Oracle® Fusion Middleware

Using Oracle WebLogic Server Proxy Plug-Ins

ORACLE®

Oracle Fusion Middleware Using Oracle WebLogic Server Proxy Plug-Ins, 12c (12.2.1.2.0)

E78107-04

Contents

Preface

1   Overview of Oracle WebLogic Server Proxy Plug-In

2   Configuring the Plug-In for Oracle HTTP Server

3   Configuring the Plug-In for Apache HTTP Server

4   Configuring the Plug-In for iPlanet Web Server

5   Configuring the Plug-In for Microsoft IIS Web Server

6   Configuring Security

7   Common Configuration Tasks

8   Parameters for Web Server Plug-Ins

Preface

This preface describes the document accessibility features and conventions used in this guide—*Using Oracle WebLogic Server Proxy Plug-Ins 12.2.1.2*.

Documentation Accessibility

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

Access to Oracle Support

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

Conventions

The following text conventions are used in this document:

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

1

Overview of Oracle WebLogic Server Proxy Plug-In

This chapter describes the plug-ins provided by Oracle for use with WebLogic Server:

- What are Oracle WebLogic Server Proxy Plug-Ins?
- Availability of Oracle WebLogic Server Proxy Plug-In
- Upgrading from 1.0 Plug-Ins
- Features of the Version 12.2.1.2.0 Plug-Ins
- Support and Patching

1.1 What are Oracle WebLogic Server Proxy Plug-Ins?

Web server plug-ins allow requests to be proxied from Oracle HTTP Server, Oracle iPlanet Web Server, Apache HTTP Server, or Microsoft Internet Information Server (IIS) to Oracle WebLogic Server. In this way, plug-ins enable the HTTP server to communicate with applications deployed on the WebLogic Server.

The plug-in enhances an HTTP server installation by allowing Oracle WebLogic Server to handle requests that require dynamic functionality. In other words, you typically use a plug-in where the HTTP server serves static pages such as HTML pages, while Oracle WebLogic Server serves dynamic pages such as HTTP Servlets and Java Server Pages (JSPs).

Oracle WebLogic Server may be operating in a different process, possibly on a different host. To the end user—the browser—the HTTP requests delegated to Oracle WebLogic Server still appear to be coming from the HTTP server.

In addition, the HTTP-tunneling facility of the WebLogic client/server protocol also operates through the plug-in, providing access to all Oracle WebLogic Server services.

1.1.1 Connection Pooling and Keep-Alive

The plug-ins improve performance using a pool of connections from the plug-in to Oracle WebLogic Server. The plug-in implements HTTP 1.1 keep-alive connections between the plug-in and Oracle WebLogic Server by reusing the same connection for subsequent requests from the same plug-ins. If the connection is inactive for more than 20 seconds, (or a user-defined amount of time), the connection is closed. See KeepAliveEnabled.

> **Note:**
>
> The web server manages client connections.

1.1.2 Proxying Requests

The plug-in proxies requests to Oracle WebLogic Server based on a configuration that you specify.

- You can proxy requests based on the URL of the request or a portion of the URL. This is called proxying by path.

- You can also proxy a request based on the MIME type of the requested file, which is called proxying by file extension.

You can also enable both methods. If you enable both methods and a request matches both criteria, the request is proxied by path.

You can also specify additional parameters for each of these types of requests that define additional behavior of the plug-in.

1.2 Availability of Oracle WebLogic Server Proxy Plug-In

Oracle WebLogic Server Proxy Plug-Ins are available for the following web servers:

Table 1-1   Availability of Version 12*c* (12.2.1.2.0) Plug-Ins

Web Server	Plug-In Availability	More Information
Oracle HTTP Server 12c	The plug-in is included in the Oracle HTTP Server installation.	For information about configuring this plug-in, see Configuring the Plug-In for Oracle HTTP Server .
Oracle iPlanet Web Server (7.0.9 and later releases) Apache HTTP Server 2.2.x Microsoft Internet Information Server (IIS) 7.0 and later	The plug-ins are available for download on the My Oracle Support (http://support.oracle.com) and Software Delivery Cloud (http://edelivery.oracle.com) web sites as zip files containing the necessary binary and helper files. For example, the following directories are included in the mod_wl.so plug-in distribution. • lib/mod_wl.so (Apache HTTP Server plug-in) • lib/*.so (native libraries) • bin/orapki or bin\orapki.bat (orapki tool) • jlib/*.jar (Java helper libraries for orapki)	For information about installing and configuring the plug-ins for Apache HTTP Server, Oracle iPlanet, and Microsoft IIS Web Servers, see the following: • Configuring the Plug-In for Apache HTTP Server • Configuring the Plug-In for iPlanet Web Server • Configuring the Plug-In for Microsoft IIS Web Server

1.3 Upgrading from 1.0 Plug-Ins

The version 1.0 plug-ins are deprecated and are not guaranteed to be available for future versions of Oracle WebLogic Server. The version 12*c* (12.2.1.2.0) plug-ins are the recommended replacement.

ORACLE®

> **✎ Note:**
>
> For Apache HTTP Server 1.3.x or 2.0.x, continue to use the version 1.0 plug-in.

This section contains the following information:

- Upgrade Instructions
- Considerations for Upgrading From Oracle WebLogic Server Proxy Plug-Ins Version 1.0 to 12*c* (12.2.1.2.0)

1.3.1 Upgrade Instructions

For upgrading from 11g plug-ins to the Oracle WebLogic Server Proxy Plug-Ins 12*c* (12.2.1.2.0), use installation instructions included in the specific chapter for your web server, as listed in Table 1-2.

Table 1-2   Upgrade Instructions by Plug-In

To upgrade to the 12*c* (12.2.1.2.0) plug-ins for:	See:
Oracle HTTP Server	Configuring the Plug-In for Oracle HTTP Server
Apache HTTP Server	Configuring the Plug-In for Apache HTTP Server
iPlanet Web Server	Configuring the Plug-In for iPlanet Web Server
Microsoft IIS Web Server	Configuring the Plug-In for Microsoft IIS Web Server

1.3.2 Considerations for Upgrading From Oracle WebLogic Server Proxy Plug-Ins Version 1.0 to 12*c* (12.2.1.2.0)

The version 12*c* (12.2.1.2.0) plug-ins are a superset of the version 1.0 plug-ins and support the existing features. However, when you upgrade, keep the following considerations in mind:

- The list of supported platforms has changed. See *Oracle Fusion Middleware Supported System Configurations* at:

  http://www.oracle.com/technetwork/middleware/ias/downloads/fusion-certification-100350.html

- Because the version 1.0 plug-ins supported both 40- and 128-bit encryption standards, the plug-in file names needed to identify which standard was supported. For example, mod_wl_22.so indicated 40-bit encryption and mod_wl128_22.so indicated 128-bit encryption. However, the version 12*c* (12.2.1.2.0) plug-ins support only 128-bit encryption, and the plug-in names are now simplified. For example, mod_wl.so is the only file name required.

> **Note:**
>
> If you upgrade from the 1.0 plug-ins and had been using 128-bit encryption, you must change your configuration file to reflect the new naming convention. For example, you must change mod_wl128_22.so to mod_wl.so.

1.4 Features of the Version 12.2.1.2.0 Plug-Ins

This section describes the additional features of the version 12*c* (12.2.1.2.0) plug-ins.

- Oracle WebLogic Server Proxy Monitoring
- Support for Multi-tenancy and Partitions
- SSL Support for IPlanet Web Server
- Documentation for Using the Plug-In with Microsoft IIS
- Deprecated Support for Certificates Signed Using the MD5 Algorithm

1.4.1 Oracle WebLogic Server Proxy Monitoring

The current release adds support for monitoring the performance of the Oracle HTTP Server. The performance metrics are specific to the Oracle WebLogic Server Proxy Plug-In where a request is proxied to the backend WebLogic server. See Understanding Oracle WebLogic Server Proxy Plug-In Performance Metrics.

