funcs=op_init,op_
     objecttype,op_service
     ```

   In the preceding lines, *path* is the path to the shared library for the plug-in. This tells the listener where the proxy shared library is, and which functions are exposed by this library.
3. Add the following configuration parameters:

   ```
   Init fn="op_init" server_defs="/path/to/proxy/definition/file" log_
   file="/path/to/proxy/log/file" log_level=loglevel
   ```

   For example:

   ```
   Init fn="op_init" server_defs="/oracle/proxyplugin/proxydefs" log_
   file="/oracle/proxyplugin/oproxy.log" log_level=error
   ```

   The proxy server definition file contains all the configuration information for the servers that can communicate with the proxy plug-in. The definition file can have any name.

   A log file and log level can be specified to log messages from the plug-in. This is optional.

   > **See Also:** See Section A.1.3.1, "Proxy Server Definition File" for a complete description and example

4. Add the following line to the `<Object name=default>` section of the `obj.conf` file, before all other lines beginning with the word `ObjectType`:

   ```
   ObjectType fn=op_objecttype
   ```

5. Add the following line before all other lines that begin with the word `Service`:

   ```
   Service type="oracle/proxy" fn="op_service"
   ```

6. Save the file.
7. Start the listener using the Sun Java System GUI or the shell script.

A.1.5 Configuring Microsoft IIS 6.0 Listener to Use Oracle Proxy Plug-In

This section provides proxy plug-in configuration instructions for the Microsoft IIS 6.0 listener on Microsoft Windows platforms. The procedure involves creating Microsoft Windows registry entries and using the Microsoft IIS 6.0 management console to add directories and filters. You must restart the listener after configuring the plug-in.

To configure the plug-in, perform the following steps:

1. Use the Microsoft Windows Registry Editor to create new registry entries.

   a. From the Start menu, Select **Run**, and the type `regedit` in the dialog box, and click **OK**. The Registry Editor window opens

   b. Expand the HKEY_LOCAL_MACHINE folder by clicking the plus sign (+) preceding its name In the Registry Editor window.

   c. Expand the SOFTWARE folder by clicking the plus sign (+) preceding its name, and then Click the **ORACLE** folder.

   d. From the Edit menu, select **New** > **Key**. A new folder is added under the ORACLE folder with the name New Key #1.

   e. Enter `IIS Proxy Adapter` for the key name.

   f. From the Edit menu, select **New** > **String Value**. A new value is added in the right window pane with the name New Value #1. Enter `server_defs` for the value name.

   g. From the Edit menu, select **Modify**. The Edit String dialog box appears.

   h. In the Value data field, enter the full path of your proxy server definition file, and then click **OK**.

   i. Optionally, you can specify log_file and log_level using the procedure specified in steps **d.) – h.)**.

      – Add a string value with the name `log_file` and the desired location of the log file, such as `d:\proxy\proxy.log`.

      – Add a string value with the name `log_level` and a value for the desired log level. Valid values are `debug`, `inform`, `error`, and `emerg`.

   j. Close the Registry Editor window by selecting **Exit** from the File menu.

2. Use the Microsoft IIS management console to add a new virtual directory to your Microsoft IIS Web site with the same physical path as that of the `oracle_proxy.dll`.

   a. Open the IIS Manager by clicking **Start** -> **Programs** -> **Administrative Tools** -> **Internet Information Services (IIS) Manager**.

   b. Expand the server folder by clicking the plus sign (+) preceding the server name.

   c. Right-click the **Default Sites** folder, and then select the **New** -> **Virtual Directory** option from the menu.

   d. In the Virtual Directory Creation Wizard window, enter **oproxy** for the Alias. Then, enter the path or browse to the directory containing the `oracle_proxy.dll` file (e.g., `c:\OProxy`) and select the **Execute (such as ISAPI applications or CGI)** check box.

   e. Click **Finish** to close the Virtual Directory Creation Wizard.

3. Use the Microsoft IIS management console to add `oracle_proxy.dll` as a filter in your Microsoft IIS Web site. The name of the filter should be **oproxy** and its executable must point to the directory containing the `oracle_proxy.dll` file.

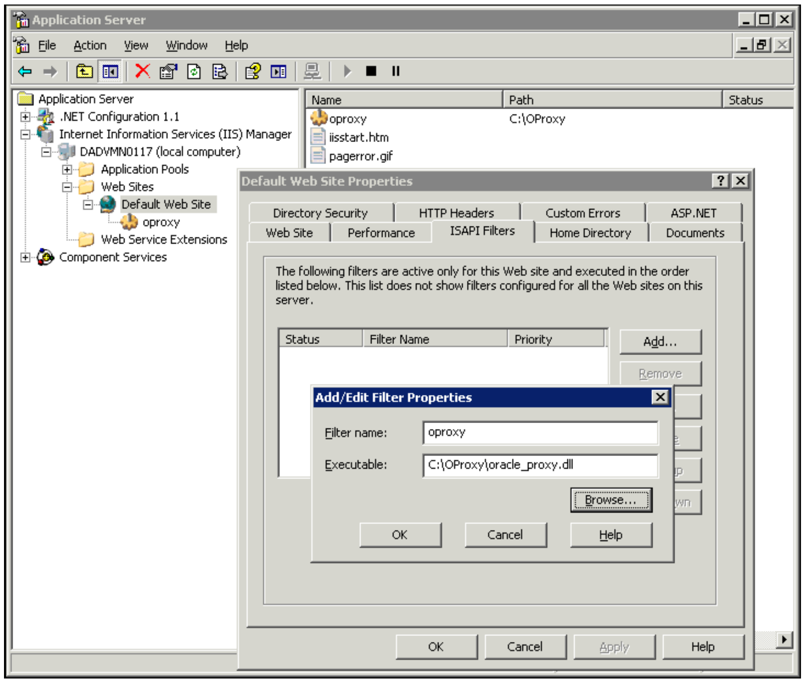
   a. Right-click the **Default Sites** folder, and then select the **Properties** option from the menu.

   b. In the Default Web Site Properties window, select the **ISAPI Filters** tab.

   c. Click **Add** to add a new filter.

The Add/Edit Filter Properties window is displayed.

d. Enter **oproxy** for the filter name. Then, enter the path or browse to the directory containing the `oracle_proxy.dll` file (e.g., `c:\OProxy\oracle_proxy.dll`).


Application Server
File Action View Window Help
Application Server
.NET Configuration 1.1
Internet Information Services (IIS) Manager
DADVMN0117 (local computer)
Application Pools
Web Sites
Default Web Site
oproxy
Web Service Extensions
Component Services
Name Path Status
oproxy C:\OProxy
iisstart.htm
pagerror.gif
Default Web Site Properties
Directory Security HTTP Headers Custom Errors ASP.NET
Web Site Performance ISAPI Filters Home Directory Documents
Status Filter Name Priority
Add... Remove
OK Cancel Apply Help
Add/Edit Filter Properties
Browse...
OK Cancel Help

The following filters are active only for this Web site and executed in the order listed below. This list does not show filters configured for all the Web sites on this server.

Filter name: oproxy

Executable: C:\OProxy\oracle_proxy.dll

e. Click **OK** to close the Add/Edit Filter Properties window.

f. Click **OK** to close the Default Web Site Properties window.

4. Under the server name folder, click the **Web Service Extensions** folder to open the Web Service Extensions page. Select **All Unknown ISAPI Extensions** in the right panel and click **Allow**. The status of this item should change from *Prohibited* to *Allowed*.

5. Select the **Application Pools** folder, and then right-click the **Properties** option from the menu. On the Application Pools Properties window, select the **Identity** tab and change the Predefined identity to **Local System**. A confirmation dialog box displays asking Are you sure you want to do this? Select **Yes**.

6. Stop and restart the Microsoft IIS 6.0 server.

Note: To restart Microsoft IIS, you must stop all the Microsoft IIS services through the control panel or restart the computer. This is the only way to ensure that the `.dll` is reloaded. Restarting Microsoft IIS 6.0 through the management console is not sufficient.

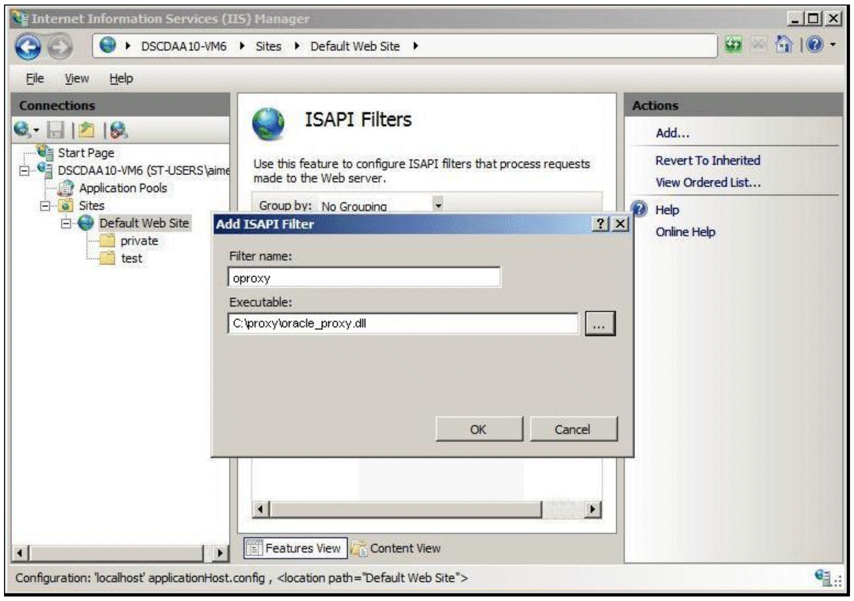
7. Make sure the newly created `oproxy` filter is marked with a green upward arrow.

A.1.6 Configuring Microsoft IIS 7.0 Listener to Use Oracle Proxy Plug-In

This section provides proxy plug-in configuration instructions for the Microsoft IIS 7.0 Listener on Microsoft Windows platforms. The procedure involves creating Microsoft Windows registry entries and using the Microsoft IIS 7.0 management console to add directories and filters. You must restart the listener after configuring the plug-in.

To configure the plug-in, perform the following steps:

1. Complete step 1 in Section A.1.5, "Configuring Microsoft IIS 6.0 Listener to Use Oracle Proxy Plug-In" to use the Registry Editor to create new registry entries for the plug-in.
2. Use the Microsoft IIS management console to add the proxy plug-in filter:
   a. Open the IIS Manager by clicking **Start** -> **Programs** -> **Administrative Tools** -> **Internet Information Services (IIS) Manager**.
   b. Expand the server folder by clicking the plus sign (+) preceding the server name (e.g, DSCDAA10-VM6).
   c. Expand the Sites folder by clicking the plus sign (+) preceding its name.
   d. Click the **Default Web Site** icon to open the Default Web Site Home page.
   e. Double-click the **ISAPI Filters** icon to open the ISAPI Filters page, and then complete the following tasks:
      – In the Actions pane, click **Add** to open the Add ISAPI Filter dialog box.
      – In the Filter Name field, enter a user-friendly name for the ISAPI filter.
      – In the Executable field, enter the file system path for the location of ISAPI filter file, or click the **ellipsis** button (...) to navigate to the folder that contains the ISAPI filter file.



Use this feature to configure ISAPI filters that process requests made to the Web server.

Filter name: oproxy

Executable: C:\proxy\oracle_proxy.dll

Configuration: 'localhost' applicationHost.config , <location path="Default Web Site">

      – Click **OK**.
3. Follow these steps to configure the newly added ISAPI filter:
   a. Click the **Default Web Site** icon in the navigation panel to view all the settings related to the application that can be modified.
   b. Click the **Handler Mappings** option to set the mappings for the handler for a particular MIME type.
   c. Click the **StaticFile** option in the Handler Mappings page, and in the Edit Module Mapping dialog box, change the Request path to `*.*`.

d. In the Actions area of the Handler Mappings page, click the **Add Script Map** option.

e. In the Edit Script Map dialog box: enter * for the Request path. Use the Executable field to browse to the `oracle_proxy.dll` file and add it as the executable. Name it as `proxy`.

f. Click the **Request Restrictions** button to open the Request Restrictions dialog box. Clear the **Invoke handler only if the request is mapped to** check box and then click **OK** to add this Handler mapping.

g. Click **Yes** on the Add Script Map dialog box.

When you click the Root node of the IIS manager tree, and then click on the ISAPI and CGI Restrictions, you should see an entry for the `oracle_proxy.dll`, as shown here:

```
[No Description] Allowed C:\proxy\oracle_proxy.dll
```

4. Restart the Microsoft IIS server by opening the Services Control Panel, and then stopping and restarting the World Wide Web Publishing Service.

Note: To restart Microsoft IIS, you must stop all the Microsoft IIS services through the control panel or restart the computer. This is the only way to ensure that the `.dll` is reloaded. Restarting Microsoft IIS 7.0 through the management console is not sufficient.

A.1.7 Oracle Proxy Plug-In Usage Information

This section highlights development and usage practices to consider when developing an application that runs behind the Oracle Proxy Plug-In. Some of the practices also apply when enabling an application to run behind Oracle Web Cache.

- Check for configurations based on Oracle HTTP Server being the entry point into the network.

  This is usually only relevant if an application has a module that plugs directly into Oracle HTTP Server. Specifically, look for dependencies on obtaining information about the client based on the connection made to Oracle HTTP Server, such as using the SSL certificate for authentication. Currently, Secure Socket Layer (SSL) is not supported, so even if the client uses SSL to connect to the third-party listener, an unencrypted HTTP message will be sent from the third-party listener to Oracle HTTP Server. This means that client certificates will not be available to components that reside behind the plug-in. The environment variable REMOTE_ADDR has been specifically preserved when Oracle Proxy Plug-In and Oracle Web Cache are used, but other client information may, in practice, represent the system on which the proxy resides rather than the actual client host. These behaviors must be discovered and eliminated in cases where Oracle HTTP Server is not the external listener for Oracle Fusion Middleware.

- Avoid embedding host names into HTML unless the link is external to the Web site.

  This includes static HTML pages, dynamic pages generated by servlets, JSPs, PL/SQL, and so on. Examine all code that obtains the server name of Oracle HTTP Server to ensure that the code is not embedding the server name into pages that are sent back to the client. To test for this behavior, use a Web crawler application (also known as a spider) to traverse all links in a Web site. Open source tools with this functionality are available.

- Avoid returning host and port information in applications (such as applets or javascript) downloaded to the client.

  If you have an application that uses browser-based code, ensure that the code does not contain the host name and port of Oracle HTTP Server that actually delivers the content. Instead, it must have the actual client-accessible address used by the third-party listener.

- Ensure that all URLs within an application can be easily mapped to a set of rules that the proxy can use.

  To successfully proxy all requests for an application, the Oracle Proxy Plug-In must have a complete description of the URL space for that application. Each Oracle Fusion Middleware application must describe the set of rules necessary to configure the plug-in for that application. This set of rules must include all URLs that the application could generate. If an application generates a URL that is not described by the proxy `urlrule` parameters, then the request will be served by the third-party HTTP listener, and a "document not found" error may occur, or a different document may be delivered to the client.