1.4.2 Support for Multi-tenancy and Partitions

In the current release, Oracle WebLogic Server Proxy Plug-Ins can be made partition-aware. This plays a role in support of Oracle WebLogic Server MT (multi-tenancy). See Working with Partitions.

1.4.3 SSL Support for IPlanet Web Server

In the current release, support has been added for using SSL with the iPlanet Web Server. See Configuring SSL with the Plug-In for iPlanet Web Server.

1.4.4 Documentation for Using the Plug-In with Microsoft IIS

Instructions have been added to use Oracle WebLogic Server Proxy Plug-Ins with Microsoft IIS Web Server. See Installing and Configuring the Plug-In for Microsoft Internet Information Server.

1.4.5 Deprecated Support for Certificates Signed Using the MD5 Algorithm

Support for certificates signed with MD5 is deprecated. See Replacing Certificates Signed Using the MD5 Algorithm.

1.5 Support and Patching

When you encounter issues with a plug-in, always report the version of the plug-in you are using. You can find this information in the apache log or the plug-in debug log (if configured). The version information looks like the following snippet:

```
WebLogic Server Plug-in version 12.2.1.2.0 <WLSPLUGINS_XXXX_XXXX_XXXXX.XXXX>
```

> **Note:**
>
> On the Apache Web Server for Linux, you can also obtain the plug-in version by issuing the following command:
>
> ```
> $ strings ${PLUGIN_HOME}/lib/mod_wl.so | grep -i wlsplugins
> ```

A patch for a plug-in typically will contain one or more shared objects to be replaced. Ensure to backup your original files as you replace them with those in the patch. Validate that the patch has been correctly updated by checking the version string in the logs.

2
Configuring the Plug-In for Oracle HTTP Server

Configure the Oracle WebLogic Server Proxy Plug-In, which is the plug-in for proxying requests from Oracle HTTP Server to Oracle WebLogic Server. The Oracle WebLogic Server Proxy Plug-In is included in the Oracle HTTP Server 12c (12.2.1.2.0) installation. You do not have to download and install it separately.

> **Note:**
>
> The Oracle WebLogic Server Proxy Plug-In provides features that are identical to those of the plug-in for Apache HTTP Server.

You can configure the Oracle WebLogic Server Proxy Plug-In either by using Fusion Middleware Control or by editing the mod_wl_ohs.conf configuration file manually.

WebLogic Server Proxy Plug-in 12.2.1.0 and later version builds are moved from Intel compiler to MSVC Compiler. When Apache HTTP Server is used as a front end with WebLogic Server Proxy Plug-in, the plug-in library depends on the two dlls — `msvcp110.dll` and `msvcr110.dll`, provided by Microsoft. These dlls are available with Microsoft Visual C ++ Redistributable Package for x64.

This chapter contains the following topics:

- Oracle HTTP Server Support Note
- Preparing for Configuring the Oracle WebLogic Server Proxy Plug-In
- Configuring the Oracle WebLogic Server Proxy Plug-In Using Fusion Middleware Control
- Configuring the Oracle WebLogic Server Proxy Plug-In Manually
- Understanding Oracle WebLogic Server Proxy Plug-In Performance Metrics
- Deprecated Directives for Oracle HTTP Server
- Troubleshooting Oracle WebLogic Server Proxy Plug-In Implementations

2.1 Oracle HTTP Server Support Note

The Oracle WebLogic Server Proxy Plug-In for Oracle HTTP Server is now able to front-end WebSocket applications.

2.2 Preparing for Configuring the Oracle WebLogic Server Proxy Plug-In

You must complete some installation and verification tasks before configuring the Oracle WebLogic Server Proxy Plug-In.

- Ensure that Oracle WebLogic Server has been installed, a domain has been created, and you can access the Oracle WebLogic Server administration console. Oracle HTTP Server and WebLogic Server can be installed either in same domain or in separate domains.

- Verify that Fusion Middleware Control has been installed and you can access the Enterprise Manager Console. This is required to configure the Oracle WebLogic Server Proxy Plug-In by using the graphical interface provided by Fusion Middleware Control. The Fusion Middleware Control is available only for WebLogic managed domains.

- To be able to test the configuration, ensure that the required Java applications are deployed to Oracle WebLogic Server—either to a single managed server or to a cluster—and are accessible.

2.2.1 Setting the WebLogic Plug-In Enabled Parameter

You must set the WebLogic Plug-In Enabled parameter if the version of the Oracle WebLogic Server instances in the back end is 10.3.4 (or later) release.

1. Log in to the Oracle WebLogic Server administration console.
2. In the Domain Structure pane, expand the **Environment** node.
   - If the server instances to which you want to proxy requests from Oracle HTTP Server are in a cluster, select **Clusters**.
   - Otherwise, select **Servers**.
3. Select the server or cluster to which you want to proxy requests from Oracle HTTP Server.
4. Select WebLogic Server drop down menu, then Administration, then General Settings.

   The Configuration: General tab is displayed.
5. Scroll down to the Advanced section, expand it, and select **Yes** from the **WebLogic Plug-In Enabled** drop-down list.

   Yes must be selected if the WebLogic Plug-ins are used with the WebLogic Server. See Understanding the WebLogic Plug-In Enabled Parameter.
6. If you selected **Servers** in step 2, repeat steps 3 and 4 for the other servers to which you want to proxy requests from Oracle HTTP Servers.
7. Click **Save**.

For the change to take effect, you must restart the server instances.

2.2.1.1 Understanding the WebLogic Plug-In Enabled Parameter

The **WebLogic Plug-In Enabled** drop-down list contains these values:

- **Yes**—**Yes** must be selected if the WebLogic Plug-ins are used with the WebLogic Server. When set to **Yes** on the server, it specifies that this server uses the proprietary `WL-Proxy-Client-IP` header, which is recommended if the server instance will receive requests from a proxy plug-in.

  When set to **Yes** on the cluster, it specifies that the cluster will receive requests from a proxy plug-in or `HttpClusterServlet`. A call to `getRemoteAddr` will return the address of the browser client from the proprietary `WL-Proxy-Client-IP` header, instead of the Web server.

- **No**—Selecting **No** for the server or cluster disables the `weblogic-plugin-enabled` parameter (`weblogic-plugin-enabled=false`) in the `config.xml` file.

- **Default**—When **Default** is selected for **WebLogic Plug-In Enabled** in the servers page, then the servers will inherit the value selected for **WebLogic Plug-In Enabled** for the cluster. When **Default** is selected for **WebLogic Plug-In Enabled** in the clusters page, then the clusters will inherit the value selected for **WebLogic Plug-In Enabled** for the domain.

2.3 Configuring the Oracle WebLogic Server Proxy Plug-In Using Fusion Middleware Control

Use Fusion Middleware Control to configure the mod_wl_ohs module. To configure the mod_wl_ohs module, complete the following tasks:

- Task 1: Navigate to the mod_wl_ohs Configuration Page
- Task 2: Specify the Configuration Settings
- Task 3: Configure Expression Overrides or Location Overrides (Optional)
- Task 4: Apply Your Changes

2.3.1 Task 1: Navigate to the mod_wl_ohs Configuration Page

The mod_wl_ohs configuration page contains the parameters for configuring the Oracle WebLogic Server Proxy Plug-In.

1. Ensure that you have fulfilled the prerequisites listed in Preparing for Configuring the Oracle WebLogic Server Proxy Plug-In.