  Developers of applications that use common top-level directories, such as a reliance on mapping /images, should be prepared to the following:

  - Change these common links to something that will not conflict with applications that might already be deployed on the third-party listener.
  - Instruct the user to copy the necessary content to the third-party listener directory structure. For performance reasons, it is a good idea to have the third-party listener handle static `.gif` and `.jpg` files, but it requires that the files be copied to the third-party listener.

A.1.8 Troubleshooting Oracle Proxy Plug-In

This section describes common problems and solutions related to Oracle Proxy Plug-In.

Listener Fails to Start

- Check for problems in the proxy server definition file. Each server in the server list line must be defined later in the file, and you must define at least one server. If a server name is listed but not defined, then the listener may not start. Ensure that there are no typographical errors or missing quotes in the proxy server definition file.
- For Sun Java System 6.0: Ensure that Init lines are added to the `magnus.conf` file, and the `ObjectType` and `Service` lines are added to the `obj.conf` file.

Listener Returns Incorrect URLs

- Verify that changes to the proxy server definition file were saved and the listener was restarted.
- Ensure that there are no typographical errors in the proxy server definition file.
- Ensure that the `urlrule` parameter is set up correctly, and determine whether the `stripcontext` option should be set to `true`.
- Verify that the `serverlist` line in the proxy server definition file specifies the back-end server you are trying to reach.
- Verify that the back-end server is running, and that the file you are attempting to retrieve exists and is accessible on the back-end server.

- Verify that the host, port and urlrule parameters in the proxy server definition file target the correct area on the back-end server.
- Ensure that client requests are being sent to the correct port on the third-party listener machine.
- Check the listener log files, the proxy log, and the back-end server logs to verify that requests are getting through. The proxy log may need to be set to debug mode You may need to restart the listener.

Changes Made to Proxy Server Definition File are Not Reflected

- Verify that changes to the proxy server definition file were saved and the listener was restarted.
- For Microsoft IIS, verify that WWW Publishing Service was stopped and started from the Control Panel. This may take a few minutes.

Microsoft IIS Listener Displays Incomplete Pages or Garbled Characters

Do not display Microsoft IIS pages with a Sun Java System browser.

Parsing Error Occurs with Sun Java System 6.0 or 7.0

If you try to change the ports or turn on security (for SSL), the server may return the error message "Unable to parse magnus.conf."

Remove any comments and added lines preceding and following the Init lines in the magnus.conf file.

File Not Found Error Occurs

If you are using a context-based `urlrule` parameter to retrieve a file that is known to exist, and the listener returns "Not Found," you probably need to set `stripcontext=true`.

See Also: Section A.1.3.2.6, "oproxy.servername.urlrule"

Sun Java System Web Server Returns Server Error with /servlet Request

The default Sun Java System configuration maps any URL requests to `/servlet` to its own servlet handler. You must edit the proxy server definition file, or change the Sun Java System configuration to correct this.

Partial URL Requests Return Unexpected Results

The Microsoft IIS and Sun Java System servers auto-complete URLs differently than others. Requests to `http://example`, `http://example/`, and `http://example/index.html` do not necessarily return the same results on different platforms. Use the `oproxy.servername.urlrule` parameter to work around this problem.

See Also: Section A.1.3.2.6, "oproxy.servername.urlrule"

Server Returns Page with Broken Image Links

If you use an exact `urlrule` parameter in the proxy server definition file, such as `urlrule=/*.html`, the server retrieves the specified page. All other links are forbidden to the user, including inline images on the page. If you use an exact `urlrule` with `stripcontext=true`, a server error is returned.

Unexpected Pages are Displayed

Clear the memory cache in your client browser. Earlier versions of Sun Java System and Microsoft Internet Explorer use cached pages even when set to retrieve the page every time and when no memory is allocated for caching. You may need to restart the browser to get this behavior to work. If you see a page you are not expecting, try refreshing the browser or reloading the page.

REMOTE_ADDR Contains Unexpected IP Address

The REMOTE_ADDR field usually contains the IP address of the client system. In some URL request cases, if there is a proxy server in the environment, the field may contain the IP address of the proxy server.

Redirects Go To Network Entry Point

If the back-end server returns a redirect to the entry point of the network, the host and port information should be updated. Choose one of the following options, based on your installation. The first option is the preferred method.

- Use the Advanced Configuration page of Fusion Middleware Control to set the following directives in the `httpd.conf` file:

  ```
  UseCanonicalName On
  ServerName name of listener host
  Port port of listener host
  ```

- Use the Advanced Configuration page of Fusion Middleware Control to set the following directives in the `httpd.conf` file:

  ```
  UseCanonicalName port
  Port port of listener host
  ```

Edit the proxy plug-in server definition file to use the following:

```
oproxy.serverName.alias=name of listener host:port of listener host
```

A.2 Using Oracle SSO Plug-In

Oracle SSO Plug-In is designed to protect native third-party listener applications using the Oracle single sign-on (SSO) infrastructure. The Oracle SSO Plug-In is provided and certified to work with Sun Java System Web Server Enterprise Edition on UNIX and Microsoft Windows systems, and Microsoft Internet Information Server (IIS) v6.0 and v7.0 on Microsoft Windows systems.

See Also:
`http://www.oracle.com/technology/software/products/ias/files/fusion_certification.html` for complete certification information

This chapter includes the following topics:

- Section A.2.1, "Overview of Oracle SSO Plug-In."
- Section A.2.2, "Installing Oracle SSO Plug-In."
- Section A.2.3, "Registering with the Oracle Single Sign-On Server."
- Section A.2.4, "Configuring the Oracle SSO Plug-In."
- Section A.2.4.2, "Rules to Protect Resources."

- Section A.2.5, "Configuring Sun Java System Listener for Single Sign-on."
- Section A.2.6, "Configuring Microsoft IIS 6.0 Listener to Use Oracle Single Sign-On."
- Section A.2.7, "Configuring Microsoft IIS 7.0 Listener to Use Oracle Single Sign-On."
- Section A.2.8, "Troubleshooting Oracle SSO Plug-In."

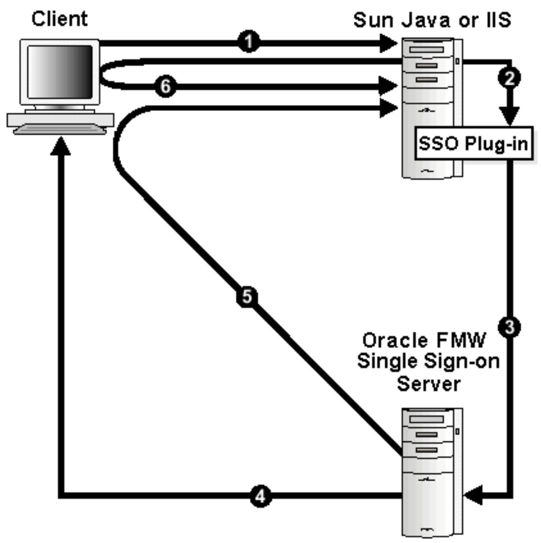
A.2.1 Overview of Oracle SSO Plug-In

Oracle SSO Plug-In is the Oracle single sign-on (SSO) solution for third-party listeners, such as Sun Java System and Microsoft IIS. The **plug-in** is designed to protect native third-party listener applications using the SSO infrastructure. With the help of the Oracle SSO Plug-In, users can be authenticated to different third-party listener applications using only one SSO password. You can integrate these SSO-protected third-party listener applications with SSO-enabled Oracle HTTP Server applications or legacy Oracle SSO enabled applications as long as they are all protected on the same SSO server.

Oracle SSO Plug-In is a simple version of mod_osso, and only implements some of its basic functionality. Features such as dynamic authentication, global logout, idle timeout, global timeout, and basic authentication for legacy application are not implemented in the current Oracle SSO Plug-In release.

Figure A–2 illustrates the process when a user requests a URL protected by the Oracle SSO Plug-In.

Figure A–2 Oracle SSO Plug-In



1. The user requests a URL through a Web browser.
2. The Web server looks for an Oracle SSO Plug-In cookie for the user. If the cookie exists, the Web server extracts the user's information and uses it to log the user in to the requested application.
3. If the cookie does not exist, the Oracle SSO Plug-In redirects the user to the single sign-on server.
4. The single sign-on server looks for its own cookie in the browser. If a cookie exists, then the single sign-on server authenticates using the cookie. If authentication is

successful, then the single sign-on server creates a cookie in the browser as a reminder that the user has been authenticated. If it finds none, it tries to authenticate the user with a user name and password.

5. The single sign-on server returns the user's encrypted information to the Oracle SSO Plug-In.
6. Oracle SSO Plug-In creates its own cookie for the user in the browser and redirects the user to the requested URL.

   During the same session, if the user again seeks access to the same or to a different application, the user is not prompted for a user name and password. The application uses an HTTP header to obtain this information from the Oracle SSO Plug-In session cookie.

See Also: *Oracle Fusion Middleware Security Guide*

A.2.2 Installing Oracle SSO Plug-In

Oracle SSO Plug-In for Sun Java System and Microsoft IIS is available on the Oracle Fusion Middleware Web Tier and Utilities CD from Microsoft Windows, which is included in your Oracle Fusion Middleware CD Pack.

Install Oracle SSO Plug-In on a system that has an Oracle Fusion Middleware installation. This installation is required only for the network and security dependent libraries and the single sign-on registration tool. It is not required to be running.

Place the configuration file and shared library in directories that are accessible to the third-party listener. For security reasons, ensure that all the configuration files and plug-in libraries are given minimum privileges.

On the Oracle Fusion Middleware Web Tier and Utilities CD, the files are located at `/plugins/solaris/` for UNIX and `/plugins/win32/` for Microsoft Windows.

Table A–2 contains information about the shared libraries for Oracle SSO Plug-In.

Table A–2 Oracle SSO Plug-In Shared Libraries

Listener	Location and Description
Sun Java System Web Listener	Location on CD-ROM: ■ UNIX: `/Disk1/plugins/sjsws/oracle_osso.so` ■ Microsoft Windows: `\Disk1\plugins\sjsws\oracle_osso.dll` To install the plug-in for the listener, copy the `.dll` file to a directory that the listener has read and execute privileges for UNIX, or a directory that the listener can access for Microsoft Windows.
Microsoft IIS	Location: ■ `\Disk1\plugins\iis\oracle_osso.dll` To install the plug-in for the listener, copy `oracle_osso.dll` to a directory the listener can access.

A.2.3 Registering with the Oracle Single Sign-On Server

The **single sign-on** registration process enables the single sign-on server and the listener to share information such as server location, protocol version, and common encryption key, before they communicate. After the registration process, this information is stored on the single sign-on server side as a single sign-on partner application entry. On the listener side, a single sign-on file called `osso_conf` is created. The `osso_conf` file is obfuscated for security purposes. Copy the file to an appropriate location so the listener can access it.

> **See Also:** *Oracle Fusion Middleware Security Guide* for details on how to register with Oracle Single Sign-On.

A.2.4 Configuring the Oracle SSO Plug-In

To configure Oracle SSO Plug-In, you must create a configuration file such as the `osso_plugin.conf` file. This file must reside in a directory that is readable by the third-party listener. You define all the plug-in functionality within the file. It can also be referred as the `osso` property file. The file contains the following:

- Plug-in directives such as `LoginServerFile` and `IpCheck`
- A set of rules that match resources to be protected.

A.2.4.1 Oracle SSO Plug-In Directives

Table A–3 lists the configuration directives for the Oracle SSO Plug-In.

Table A–3 Oracle SSO Plug-In Configuration Directives

Directive	Function
`LoginServerFile`	Specifies the location of the single sign-on server configuration file such as the `osso.conf` file that is attained from the SSO registration process. This is a global parameter and should not be used on a per-resource basis. You must provide one and only one single sign-on server configuration file. ■ Value: The full path of your Single Sign-On Server configuration file ■ Default: None ■ Example: `LoginServerFile=c:\OSSO\config\osso.conf`
`IpCheck`	Specifies whether the Oracle SSO plug-in should check the IP address of each request when it examines the cookie. Setting it to `true` prevents cookies from being accessed by another person. ■ Values: `true	false` ■ Default: `false` ■ Example: `IpCheck=true`  **Note:** Set it to `false` if you have a proxy server or firewall between your Sun Java System server and your client browsers.

A.2.4.2 Rules to Protect Resources

To ensure resource protection via the Oracle SSO Plug-In, a set of rules must be defined. The rules are defined according to the following format:

```
<OSSO url-matching-rule>
  SSO_configuration_directives
</OSSO>
```

Use the following rules to define the url-matching-rule:

Rule Name	Description
Exact Match	This option identifies an exact file as a protected resource, for example: /examples/hello.html

Rule Name	Description
Context Match	This option identifies a directory as a protected resource, for example: /examples/*
Extension Match	This option identifies files with a certain extension in a particular directory as a protected resource, for example: /examples/*.jsp

When multiple rules apply to the same URL, the following precedence applies:

1. Exact matches
2. Longest context match plus suffix match
3. Longest context match

Some examples of the precedence are:

- /foo/bar/index.html would take precedence over /foo/bar/*
- /foo/bar/*.jsp would take precedence over /foo/bar/*
- /foo/bar/* would take precedence over /foo/*

Example A–2 shows a simple file with the commands for resource protection. In the example, the `IpCheck` directive is set to `false` for the `/private/hello.html` file, but it is set to `true` for `/private2/*.jsp`. This setting ensures the cookies used with requests to the `/private2/*.jsp` files are not accessed by another user.

Example A–2 Simple Single Sign-on Configuration File, osso_plugin.conf

```
LoginServerFile=c:\OSSO\conf\osso.conf
<OSSO /private/hello.html>
  IpCheck=false
</OSSO>
<OSSO /private1/*>
</OSSO>
<OSSO /private2/*.jsp>
  IpCheck=true
</OSSO>
```

A.2.5 Configuring Sun Java System Listener for Single Sign-on

This section provides Oracle SSO Plug-In configuration information for the Sun Java System listener. You can use any text editor to edit the files.

> **Note:** If you are configuring the Sun Java System listener on Microsoft Windows, use forward slashes (/) in all paths.

1. Open the `magnus.conf` configuration file located in the `/config` directory for the Sun Java System version 6 or 7 listener.
2. Add the following lines to the file. The two lines beginning with `Init` must be added at the end of the file.

   On UNIX:

```
Init fn="load-modules" shlib="/path1/oracle_osso.so" funcs="osso_init, \
oracle_single_sign_on,osso_redirect_service,osso_success_service"

Init fn="osso_init" osso_properties="/path2/osso_plugin.conf"   \
log_file="/path2/plugin.log" log_level=error
```

On Microsoft Windows:

```
Init fn="load-modules" shlib="/path1/oracle_osso.so" \
funcs="osso_init,oracle_single_sign_on,osso_redirect_service, \
osso_success_service"

Init fn="osso_init" osso_properties="/path2/osso_plugin.conf"   \
log_file="/path2/plugin.log" log_level=error
```

In the preceding lines, the following variables were used:

- *path1* is the path to the shared library for the plug-in. This line tells the listener where the proxy shared library is, and which functions are exposed by this library.
- *path2* is the path to the plug-in configuration file you just created. This line can specify a log file and log level to log messages from the plug-in (optional).