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

3. Select **mod_wl_ohs Configuration** from the Administration menu. The **mod_wl_ohs Configuration** page appears.
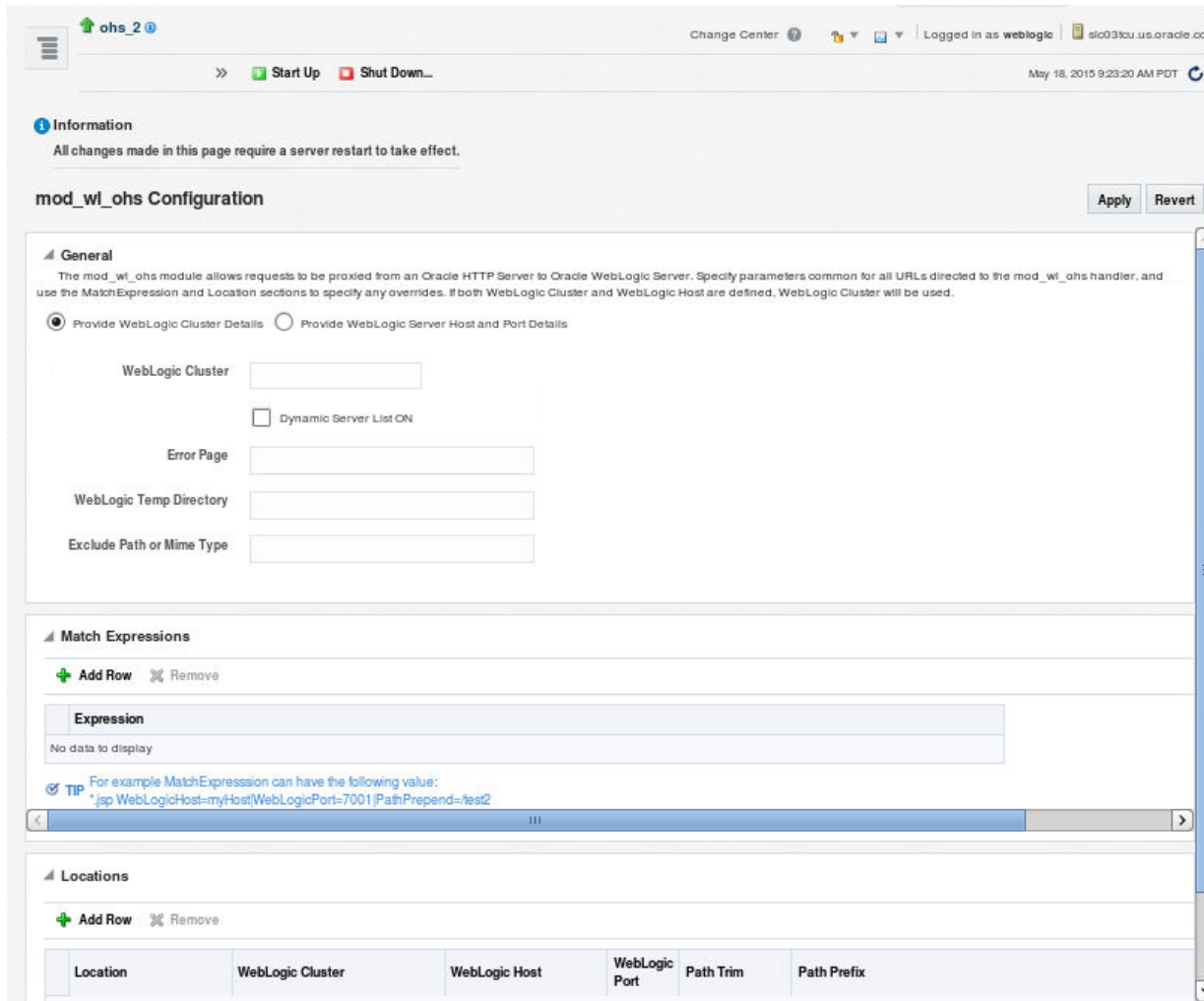
The following table describes the fields in the **mod_wl_ohs** page.

Field	Description
Provide WebLogic Cluster Details	List of Oracle WebLogic clusters that can be used for load balancing. 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.
	If you are not sure of the correct cluster, you can click the search icon to see a list of all associated clusters. See Using the Search Function.
	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. Oracle 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.
Provide WebLogic Server Host and Port Details	• WebLogic Host
	Oracle WebLogic Server host (or virtual host name as defined in Oracle WebLogic Server) to which HTTP requests should be forwarded. If you are using a WebLogic cluster, use the **WebLogic Cluster** parameter instead of **WebLogic Host**.
	If you are not sure of the correct server, you can click the search icon to see a list of all associated clusters. See Using the Search Function.
	• WebLogic Port
	Port at which the Oracle WebLogic Server host is listening for connection requests from the module (or from other servers). (If you are using SSL between the module and Oracle WebLogic Server, set this parameter to the SSL listen port.)
Dynamic Server List ON | OFF	When set to OFF, the module ignores the dynamic cluster list used for load balancing requests proxied from the module and only uses the static list specified with the WebLogic Cluster parameter. Normally this parameter should be set to ON.
	There are some implications for setting this parameter to OFF:
	• If one or more servers in the static list fails, the module could waste time trying to connect to a terminated server, resulting in decreased performance.
	• If you add a new server to the cluster, the module cannot proxy requests to the new server unless you redefine this parameter. Oracle WebLogic Server automatically adds new servers to the dynamic server list when they become part of the cluster.
Error Page	You can create your own error page to appear when your Web server cannot forward requests to Oracle WebLogic Server.

Field	Description
WebLogic Temp Directory	Specifies the location of the _wl_proxy directory for post data files.
Exclude Path or MIME Type	This parameter allows you exclude certain requests from proxying.
	This parameter can be defined locally at the Location tag level and globally. When the property is defined locally, it does not override the global property but defines a union of the two parameters.
Match Expressions	Use this region to specify any Expression overrides. For example, if you were proxying by MIME type, you might enter:
	`*.jsp WebLogicHost=myHost|paramName=value`
	You can define a new parameter for Match Expression by using the following syntax:
	`*.jsp PathPrepend=/test|PathTrim=/foo`
	(parameters are separated by a |)
Location	Use this table to specify any location overrides. See Task 3: Configure Expression Overrides or Location Overrides (Optional).
Add Cross Component Wiring	This button appears only if you have installed Oracle HTTP Server in full JRF mode (collocated) and there is a backing database.
	Selecting this button opens the Service Tables page. A service table provides a way for service providers to publish endpoint information about their services, and clients of these services to query and bind to these services. A service table is a single table in a database schema. There is one row for every endpoint that is published to it. The service table schema is initially created by the Repository Creation Utility.
	See Wiring Components to Work Together in *Administering Oracle Fusion Middleware*

2.3.2 Task 2: Specify the Configuration Settings

Specify the configuration settings for the Oracle WebLogic Server Proxy Plug-In. In the General section, you can configure mod_wl_ohs for a WebLogic cluster or for WebLogic servers.

- If you select the **Provide WebLogic Cluster Details** radio button, then provide values for the WebLogic Cluster, Dynamic Server List ON, Error Page, WebLogic Temp Directory, and Exclude Path or MIME Type fields.

- If you select the **Provide WebLogic Server Host and Port Details** radio button, then provide values for the WebLogic Host, WebLogic Port, Dynamic Server List ON, Error Page, WebLogic Temp Directory, and Exclude Path or MIME Type fields.

2.3.3 Task 3: Configure Expression Overrides or Location Overrides (Optional)

If necessary, you can add expression or location overrides to your configuration.