3. Add the following line to the `<Object name=default>` section of the file, before all other lines:

   ```
   AuthTrans fn="oracle_single_sign_on"
   ```

4. Add the following line to the `<Object name=default>` section before all other lines that begin with the word `Service`:

   ```
   Service type="oracle/sso_redirect" fn="osso_redirect_service"
   ```

5. Add the following lines:

   ```
   <Object ppath="/path/osso_login_success">
   Service fn="osso_success_service"
   </Object>
   ```

   path is the path of your document root, for example, `/home/Sun Java/docs/` or `$docroot`.

6. Change the library path variable in the start script to include the location of the `ORACLE_HOME/lib32` directory.
7. Restart the listener.

A.2.6 Configuring Microsoft IIS 6.0 Listener to Use Oracle Single Sign-On

This section provides instructions on configuring the Microsoft IIS 6.0 Listener to use Oracle SSO Plug-In. The plug-in consists of a single `.dll` file, `oracle_osso.dll`. To install the plug-in, copy the `.dll` to the host on which Microsoft IIS 6.0 resides and perform the following steps:

1. Use the Microsoft Windows Registry Editor to create new registry entries.
   a. From the Start menu, Select **Run**, and the type `regedit` in the dialog box, and click **OK**. The Registry Editor window opens
   b. Expand the HKEY_LOCAL_MACHINE folder by clicking the plus sign (+) preceding its name In the Registry Editor window.
   c. Expand the SOFTWARE folder by clicking the plus sign (+) preceding its name, and then Click the **ORACLE** folder.
   d. From the Edit menu, select **New** > **Key**. A new folder is added under the ORACLE folder with the name New Key #1.

e. Enter `IIS OSSO Adapter` for the key name.

f. From the Edit menu, select **New** > **String Value**. A new value is added in the right window pane with the name New Value #1. Enter `cfg_file` for the value name.

g. From the Edit menu, select **Modify**. The Edit String dialog box appears.

h. In the Value data field, enter the full path of the OSSO plug-in configuration file you created (e.g., `c:\osso\osso_plugin.conf`).

Note: This is the plug-in configuration file and not the encrypted `osso.conf file` generated by the SSO registration process.

i. Optionally, you can specify log_file and log_level using the procedure specified in steps **d.)** through **h.)**.

   – Add a string value with the name `log_file` and the desired location of the log file, such as `c:\osss\osso_plugin.log`.

   – Add a string value with the name `log_level` and a value for the desired log level. Valid values are `debug`, `inform`, `error`, and `emerg`.

j. Close the Registry Editor window by selecting **Exit** from the File menu.

2. Use the Microsoft IIS management console to add a new virtual directory to your Microsoft IIS Web site with the same physical path as that of the `osso.dll`.

   a. Open the IIS Manager by clicking **Start** -> **Programs** -> **Administrative Tools** -> **Internet Information Services (IIS) Manager**.

   b. Expand the server folder by clicking the plus sign (+) preceding the server name.

   c. Right-click the **Default Sites** folder, and then select the **New** -> **Virtual Directory** option from the menu.

   d. In the Virtual Directory Creation Wizard window, enter **osso** for the Alias. Then, enter the path or browse to the directory containing the `oracle_osso.dll` file (e.g., `c:\osso`) and select the **Execute (such as ISAPI applications or CGI)** check box.

   e. Click **Finish** to close the Virtual Directory Creation Wizard.

3. Use the Microsoft IIS management console to add `oracle_osso.dll` as a filter in your Microsoft IIS Web site. The name of the filter should be **osso** and its executable must point to the directory containing the `oracle_osso.dll` file.

   a. Right-click the **Default Sites** folder, and then select the **Properties** option from the menu.

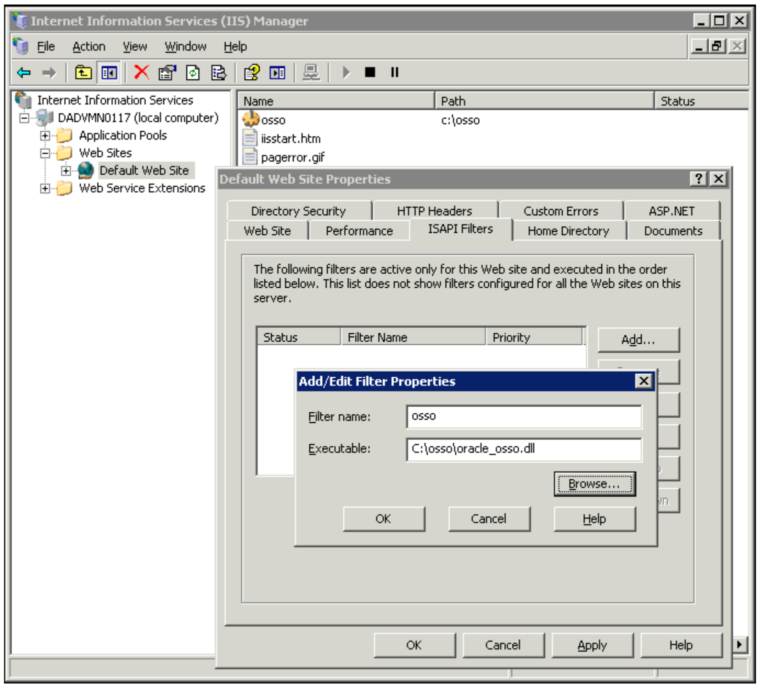
   b. In the Default Web Site Properties window, select the **ISAPI Filters** tab.

   c. Click **Add** to add a new filter.

      The Add/Edit Filter Properties window is displayed.

   d. In the Filter Name field, enter **osso** for the filter name.

   e. In the Executable field, enter the path for the location containing the `oracle_osso.dll` (e.g., `c:\osso\oracle_osso.dll`), or click the **ellipsis** button (...) to navigate to the folder that contains the `oracle_osso.dll` file.



The following filters are active only for this Web site and executed in the order listed below. This list does not show filters configured for all the Web sites on this server.

Filter name: osso

Executable: C:\osso\oracle_osso.dll

f. Click **OK** to close the Add/Edit Filter Properties window.

g. Click **OK** to close the Default Web Site Properties window.

4. Configure security settings for Oracle Home directory. Make sure you login to machine as an administrator user.

   a. In Windows Explorer, right-click the `ORACLE_HOME\bin` folder, select **Properties** from the menu, and then click the **Security** tab.

   b. Add the IIS_WPG, NETWORK and NETWORK SERVICE groups with Read and Execute permissions.

   c. Click **OK**.

5. Stop and restart the Microsoft IIS 6.0 Server.

Notes:

- To restart Microsoft IIS 6.0, you must stop all the Microsoft IIS 6.0 services through the control panel or restart the computer. This is the only way to ensure that the `.dll` file is reloaded. Restarting Microsoft IIS 6.0 through the management console is not sufficient.
- If you want multiple Oracle installations on the same home, then the `ORACLE_HOME\bin` PATH entry for the installation that you plan to use in conjunction with the Oracle SSO Plug-In must appear first in your PATH.
- Make sure the newly added ISAPI filter is marked with a green upward arrow.

A.2.7 Configuring Microsoft IIS 7.0 Listener to Use Oracle Single Sign-On

This section provides instructions on configuring the Microsoft IIS 7.0 Listener to use Oracle SSO Plug-In. The plug-in consists of a single `.dll` file, `oracle_osso.dll`. To install the plug-in, copy the `.dll` to the host on which Microsoft IIS 7.0 resides and perform the following steps:

1. Complete step 1 in Section A.2.6, "Configuring Microsoft IIS 6.0 Listener to Use Oracle Single Sign-On" to use the Microsoft Windows Registry Editor to create new registry entries for the plug-in.

2. Use the Microsoft IIS management console to add the `oracle_osso.dll` as a filter in your Microsoft IIS Web site:

   a. Open the IIS Manager by clicking **Start** -> **Programs** -> **Administrative Tools** -> **Internet Information Services (IIS) Manager**.

   b. Expand the server folder by clicking the plus sign (+) preceding the server name (e.g, DSCDAA10-VM6).

   c. Expand the Sites folder by clicking the plus sign (+) preceding its name.

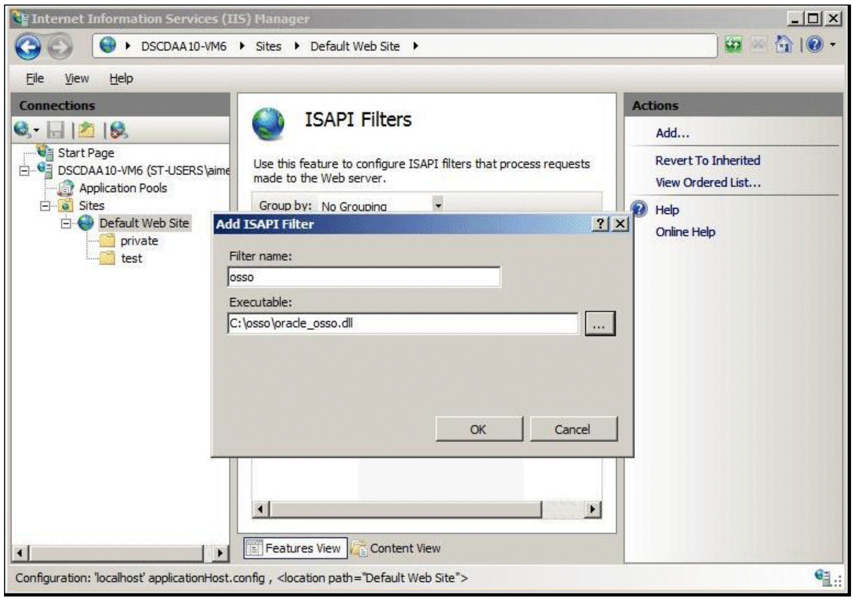
   d. Click the **Default Web Site** icon to open the Default Web Site Home page.

   e. Double-click the **ISAPI Filters** icon to open the ISAPI Filters page, and then complete the following tasks:

      – In the Actions pane, click **Add** to open the Add ISAPI Filter dialog box.

      – In the Filter Name field, enter osso.

      – In the Executable field, enter the file system path for the location containing the `oracle_osso.dll` (e.g., `c:\osso\oracle_osso.dll`), or click the **ellipsis** button (...) to navigate to the folder that contains the `oracle_osso.dll` file.



Use this feature to configure ISAPI filters that process requests made to the Web server.

Filter name: osso

Executable: C:\osso\oracle_osso.dll

Configuration: 'localhost' applicationHost.config , <location path="Default Web Site">

      – Click **OK**.

3. Configure security settings for Oracle Home directory. Make sure you log in to machine as an administrator user.

a. In Windows Explorer, right-click the `ORACLE_HOME\bin` folder, select **Properties** from the menu, and then click the **Security** tab.

b. Add the IIS_WPG, NETWORK and NETWORK SERVICE groups with Read and Execute permissions.

c. Click **OK**.

4. Restart the Microsoft IIS server by opening the Services Control Panel, and then stopping and restarting the World Wide Web Publishing Service.

Notes:

- To restart Microsoft IIS 7.0, you must stop all the Microsoft IIS 7.0 services through the Services Control Panel or restart the computer. This is the only way to ensure that the `.dll` file is reloaded. Restarting Microsoft IIS 7.0 through the management console is not sufficient.
- If you want multiple Oracle installations on the same home, then the `ORACLE_HOME\bin` PATH entry for the installation that you plan to use in conjunction with the Oracle SSO Plug-In must appear first in your PATH.

A.2.8 Troubleshooting Oracle SSO Plug-In

This section describes common problems and solutions.

Oracle Dependency Libraries Not Found

You may not have included *`ORACLE_HOME`* in your path.

Solution

Check to see that you have *`ORACLE_HOME`*`/lib` included in your library path variable on UNIX. On Microsoft Windows, ensure that you have *`ORACLE_HOME`*`\bin` in your path.

Microsoft IIS Oracle SSO Plug-In Does not Work with HTML Authentication

The Oracle SSO Plug-In is designed not to work with other authentication modules. Authentication is either a native listener authentication module or a third-party module.

B Frequently Asked Questions

This appendix provides answers to frequently asked questions about Oracle HTTP Server.

Documentation from the Apache Software Foundation is referenced when applicable.

Note: Readers using this guide in PDF or hard copy formats will be unable to access third-party documentation, which Oracle provides in HTML format only. To access the third-party documentation referenced in this guide, use the HTML version of this guide and click the hyperlinks.

B.1 How Do I Create Application-Specific Error Pages?

Oracle HTTP Server has a default content handler for dealing with errors. You can use the `ErrorDocument` directive to override the defaults.

See Also: ErrorDocument directive in the Apache Server documentation

B.2 What Type of Virtual Hosts Are Supported for HTTP and HTTPS?

For HTTP, Oracle HTTP Server supports both name-based and IP-based virtual hosts. Name-based virtual hosts are virtual hosts that share a common listening address (IP plus port combination), but route requests based on a match between the Host header sent by the client and the `ServerName` directive set within the `VirtualHost`. IP-based virtual hosts are virtual hosts that have distinct listening addresses. IP-based virtual hosts route requests based on the address they were received on.

For HTTPS, only IP-based virtual hosts are possible with Oracle HTTP Server. This is because for name-based virtual hosts, the request must be read and inspected to determine which virtual host is used to process the request. If HTTPS is used, an SSL handshake must be performed before the request can be read. In order to perform the SSL handshake, a server certificate must be provided. In order to have a meaningful server certificate, the hostname in the certificate must match the hostname the client requested, which implies a unique server certificate per virtual host. However, because the server cannot know which virtual host to route the request to until it has read the request, and it can't properly read the request unless it knows which server certificate to provide, there is no way to make name-based virtual hosting work with HTTPS.

Note: This is not a restriction of Oracle HTTP Server; instead, it is a restriction of the HTTPS protocol itself.

B.3 Can I Use Oracle HTTP Server As Cache?

You can use Oracle HTTP Server as a cache by using the `ProxyRequests` and `CacheRoot` directives. In general, however, Oracle recommends using Oracle Web Cache instead. Oracle Web Cache is a content-aware server accelerator and secure reverse proxy server that improves the performance, scalability, and availability of Web sites. For more details, refer to the *Oracle Fusion Middleware Administrator's Guide for Oracle Web Cache*.

See Also: ProxyRequests and CacheRoot directives in the Apache Server documentation

B.4 Can I Use Different Language and Character Set Versions of Document?

Yes, you can use multiviews, a general name given to the Apache server's ability to provide language and character-specific document variants in response to a request.

See Also: Multiviews in the Apache Server documentation

B.5 How do I Send Proxy Sensitive Requests to Oracle HTTP Server Behind a Firewall?

Use the proxy directives, and not the cache directives, to send proxy sensitive requests through firewalls.

B.6 Can I Apply Apache Security Patches to Oracle HTTP Server?

No, you cannot apply the Apache security patches to Oracle HTTP Server for the following reasons:

- Oracle tests and appropriately modifies security patches before releasing them to Oracle HTTP Server users.
- In many cases, the Apache alerts, such as openSSL alerts, may not be applicable because Oracle has removed those components from the stack.
- Oracle releases the patches in a timely manner that the impact of getting the patch from Oracle instead of an open source organization is minimal. The benefit of using an Oracle patch, with respect to supportability, is tremendous.