1. Add any expression overrides in the **Match Expression** field.

2. Add any location overrides in the **Location** table.

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

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

   c. Complete the **WebLogic Cluster**, **WebLogic Host**, and **WebLogic Port** fields. You can automatically complete these fields by clicking **AutoFill** (see Using the AutoFill Function).

   d. Complete the **Path Trim** field.

   According to 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:**
   >
   > If you are converting an existing third-party server to proxy requests to WebLogic Server using the module for the first time, you must change application paths to /foo to include weblogic/foo. You can use Path Trim and Path Prepend in combination to change this path

   e. Complete the **Path Prepend** field.

   According to 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:**
>
> If you need to append File Name, use the `DefaultFileName` module parameter instead of Path Prepend.

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

2.3.4 Task 4: Apply Your Changes

Apply your changes to the mod_wl_ohs Configuration Page and restart Oracle HTTP Server.

1. 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.

2. Restart Oracle HTTP Server by selecting **Control** from the Oracle HTTP Server menu, and then selecting **Start Up**.

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

2.3.5 Using the Search Function

The search function allows you to search for a particular WebLogic Cluster or WebLogic Host that is available to the selected Oracle HTTP Server instance. By clicking the search icon 🔍, you can see a list of clusters or servers available to the selected Oracle HTTP Server instance. To use the search function, do the following:

1. Click the search icon for either WebLogic Cluster or WebLogic Host. The Select WebLogic Cluster/Server dialog box appears.

2. Select the cluster or server you want to use and click **OK**.

The selected cluster or server name appears in the appropriate field.

2.3.6 Using the AutoFill Function

> **Note:**
>
> The AutoFill function is available only if you are using Oracle WebLogic Server in full-JRF mode. It is not available if you are using Restricted-JRF.

You can easily add valid WebLogic Server and endpoint locations for a specified Base URL to the Locations table on the Oracle WebLogic Server Proxy Plug-In Configuration screen by using the AutoFill button. To do so:

1. Click **Add** to add a new location,

2. Enter a location name in the Location field.

3. Click **AutoFill**.

Data for any location of the same name will be updated and any new locations will be added to the table.

2.4 Configuring the Oracle WebLogic Server Proxy Plug-In Manually

Specify directives in the `mod_wl_ohs.conf` file to manually configure the Oracle WebLogic Server Proxy Plug-In.

1. Ensure that you have fulfilled the prerequisites listed in Preparing for Configuring the Oracle WebLogic Server Proxy Plug-In.

2. Open the `mod_wl_ohs.conf` file in a text editor.

   The `mod_wl_ohs.conf` file is located in the following directory:

   `DOMAIN_HOME/config/fmwconfig/components/OHS/componentHome`

3. Add directives within the `<IfModule weblogic_module>` element in the configuration file.

   For examples, see Examples of <IfModule weblogic_module> Element Configurations.

   For information about the other directives that you can specify in the `mod_wl_ohs.conf` file, see Parameters for Web Server Plug-Ins.

4. Restart Oracle HTTP Server by using one of the techniques described in Starting Oracle HTTP Serverin *Administering Oracle HTTP Server*.

2.4.1 Examples of <IfModule weblogic_module> Element Configurations

The configuration of the predefined `<IfModule weblogic_module>` element determines how requests are sent to Oracle WebLogic Server. These examples demonstrates the different ways in which you can configure this element.

> ✎ **Note:**
>
> Oracle recommends that you specify directives within the predefined `<IfModule weblogic_module>` element.
>
> If you specify directives outside the predefined `<IfModule weblogic_module>` element, or in additional `<IfModule weblogic_module>` elements, or in configuration files other than mod_wl_ohs.conf, the Oracle WebLogic Server Proxy Plug-In might work, but the configuration state of the module, as displayed in Fusion Middleware Control, could be inconsistent with the directives specified in the mod_wl_ohs.conf configuration file.

To Forward Requests to a Single Oracle WebLogic Server Instance

To forward requests to an application running on a **single Oracle WebLogic Server instance**, specify the details of that destination server within a `<location>` element.

Syntax:

```
<IfModule weblogic_module>
<Location path>
WLSRequest On
WebLogicHost host
WeblogicPort port
</Location>
</IfModule>
```

Example:

With the following configuration, requests for the `/myapp1` URI received at the Oracle HTTP Server listen port will be forwarded to `/myapp1` on the Oracle WebLogic Server with the listen port `localhost:7001`

```
<IfModule weblogic_module>
<Location /myapp1>
WLSRequest On
WebLogicHost localhost
WeblogicPort 7001
</Location>
</IfModule>
```

To Forward Requests to a Cluster of Oracle WebLogic Server Instances

To forward requests to an application running on a **cluster of Oracle WebLogic Server instances**, specify the details of that destination cluster within a new `<location>` element.

Syntax:

```
<IfModule weblogic_module>
<Location path>
WLSRequest On
WebLogicCluster host:port,host:port,...
</Location>
</IfModule>
```

Example:

With the following configuration, requests for the `/myapp2` URI received at the Oracle HTTP Server listen port will be forwarded to `/myapp2` on the Oracle WebLogic Server cluster containing the managed servers with the listen ports `localhost:8002` and `localhost:8003`.

```
<IfModule weblogic_module>
<Location /myapp2>
WLSRequest On
WebLogicCluster localhost:8002,localhost:8003
</Location>
</IfModule>
```

To Configure Multiple Destinations

To configure multiple destinations—say, an application running on a single Oracle WebLogic Server instance and another application running on a cluster—you must specify each destination in a distinct `<location>` child element. All the `<location>` child elements should be at the same level within the `<IfModule weblogic_module>` element, as shown in the following syntax:

```
<IfModule weblogic_module>
#For an application running on a single server instance
<Location path1>
WLSRequest On
WebLogicHost host
WeblogicPort port
</Location>

#For an application running on a cluster
<Location path2>
WLSRequest On
WebLogicCluster host:port,host:port,...
</Location>

</IfModule>
```

To Link to Managed Servers

To configure the Oracle WebLogic Server Proxy Plug-In so that it can link to managed servers, for example to enable a high availability deployment of Oracle HTTP Server, edit the mod_wl_ohs.conf file as follows:

```
<IfModule mod_weblogic.c>
      WebLogicCluster apphost1.mycompany.com:7050,apphost2.mycompany.com:7050
      MatchExpression *.jsp
 </IfModule>

<Location /weblogic>
  WLSRequest On
  WebLogicCluster apphost1.mycompany.com:7050,apphost2.com:7050
  DefaultFileName index.jsp
</Location>
```

> ✎ **Note:**
>
> If you are using SSL termination and routing requests to WebLogic, the following additional configuration is required.
>
> In the WebLogic console, **WebLogic Plugin Enabled** must be set to true, either at the domain, cluster or Managed Server level.
>
> In the Location block which directs requests to the WebLogic managed servers, one of the following lines also must be added.
>
> ```
> WLProxySSL ON
> WLProxySSLPassThrough ON
> ```
>
> (To help determine which parameter to use, see SSL Parameters for Web Server Plug-Ins.)
>
> For example:
>
> ```
> <Location /weblogic>
>   WLSRequest On
>   WebLogicCluster apphost1.mycompany.com:7050,apphost2.com:7050
>   WLProxySSL On
>   WLProxySSLPassThrough ON
>   DefaultFileName index.jsp
> </Location>
> ```
>
> After enabling the WebLogic plugin, restart the Administration Server. See Terminating SSL Requests in *Administering Oracle HTTP Server*.

These examples show two different ways of routing requests to Oracle WebLogic managed servers:

- The `<IfModule>` block sends any requests ending in *.jsp to the WebLogic Managed Server cluster located on Apphost1 and Apphost2.
- The `<Location>` block sends any requests with URLs prefixed by `/weblogic` to the WebLogic Managed Server cluster located on Apphost1 and Apphost2.