The latest security related fixes to Oracle HTTP Server are performed through the Oracle Critical Patch Update (CPU). For more details, refer to Oracle's Critical Patch Updates and Security Alerts Web page.

Note: After applying a CPU, the Apache-based version may stay the same, but the vulnerability will be fixed. There are third-party security detection tools that can check the version, but do not check the vulnerability itself.

B.7 Can I Upgrade the Apache Version of Oracle HTTP Server?

No, you cannot upgrade only the Apache version inside Oracle HTTP Server. Oracle provides a newer version of Apache that Oracle HTTP Server is based on, which is part of either a patch update or the next major or minor release of Oracle Fusion Middleware.

B.8 Can I Compress Output From Oracle HTTP Server?

In general, Oracle recommends using Oracle Web Cache for this purpose. There are other freeware modules (for example, mod_gzip) that can be plugged in for this purpose, but their use is not supported. Oracle Web Cache provides efficient delivery of contents by using on-the-fly compression, dynamically learning which MIME types are compressible, and throttling responses to slower network clients. For more details, refer to the *Oracle Fusion Middleware Administrator's Guide for Oracle Web Cache*.

B.9 How Do I Create a Namespace That Works Through Firewalls and Clusters?

The general idea is that all servers in a distributed Web site should use a single URL namespace. Every server serves some part of that namespace, and is able to redirect or proxy requests for URLs that it does not serve to a server that is closer to that URL. For example, your namespaces could be the following:

```
/app1/login.html
/app1/catalog.html
/app1/dologin.jsp
/app2/orderForm.html
/apps/placeOrder.jsp
```

You could initially map these name spaces to two Web servers by putting app1 on server1 and app2 on server2. The configuration for server1 might look like the following:

```
Redirect permanent /app2 http://server2/app2
Alias /app1 /myApps/application1
<Directory /myApps/application1>
  ...
</Directory>
```

The configuration for Server2 is complementary.

If you decide to partition the namespace by content type (HTML on server1, and JSP on server2), then you can change server configuration and move files around, but you do not have to make changes to the application itself. The resulting configuration of server1 might look like the following:

```
RedirectMatch permanent (.*) \.jsp$ http://server2/$1.jsp
AliasMatch ^/app(.*) \.html$ /myPages/application$1.html
<DirectoryMatch "^/myPages/application\d">
```

```
  ...
</DirectoryMatch>
```

The amount of actual redirection can be minimized by configuring a hardware load balancer like F5 system BIG-IP to send requests to server1 or server2 based on the URL.

B.10 How do I Protect the Web Site from Hackers?

There are many attacks by hackers, and new attacks are invented everyday. The following are some general guidelines for securing your site. You can never be completely secure, but you can avoid being an easy target.

- Configure the mod_security module that comes with Oracle HTTP Server. ModSecurity is a Web application firewall that provides an increased external security layer to detect and/or prevent attacks before they reach Web applications.
- Use a commercial firewall, such as Checkpoint FW-1 or Cisco PIX between your ISP and your Web server. Remember not all hackers are outside your organization.
- Use switched Ethernet to limit the amount of traffic a compromised server can detect. Use additional firewalls between Web server machines and highly sensitive internal servers running the database and enterprise applications.
- Remove unnecessary network services such as RPC, Finger, and telnet from your server.
- Carefully validate all input from Web forms. Be especially wary of long input strings and input that contains non-printable characters, HTML tags, or javascript tags.
- Encrypt or randomize the contents of cookies that contain sensitive information to prevent a hacker from hijacking a valid session. For example, it should be difficult to guess a valid sessionID.
- Check often for security patches for all your system and application software, and install them as soon as possible. Be sure these patches come from reliable sources. Only download patches from trusted sites and verify the cryptographic checksum.
- Use an intrusion detection package to monitor for defaced Web pages, viruses, and presence of rootkits that indicate hackers have broken into your site. If possible, mount system executables and Web content on read-only file systems.
- Have a forensic analysis package on hand to capture evidence of a break in as soon as detected. This aids in prosecution of the hackers.

B.11 Do I Need to Re-register Partner Applications with the SSO Server If I Disable or Enable SSL?

Yes, if you enable or disable SSL, you have to re-register partner applications with the SSO server. When you make any changes that affect the URL (for example, changing the hostname or port, or enabling or disabling SSL), you have re-register partner applications with the SSO server because the old URL registered with the SSO server is no longer valid. You have to re-register the partner applications with the new URL.

C Troubleshooting Oracle HTTP Server

This appendix describes common problems that you might encounter when using Oracle HTTP Server, and explains how to solve them. It includes the following topics:

- Section C.1, "Oracle HTTP Server Unable to Start Due to Port Conflict"
- Section C.2, "System Overloaded by Number of httpd Processes"
- Section C.3, "Permission Denied When Starting Oracle HTTP Server On a Port Below 1024"
- Section C.4, "Oracle HTTP Server May Fail To Start If PM Files Are Not Located Correctly"
- Section C.5, "Using Log Files to Locate Errors"

C.1 Oracle HTTP Server Unable to Start Due to Port Conflict

You can get the following error if Oracle HTTP Server is unable to start due to port conflict:

```
[VirtualHost: main] (98)Address already in use: make_sock: could not bind to
address [::]:7777
```

Solution

Determine what process is already using that port, and then either change the IP:port address of Oracle HTTP Server or the port of the conflicting process.

C.2 System Overloaded by Number of httpd Processes

When too many httpd processes are running on a system, the response time degrades because there are insufficient resources for normal processing.

Solution

Lower the value of `MaxClients` to a value the machine can accommodate.

C.3 Permission Denied When Starting Oracle HTTP Server On a Port Below 1024

You will get the following error if you try to start Oracle HTTP Server on a port below 1024:

```
[VirtualHost: main] (13)Permission denied: make_sock: could not bind to address
```

```
[::]:443
```

Oracle HTTP Server will not start on ports below 1024 because root privileges are needed to bind these ports.

Solution

Follow the steps in Section 4.1.2.3, "Starting Oracle HTTP Server on a Privileged Port" to start Oracle HTTP Server on a Privileged Port.

C.4 Oracle HTTP Server May Fail To Start If PM Files Are Not Located Correctly

If Oracle HTTP Server is not able to locate Perl module (PM) files in the path defined in the `PERL5LIB` variable, Oracle HTTP Server may encounter the following errors, and fail to start:

```
[error] Can't locate mod_perl.pm in @INC (@INC contains:$ORACLE_HOME/perl/...)
```

or:

```
[error] Can't locate Apache::Registry.pm in @INC (@INC contains: $ORACLE_
HOME/perl/...)
```

Solution

Check that *`ORACLE_HOME`*`/ohs/bin/apachectl` is correctly defined in the `PERL5LIB` variable. It should point to the path(s) containing the PM files. By default, it points to PM files in the following directories:

```
ORACLE_HOME/ohs/mod_perl/lib/site_perl/5.10.0
ORACLE_HOME/perl/lib/5.10.0
ORACLE_HOME/perl/lib/site_perl/5.10.0
```

C.5 Using Log Files to Locate Errors

You can use the following log files to help locate errors:

- Rewrite Log
- Script Log
- Error Log

C.5.1 Rewrite Log

This log file is necessary for debugging when mod_rewrite is used. The log file produces a detailed analysis of how the rewriting engine transforms requests. The level of detail is controlled by the RewriteLogLevel directive.

See Also: Rewrite Log in the Apache Server documentation.

C.5.2 Script Log

This log file enables you to record the input to and output from the CGI scripts. This should only be used in testing, and not for production servers.