To Configure One-way and Two-way SSL

For information about configuring the Oracle WebLogic Server Proxy Plug-In to support one-way and two-way SSL between Oracle HTTP Server and Oracle WebLogic Server, see Using SSL with Plug-Ins.

2.5 Understanding Oracle WebLogic Server Proxy Plug-In Performance Metrics

Oracle HTTP Server provides performance metrics specific to the Oracle WebLogic Server Proxy Plug-In (`mod_wl_ohs`) module, where a request is proxied to the backend WebLogic server.

These metrics are provided through the Oracle Dynamic Monitoring Service (DMS) which enables Oracle Fusion Middleware components to provide administration tools, such as Fusion Middleware Control, with data regarding the component's

performance, state and on-going behavior. For the Oracle WebLogic Server Proxy Plug-In module, for example, it could return the number of requests proxied, the number of failed requests, and other specific metrics. For more information on DMS, see Using the Oracle Dynamic Monitoring Service in *Tuning Performance Guide*.

> **✎ Note:**
>
> The Oracle WebLogic Server Proxy Plug-In module metrics are available only for Oracle HTTP Server and Apache server plug-ins. They are not available for Microsoft IIS and iPlanet server plug-ins.

This section contains the following information on DMS metrics.

- Configuring DMS Metrics for Oracle HTTP Server Proxy Plug-in—How to configure DMS metrics for Oracle WebLogic Server.

- Viewing Performance Metrics for Oracle HTTP Server Proxy Plug-in—How to view DMS metrics for Oracle WebLogic Server.

- DMS State Metrics—These metrics represent a single data point, for example, a counter, a status and so on.

- DMS Event Metrics—These metrics represent an event, for example a login failure and so on.

- DMS PhaseEvent Metrics—These metrics represent a "phase" of an event.

2.5.1 Configuring DMS Metrics for Oracle HTTP Server Proxy Plug-in

The DMS metrics for Oracle WebLogic Server Proxy Plug-In are enabled by default in the `admin.conf` file. They are included as part of the regular DMS metrics collection.

2.5.2 Viewing Performance Metrics for Oracle HTTP Server Proxy Plug-in

You can view the performance metrics by using either the administration port, WLST commands or Fusion Middleware Control. For details of each of the performance metrics, see DMS State Metrics, DMS Event Metrics, and DMS PhaseEvent Metrics.

Using the Administration Port:

If administration port is configured, for example, at `127.0.0.1:9999`, then you can view the raw DMS metrics at the URL `http://127.0.0.1/dms/`.

The metrics under the section `/WebLogicProxy [type=OHSWebLogic]` are the metrics coming from Oracle WebLogic Server plug-in.

Using WLST (Collocated Mode Only)

Use the WLST command `displayMetricTables` to view performance metrics, for example:

```
displayMetricTables(servertype="OHS", servers=<instancename>)
```

The metrics under the section `/WebLogicProxy [type=OHSWebLogic]` are the metrics coming from Oracle WebLogic Server Proxy Plug-in.

Using Fusion Middleware Control (Collocated Mode Only)

To view performance metrics in Fusion Middleware Control, select Oracle HTTP Server, then Monitoring, then Performance Summary. The metrics towards the bottom of this page will have Oracle WebLogic Server Proxy Plug-in specific metrics. See Viewing Performance Metrics in *Administering Oracle HTTP Server*.

2.5.3 DMS State Metrics

A state metric tracks system status information or to track a metric that is not associated with an event. Table 2-1 describes the State metrics available for the Oracle WebLogic Server Proxy Plug-In module.

These metrics can be returned for Oracle WebLogic Server and Apache HTTP Server Plug-ins.

Table 2-1   State Metrics for the Oracle WebLogic Server Proxy Plug-In Module

Metric Name	Description
totalDeclines	The total number of requests declined (not processed by mod_wl_ohs). This number indicates the requests that are not configured, and/or rejected by the plug-in (for example, custom HTTP methods are always rejected by the plug-in)
totalErrors	Number of requests that could not be processed successfully. See Event Metrics for errors.
totalHandled	The total number of requests serviced by the mod_wl_ohs plug-in.
totalRequests	The total number of requests received by mod_wl_ohs. The number includes all the requests that are targeted to the plug-in, plus the requests that are not targeted to any module (not configured).
totalRetries	Number of times a request was retried. Requests are generally retried on failure (depending on configuration). If a request is ever retried, this metric will increment (once per request, irrespective of how many times the request was retried).
totalSuccess	The number of requests successfully processed. If the requests are processed successfully (proxied to Oracle WebLogic Server, and sent the response back to client), then this metric will be incremented.
websocketActive	Number of WebSocket upgrade requests currently active.
websocketClose	Number of WebSocket upgrade requests closed. If the WebSocket session is terminated (for any reason), then this metric is updated.

Table 2-1   (Cont.) State Metrics for the Oracle WebLogic Server Proxy Plug-In Module

Metric Name	Description
websocketMax	Maximum number of simultaneous WebSocket requests that can be active.
	If the WLMaxWebSocketClients parameter is configured, the value will be the lower of these:
	• The configured value, OR
	• 0.75 of the value of MaxRequestWorkers (Oracle HTTP Server and Apache 2.4), OR
	• 0.75 of the value of MaxClients (Apache 2.2), OR
	• 0.75 of the value of ThreadsPerChild (Oracle HTTP Server and Apache 2.2/2.4, Windows only)
	If WLMaxWebSocketClients parameter is not configured, the value will be:
	• 0.5 of the value of MaxRequestWorkers (Oracle HTTP Server and Apache 2.4), OR
	• 0.5 of the value of MaxClients (Apache 2.2), OR
	• 0.5 of the value of ThreadsPerChild (Oracle HTTP Server and Apache 2.2/2.4, Windows only)
	For more information on the WLMaxWebSocketClients parameter, see Tuning Oracle HTTP Server and Apache HTTP Server for High Throughput for WebSocket Upgrade Requests.
websocketPercent	This value is defined by the number of active WebSockets (websocketActive) divided by the maximum number of simultaneous WebSocket requests (websocketMax) multiplied by 100:
	(websocketActive/webocketMax)*100.
websocketRequests	The number of WebSocket upgrade requests made. If the request URI is an WebSocket upgrade request, this metric will be incremented.
websocketSuccess	Number of WebSocket upgrade requests completed successfully. If Oracle WebLogic Server responds to a WebSocket upgrade request with 101 Switching Protocols, then this metric is updated.

2.5.4 DMS Event Metrics

A DMS event metric counts system events. A DMS event tracks system events that have a short duration, or where the duration of the event is not of interest but the occurrence of the event is of interest. Table 2-2 describes the Event metrics available for the Oracle WebLogic Server Proxy Plug-In module.

These metrics can be returned for Oracle WebLogic Server and Apache HTTP Server Proxy Plug-ins.

Table 2-2   Event Metrics for the Oracle WebLogic Server Proxy Plug-In Module.

Metric Name	Description
errConnRefused	The number of CONNECTION_REFUSED errors. Indicates the number of times the configured WebLogicHost and/or WebLogicPort is either not reachable or not listening.
errNoResources	The number of NO_RESOURCES errors. One scenario where this exception can occur is when SSL is configured in the plug-in, but the corresponding SSL configuration is not defined in the managed server.
errOthers	The number of any other errors. For example, POST data size is greater than the value of MaxPostSize.
errReadClient	The number of READ_ERROR_FROM_CLIENT errors. Indicates the number of times that the plug-in could not read from the client (browser).
errReadServer	The number of READ_ERROR_FROM_SERVER errors. Indicates the number of times a read operation could not be successfully performed on Oracle WebLogic Server.
errReadTimeout	The number of READ_TIMEOUT errors. An example is Oracle WebLogic Server not responding within WLIOTimeoutSecs.
errWriteClient	The number of WRITE_ERROR_TO_CLIENT errors. Indicates the number of times that the plug-in could not write to client. This can be seen when the client sends a request but closes the connection before receiving the response.
errWriteWLS	The number of WRITE_ERROR_TO_SERVER errors. Indicates the number of times that the plug-in could not write to Oracle WebLogic Server.
wsClientClose	Number of WebSocket upgrade requests closed by client. If the client sends a WebSocket upgrade request, and client closes the connection, then this metric is updated.
wsErrorClose	Number of WebSocket sessions terminated due to error. If there is any error which causes the WebSocket connection to close, then this metric is updated.
wsNoUpgrade	The number of times the WebSocket upgrade request was rejected. The response to WebSocket upgrade request is not "101 Switching Protocols". This can happen when the upgrade request is sent to Oracle WebLogic Server that does not support WebSockets (Oracle WebLogic Server version 12.1.2 or earlier).
wsServerClose	Number of WebSocket upgrade requests closed by server. If Oracle WebLogic Server initiates a close of WebSocket communication, then this metric is updated. For example, timeout or no communication (by default, 5 minutes) after upgrading the request.

2.5.5 DMS PhaseEvent Metrics

A DMS PhaseEvent metric measures the time spent in a specific section of code that has a beginning and an end. A PhaseEvent tracks time in a method or in a block of code. For each phase event, an "active count", "completed count", "total time", "min

time", "max time", and "average time" value is included. Table 2-3 describes the PhaseEvent metrics available for the Oracle WebLogic Server Proxy Plug-In module.

These metrics can be returned for Oracle WebLogic Server and Apache HTTP Server Proxy Plug-ins.

Table 2-3   PhaseEvent Metrics for the Oracle WebLogic Server Proxy Plug-In Module

Metric Name	Description
websocketPhase	WebSocket communication in progress. The phase (time) between "WebSocket upgrade succeeded" and "WebSocket connection closed"
wlsWait	The phase (time) between "the request sent to Oracle WebLogic Server" and "Waiting for response".

2.6 Deprecated Directives for Oracle HTTP Server

The WebLogic Server plug-in logs for Oracle WebLogic Server Proxy Plug-In are now part of the Web Server error log mechanism. References can be identified with module name as weblogic. For example:

```
[2015-05-14T00:43:27.8355-06:00] [OHS] [TRACE:16] [OH99999] [weblogic] [client_id: ::
1] [host_id: XXXXXXXX] [host_addr: XX.XXX.XXX.XXX] [pid: 1240] [tid: 2424] [user:
sramavan] [ecid: 00iT9hK4DrhFw0zobn063z0BvEE3zsYyk0000JO000000H] [rid: 0]
[VirtualHost: main] ================New Request: [GET /favicon.ico HTTP/1.1]
==================
```

The WLLogFile and Debug directives are deprecated. If the configuration uses these directives, the following note appears in the node manager plug-in log file (ohs_nm.log):

```
<2015-05-14 00:36:25> <INFO> <OHS-0> <[Thu May 14 00:36:25.723286 2015]
[weblogic:warn] [pid 5084:tid 668] The Debug directive is ignored. The web server
log level is used instead.>
```

```
<2015-05-14 00:36:25> <INFO> <OHS-0> <[Thu May 14 00:36:25.724263 2015]
[weblogic:warn] [pid 5084:tid 668] The WLLogFile directive is ignored. The web
server log file is used instead.>
```

To enable plug-in logs:

- If **OraLogMode** is set to *ODL-text,* set **OraLogSeverity** to TRACE:16. The logs appear in the directory **OraLogDir** (instance-name.log). This is the default.

- If **OraLogMode** is set to *apache*, set **LogLevel** to *debug*. The directive **ErrorLog** points to the file where the errors are logged.

See Managing Oracle HTTP Server Logs in *Administering Oracle HTTP Server* guide.

2.7 Troubleshooting Oracle WebLogic Server Proxy Plug-In Implementations

You might encounter some of the following problems when using the Oracle WebLogic Server Proxy Plug-In. Descriptions of how to solve these problems are provided.

- WLS Session Issues
- CONNECTION_REFUSED Errors
- NO_RESOURCES Errors
- Changing the Oracle WebLogic Server Keystore Causes Unexpected Behavior

2.7.1 WLS Session Issues

The Oracle WebLogic Server Proxy Plug-In routes the requests to backend Oracle WebLogic Server or cluster. Oracle WebLogic Server maintains sessions so that subsequent requests from the same client are routed to the same server. However, due to various reasons, if the Oracle WebLogic Server Proxy Plug-In cannot communicate with the Oracle WebLogic Server server, the request is handled in the following ways:

- If the request is routed to a single WebLogic Server instance, the Oracle WebLogic Server Proxy Plug-In continues trying to connect to that same WebLogic Server instance for the maximum number of retries as specified by the ratio of ConnectTimeoutSecs and ConnectRetrySecs. If all attempts fail, an HTTP 503 error message is returned back to the client.

- If the request is routed to WebLogic Cluster, then the current WLS server is marked as bad, and the request is routed to the next available WLS server. If all attempts fail, an HTTP 503 error message is returned back to the client.

In addition to sending a HTTP 503 error message, the following is displayed as a response in the HTTP client:

```
Failure of Web Server bridge:
No backend server available for connection: timed out after xx seconds or idempotent
set to OFF or method not idempotent.
```

2.7.2 CONNECTION_REFUSED Errors

Occasionally, under stress conditions, a few requests might fail with the following error logged in the error log file.

```
weblogic: Trying GET /uri at backend host 'xx.xx.xx.xx/port; got exception
'CONNECTION_REFUSED [os error=xxx, line xxxx of URL.cpp]: apr_socket_connect call
failed with error=xxx, host=xx.xx.xx.xx, port=xxxx'
```

As mentioned in Tips for Reducing CONNECTION_REFUSED Errors, Oracle WebLogic Server might have reached the maximum allowed backlog connections.

To resolve, follow the steps mentioned in Tips for Reducing CONNECTION_REFUSED Errors.

2.7.3 NO_RESOURCES Errors

Occasionally, under stress conditions, a few requests might fail with the following error logged in the error log file.

```
weblogic: *******Exception type [NO_RESOURCES] (apr_socket_connect call failed with
error=70007, host=xx.xx.xx.xx, port=xxxx) raised at line xxxx of URL.cpp
```

This usually occurs if Oracle WebLogic Server is too busy to respond to the connect request from the Oracle WebLogic Server Proxy Plug-In. This can be resolved by

setting `WLSocketTimeoutSecs` to a higher value. This allows the Oracle WebLogic Server Proxy Plug-In to wait longer for the connect request to be responded to by the Oracle WebLogic Server.

2.7.4 Changing the Oracle WebLogic Server Keystore Causes Unexpected Behavior

If the Oracle WebLogic Server keystore is changed (for example, by setting up backend one-way SSL between Oracle HTTP Server and Oracle WebLogic Server) it is possible to break communication between Oracle WebLogic Server and the NodeManager. If this happens, this will also affect all provisioning and process management commands issued through WLST. The error messaging in this situation is poor and the situation can be confusing. For example, some errors indicate NodeManager is down, when the user can see that it is clearly up.

Here are a few examples of the unexpected behavior that can occur. Note that in these examples, the NodeManager is up and running.