See Also: Script Log in the Apache Server documentation.

C.5.3 Error Log

This log file records overall server problems. Refer to Chapter 7, "Managing Oracle HTTP Server Logs" for details on configuring and viewing error logs.

Glossary

Apache

Apache is a public domain HTTP server derived from the National Center for Supercomputing Applications (NCSA).

authentication

The process of verifying the identity of a user, device, or other entity in a host system, often as a prerequisite to granting access to resources in a system. A recipient of an authenticated message can be certain of the message's origin (its sender). Authentication is presumed to preclude the possibility that another party has impersonated the sender.

availability

The percentage or amount of scheduled time that a computing system provides application service.

certificate

Also called a **digital certificate**. An ITU x.509 v3 standard data structure that securely binds an identity to a public key.

A certificate is created when an entity's public key is signed by a trusted identity, a **certificate authority** The certificate ensures that the entity's information is correct and that the public key actually belongs to that entity.

A certificate contains the entity's name, identifying information, and public key. It is also likely to contain a serial number, expiration date, and information about the rights, uses, and privileges associated with the certificate. It also contains information about the certificate authority that issued it.

certificate authority

A trusted third party that certifies that other entities—users, databases, administrators, clients, servers—are who they say they are. When it certifies a user, the certificate authority first seeks verification that the user is not on the certificate revocation list (CRL), then verifies the user's identity and grants a certificate, signing it with the certificate authority's private key. The certificate authority has its own certificate and public key which it publishes. Servers and clients use these to verify signatures the certificate authority has made. A certificate authority might be an external company that offers certificate services, or an internal organization such as a corporate MIS department.

CGI

Common Gateway Interface (CGI) is the industry-standard technique for transferring information between a Web server and any program designed to accept and return data that conforms to the CGI specifications.

ciphertext

Data that has been encrypted. Ciphertext is unreadable until it has been converted to plain text (decrypted) with a key. See **decryption**.

cleartext

See **plaintext**.

cryptography

The art of protecting information by transforming it (encrypting) into an unreadable format. See **encryption**.

DAD

See **database access descriptor**.

database access descriptor

A database access descriptor (DAD) is a set of values that specify how an application connects to an Oracle database to fulfill an HTTP request. The information in the DAD includes the username (which also specifies the schema and the privileges), password, connect-string, error log file, standard error message, and national language support (NLS) parameters such as NLS language, NLS date format, NLS date language, and NLS currency.

decryption

The process of converting the contents of an encrypted message (**ciphertext**) back into its original readable format (**plaintext**).

digital certificate

See **certificate**.

digital wallet

See **wallet**.

encryption

The process of converting a message thereby rendering it unreadable to any but the intended recipient. Encryption is performed by converting data into code that cannot be understood by unauthorized people or systems. There are two main types of encryption: **public-key encryption** (also known as asymmetric-key encryption) and symmetric-key encryption.

entry

In the context of a directory service, entries are the building blocks of a directory. An entry is a collection of information about an object in the directory. Each entry is composed of a set of attributes that describe one particular trait of the object. For example, if a directory entry describes a person, that entry can have attributes such as first name, last name, telephone number, or e-mail address.

failover

The ability to reconfigure a computing system to utilize an alternate active component when a similar component fails.

Fusion Middleware Control

See **Oracle Enterprise Manager Fusion Middleware Control**.

HTTP

See **Hypertext Transfer Protocol**.

Hypertext Transfer Protocol

Hypertext Transfer Protocol (HTTP) is the underlying format used by the Web to format and transmit messages and determine what actions Web servers and browsers should take in response to various commands. HTTP is the protocol used between Oracle Fusion Middleware and clients.

LDAP

See **Lightweight Directory Access Protocol**.

Lightweight Directory Access Protocol

A standard, extensible directory access protocol. It is a common language that LDAP clients and servers use to communicate. The framework of design conventions supporting industry-standard directory products, such as the Oracle Internet Directory.

modules

Modules extend the basic functionality of a Web server, and support integration between Oracle HTTP Server and other Oracle Fusion Middleware components.

Oracle Enterprise Manager Fusion Middleware Control

Oracle Enterprise Manager Fusion Middleware Control (Fusion Middleware Control) provides Web-based management tools designed specifically for Oracle Fusion Middleware. Using Fusion Middleware Control, you can monitor and configure the components of your application server, such as deploy applications, manage security, and create and manage Oracle Fusion Middleware clusters.

PEM

Privacy-enhanced Electronic Mail. An **encryption** technique that provides encryption, authentication, message integrity, and key management.

PL/SQL

PL/SQL is the Oracle proprietary extension to the SQL language. PL/SQL adds procedural and other constructs to SQL that make it suitable for writing applications.

plaintext

Also called cleartext. Unencrypted data in ASCII format.

plug-in

A module that adds a specific feature or service to a larger system. For example, Oracle Proxy Plug-in or Oracle SSO Plug-in.

port

A port is a number that TCP uses to route transmitted data to and from a particular program.

private key

In **public-key cryptography**, this key is the secret key. It is primarily used for decryption but is also used for encryption with digital signatures. See **public/private key pair**.

proxy server

A proxy server typically resides on a network firewall and allows clients behind the firewall to access Web resources. All requests from clients go to the proxy server rather than directly to the destination server. The proxy server forwards the request to the destination server and passes the received information back to the client. The proxy server channels all Web traffic at a site through a single, secure port; this allows an organization to create a secure firewall by preventing Internet access to internal systems, while allowing Web access.

public key

In **public-key cryptography**, this key is made public to all. It is primarily used for encryption but can be used for verifying signatures. See **public/private key pair**.

public-key cryptography

Encryption method that uses two different random numbers (keys). See **public key** and **public-key encryption**.

public-key encryption

The process where the sender of a message encrypts the message with the public key of the recipient. Upon delivery, the message is decrypted by the recipient using its private key.

public/private key pair

A set of two numbers used for **encryption** and **decryption**, where one is called the **private key** and the other is called the **public key**. Public keys are typically made widely available, while private keys are held by their respective owners. Though mathematically related, it is generally viewed as computationally infeasible to derive the private key from the public key. Public and private keys are used only with asymmetric encryption algorithms, also called **public-key encryption** algorithms, or public-key cryptosystems. Data encrypted with either a public key or a private key from a key pair can be decrypted with its associated key from the key-pair. However, data encrypted with a public key cannot be decrypted with the same public key, and data encrypted with a private key cannot be decrypted with the same private key.

RSA

A **public-key encryption** technology developed by RSA Data Security. The RSA algorithm is based on the fact that it is laborious to factor very large numbers. This makes it mathematically unfeasible, because of the computing power and time required to decode an RSA key.

scalability

A measure of how well the software or hardware product is able to adapt to future business needs.

Secure Sockets Layer

Secure Sockets Layer (SSL) is a standard for the secure transmission of documents over the Internet using HTTPS (secure HTTP). SSL uses digital signatures to ensure that transmitted data is not tampered with.

single sign-on

Single sign-on enables a you to authenticate once, combined with strong authentication occurring transparently in subsequent connections to other databases or applications. It lets you access multiple accounts and applications with a single password, entered during a single connection.

SSL

See **Secure Sockets Layer**.

wallet

Also called a digital wallet. A wallet is a data structure used to store and manage security credentials for an individual entity. It implements the storage and retrieval of credentials for use with various cryptographic services. A **Wallet Resource Locator** (WRL) provides the necessary information to locate the wallet.

Wallet Resource Locator

A wallet resource locator (WRL) provides all necessary information to locate a wallet. It is a path to an operating system directory that contains a wallet.

WRL

See **Wallet Resource Locator**.

X.509

A standard for creating digital certificates.

Index