- All `state()` commands run against Oracle HTTP Server instances return `UNKNOWN`.

  ```
  wls:/OHSDomain/serverConfig/> state('ohs1')
  Current state of "ohs1" : UNKNOWN
  ```

- Oracle HTTP Server process management commands return an `SSLEngine` failure:

  ```
  wls:/OHSDomain/serverConfig/> start('ohs1')
  Starting system component "ohs1" ...

  General SSLEngine problem
  Traceback (innermost last):
    File "<console>", line 1, in ?
    File "<iostream>", line 1384, in start
    File "<iostream>", line 553, in raiseWLSTException
  WLSTException: Error occurred while performing start : System component with
  name "ohs1" failed to start : General SSLEngine problem
  ....
  ```

- The Oracle HTTP Server custom command `ohs_createInstance` returns a message that NodeManager is down and the command will be completed when it is back up, for example:

  ```
  ohs_createInstance(instanceName='ohs1',machine='myMachine.myCompany.com')
  ....
  The node manager for "ohs1" is not reachable. Changes will be completed when the
  node manager is available.
  The node manager error is: Node Manager is not available on machine
  myMachine.myCompany.com
  Activation completed.
  ```

Workaround:

To workaround this issue, add the NodeManager demo trust certificate to the Java standard trust. Note that this workaround applies only when the user is setting up one-way backend SSL. Other steps may be needed for other scenarios.

1. Use `keytool` command to export the NodeManager demo trust certificate to the Java keystore.

```
keytool -exportcert -rfc -alias wlscertgenca -storepass
DemoTrustKeyStorePassPhrase -file /<path to location of nmCert.crt> -
keystore $ORACLE_HOME/wlserver/server/lib/DemoTrust.jks
```

2. Use `keytool` command to import the NodeManager demo trust certificate to the Java standard trust.

```
keytool -importcert -alias wlscertgenca -file /<path to location of nmCert.crt>
-keystore $JAVA_HOME/jre/lib/security/cacerts -trustcacerts -storepass changeit -
noprompt
```

3. Oracle WebLogic Server may need to be bounced (shutdown and restarted) after applying the `keytool` commands.

3

Configuring the Plug-In for Apache HTTP Server

For proxying requests from Oracle HTTP Server to Oracle WebLogic Server, use the `mod_wl_ohs` plug-in. This plug-in is similar to the plug-in for Apache HTTP Server. However, you do not need to download and install the plug-in separately. For information about configuring `mod_wl_ohs`, see Configuring the Plug-In for Oracle HTTP Server .

WebLogic Server Proxy Plug-in 12.2.1.0 and later version builds are moved from Intel compiler to MSVC Compiler. When Apache HTTP Server is used as a front end with WebLogic Server Proxy Plug-in, the plug-in library depends on the two dlls — `msvcp110.dll` and `msvcr110.dll`, provided by Microsoft. These dlls are available with Microsoft Visual C ++ Redistributable Package for x64.

To install and configure the Oracle WebLogic Server Proxy Plug-In for Apache HTTP Server, read the following sections:

- Apache HTTP Server Support Note
- Install the Oracle WebLogic Server Proxy Plug-In for Apache HTTP Server
- Configure the Apache HTTP Server Plug-In
- Understanding DMS Metrics for Apache HTTP Server Plug-in
- Deprecated Directives for Apache HTTP Server

3.1 Apache HTTP Server Support Note

The Oracle WebLogic Server Proxy Plug-In for Apache HTTP Server is supported on Apache 2.2 and 2.4 web servers and can front-end WebSocket applications.

3.2 Install the Oracle WebLogic Server Proxy Plug-In for Apache HTTP Server

After you download the Oracle WebLogic Server Proxy Plug-In for Apache HTTP Server, as described in Availability of Oracle WebLogic Server Proxy Plug-In, you can install it as an Apache HTTP Server module in your Apache HTTP Server installation.

- Installation Prerequisites
- Installing the Apache HTTP Server Plug-In

3.2.1 Installation Prerequisites

Before you install the Oracle WebLogic Server Proxy Plug-In for Apache HTTP Server, complete these installation and verification tasks.

- Download the Oracle WebLogic Server Proxy Plug-In for Apache HTTP Server, as described in Availability of Oracle WebLogic Server Proxy Plug-In.

- Extract the plug-ins zip distribution to PLUGIN_HOME. For example, `/home/myhome/weblogic-plugins-12.2.1.3.0/`. This is the directory to which the extract the plug-in is extracted.

  This distribution contains these files:

Table 3-1   Files Included in the Apache Web Server Plug-in Zip

(path)/filename	Description
`README.txt`	The README file for the plug-in.
`bin/orapki.bat`	orapki tool for configuring Oracle wallets
`jlib/*.jar`	orapki helper Java libraries
`lib/mod_wl.so`	WebLogic proxy module for Apache 2.2
`lib/*.so`	Helper libraries
`lib/mod_wl_24.so`	WebLogic proxy module for Apache 2.4

- Install JDK 8 to use SSL. The JDK 8 installation is required to use the orapki utility, which manages public key infrastructure (PKI) elements, such as wallets and certificate revocation lists, for use with SSL.

- Ensure that you have a supported Apache HTTP Server installation. See `http://www.oracle.com/technetwork/middleware/ias/downloads/fusion-certification-100350.html` .

- Ensure that a supported version of Oracle WebLogic Server is configured and running on a target system. This server does not need to be running on the system on which you extracted the plug-in zip distribution. For the supported Oracle WebLogic Server versions, see `http://www.oracle.com/technetwork/middleware/ias/downloads/fusion-certification-100350.html` .

3.2.2 Installing the Apache HTTP Server Plug-In

The Oracle WebLogic Server Proxy Plug-In for Apache HTTP Server is distributed as a shared object (.so) file. You can obtain the plug-in here:

`http://www.oracle.com/technetwork/middleware/webtier/overview/index.html`

To install the Apache HTTP Server plug-in:

1. Ensure that the weblogic-plugins-12.2.1.3.0/lib folder is included in LD_LIBRARY_PATH on UNIX systems (and PATH on Windows systems). If you do not do this, then you see linkage errors when starting Apache HTTP Server.

2. In the location where you unzipped the downloaded plug-in file, locate lib/mod_wl.so; for example, /home/myhome/weblogic-plugins-12.2.1.3.0/lib/mod_wl.so.

3. Verify that the mod_so.c module is enabled.

   If you installed Apache HTTP Server using the script supplied by Apache, mod_so.c is already enabled. Verify that mod_so.c is enabled by executing the following command:

   - UNIX/Linux

```
APACHE_HOME/bin/apachectl -l
```

(*APACHE_HOME* is the directory that contains the Apache HTTP Server installation.)

This command lists all enabled modules. If mod_so.c is not listed, you must rebuild your Apache HTTP Server, making sure that the following configure option is specified:

```
...
--enable-module=so
...
```

4. Make a copy of the *APACHE_HOME*/bin/httpd.conf file for backup.

5. Open the httpd.conf file.

6. Depending on your version of Apache, use one of the following commands to install the plug-in:

    •   Install the Apache HTTP Server plug-in module for Apache 2.2.x by adding the following line.

        ```
        LoadModule weblogic_module /home/myhome/weblogic-plugins-12.2.1.3.0/lib/
        mod_wl.so
        ```

    •   Install the Apache HTTP Server plug-in module for Apache 2.4.x by adding the following line.

        ```
        LoadModule weblogic_module /home/myhome/weblogic-plugins-12.2.1.3.0/lib/
        mod_wl_24.so
        ```

7. Verify the syntax of the `httpd.conf` file by running the following command:

    •   UNIX/Linux

        ```
        > APACHE_HOME/bin/apachectl -t
        ```

    If the httpd.conf file contains any errors, the output of this command shows the errors; otherwise, the command returns the following:

    ```
    Syntax OK
    ```

3.3 Configure the Apache HTTP Server Plug-In

This section describes how to edit the httpd.conf file to proxy requests by path or by MIME type, to enable HTTP tunneling, and to use other Oracle WebLogic Server plug-in parameters.

•   Configuring the httpd.conf File

•   Placing WebLogic Properties Inside Location or VirtualHost Blocks

•   Example: Configuring the Apache Plug-In

•   Including a weblogic.conf File in the httpd.conf File

3.3.1 Configuring the httpd.conf File

To configure the Apache HTTP Server plug-in, edit the `httpd.conf` file in your Apache HTTP Server installation. Complete the following tasks:

•   Task 1: Configure MIME Requests

3.3.1.1 Task 1: Configure MIME Requests

You can proxy requests by MIME type and/or by path. Open the httpd.conf file in a text editor.

- • Configuring Proxy Requests by MIME Type
- • Configuring Proxy Requests by Path

> **Note:**
>
> If both MIME type and proxying by path are enabled, proxying by path takes precedence over proxying by MIME type.

3.3.1.1.1 Configuring Proxy Requests by MIME Type

Follow these steps to configure MIME requests by MIME type in the httpd.conf file.

1. Add an `<IfModule>` block that defines one of the following:

   - • For a non-clustered WebLogic Server: define the WebLogicHost and WebLogicPort parameters.

   - • For a cluster of WebLogic Servers: define the WebLogicCluster parameter.

   Example:

   ```
   <IfModule mod_weblogic.c>
    WebLogicHost myweblogic.example.com
    WebLogicPort 7001
    DebugConfigInfo ON
   </IfModule>
   ```

2. Add a MatchExpression line to the `<IfModule>` block.

   For example, the following `<IfModule>` block for a non-clustered WebLogic Server specifies that all files with MIME type .jsp are proxied:

   ```
   <IfModule mod_weblogic.c>
    WebLogicHost my-weblogic.server.com
    WebLogicPort 7001
    MatchExpression *.jsp
    DebugConfigInfo ON
   </IfModule>
   ```

   You can also use multiple MatchExpressions, for example:

   ```
   <IfModule mod_weblogic.c>
    WebLogicHost my-weblogic.server.com
    WebLogicPort 7001
    MatchExpression *.jsp
    MatchExpression *.xyz
   ```

```
 DebugConfigInfo ON
</IfModule>
```

If you are proxying requests by MIME type to a cluster of WebLogic Servers, use the `WebLogicCluster` parameter instead of the `WebLogicHost` and `WebLogicPort` parameters. For example:

```
<IfModule mod_weblogic.c>
 WebLogicCluster w1s1.com:7001,w1s2.com:7001,w1s3.com:7001
 MatchExpression *.jsp
 MatchExpression *.xyz
</IfModule>
```

3.3.1.1.2 Configuring Proxy Requests by Path

Follow these steps to configure MIME requests by path in the httpd.conf file.

1. Use the `<Location>` block and the `WLSRequest` statement to configure MIME requests by path. `WLSRequest` specifies the handler for the Oracle WebLogic Server Proxy Plug-In for Apache HTTP Server module. For example the following `Location` block proxies all requests containing **/weblogic** in the URL:

```
<Location /weblogic>
WLSRequest On
PathTrim /weblogic
</Location>
```

2. Configure the PathTrim parameter inside the `<Location>` tag.

The PathTrim parameter specifies a string trimmed from the beginning of the URL before the request is passed to the WebLogic Server instance (see General Parameters for Web Server Plug-Ins).

These known issues arise when you configure the Oracle WebLogic Server Proxy Plug-In for Apache HTTP Server to use SSL

- The following configuration is **incorrect**:

```
<Location /weblogic>
 WLSRequest On
</Location>

<IfModule mod_weblogic.c>
 WebLogicHost localhost
 WebLogicPort 7001
 PathTrim /weblogic
</IfModule>
```

The following configuration is **correct**:

```
<Location /weblogic>
 WLSRequest On
 PathTrim /weblogic
</Location>
```

- The current implementation of the Oracle WebLogic Server Proxy Plug-In for Apache HTTP Server does not support the use of multiple certificate files with Apache SSL.

3.3.1.2 Task 2: Define Additional Parameters for Oracle WebLogic Server Proxy Plug-In

Define any additional parameters for the Oracle WebLogic Server Proxy Plug-In for Apache HTTP Server.

The Oracle WebLogic Server Proxy Plug-In for Apache HTTP Server recognizes the parameters listed in General Parameters for Web Server Plug-Ins. To modify the behavior of your Oracle WebLogic Server Proxy Plug-In for Apache HTTP Server, define these parameters either:

- In a `<Location>` block, for parameters that apply to proxying by path, or

- At global or virtual host scope, for parameters that apply to proxying by MIME type.

3.3.1.3 Task 3: Enable HTTP Tunneling (Optional)

You can enable HTTP tunneling for the t3 or IIOP protocols by configuring `<Location>` blocks.

- To enable HTTP tunneling if you are using the t3 protocol and weblogic.jar, add the following `<Location>` block to the httpd.conf file:

```
<Location /bea_wls_internal>
 WLSRequest On
</Location>
```

- To enable HTTP tunneling if you are using the IIOP, the only protocol used by the WebLogic Server thin client, wlclient.jar, add the following `Location` block to the httpd.conf file:

```
<Location /bea_wls_internal>
 WLSRequest On
</Location>
```

3.3.1.4 Task 4: Enable Web Services Atomic Transaction (Optional)

You can enable Web Services Atomic Transaction (WS-AT) by configuring the `<Location>` blocks. The <wls-wsat> parameter applies to proxying by path. You can optionally define the parameter to modify the behavior of your WebLogic Web Server Proxy Plug-In for Apache HTTP Server.

```
<Location /wls-wsat>
 WLSRequest On
</Location>
```

WebLogic web services enable interoperability with other external transaction processing systems, such as IBM WebSphere, JBoss, Microsoft .NET. For more information about Web Services Atomic Transaction (WS-AtomicTransaction), see https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=ws-tx.

3.3.1.5 Task 5: Verify and Apply Your Configuration

Follow these steps to verify your httpd.conf configuration and apply it to the Apache HTTP Server.

1. Verify the syntax of the httpd.conf file by running the following command:

   - UNIX/Linux

     > *APACHE_HOME*/bin/apachectl -t

   If the httpd.conf file contains any errors, the output of this command shows the errors; otherwise, the command returns the following:

   ```
   Syntax OK
   ```

2. Start the Apache HTTP Server.

   - UNIX/Linux

     > *APACHE_HOME*/bin/apachectl start

3. Send a request to `http://apache-host:apache-port/mywebapp/my.jsp` from the browser. Validate the response.

3.3.2 Placing WebLogic Properties Inside Location or VirtualHost Blocks

If you choose to not use the `<IfModule>`, you can instead directly place the WebLogic properties inside `Location` or `<VirtualHost>` blocks. Consider the following examples of the `<Location>` and `<VirtualHost>` blocks:

```
<Location /weblogic>
WLSRequest On
WebLogicHost myweblogic.server.com
WebLogicPort 7001
</Location>

<Location /weblogic>
WLSRequest On
WebLogicCluster w1s1.com:7001,w1s2.com:7001,w1s3.com:7001
</Location>

<VirtualHost apachehost:80>
WLSRequest On
WebLogicServer weblogic.server.com
WebLogicPort 7001
</VirtualHost>
```

3.3.3 Example: Configuring the Apache Plug-In

This example demonstrates basic instructions for quickly setting up the Apache plug-in to proxy requests to a backend WebLogic Server.

1. Make a copy of ${*APACHE_HOME*}/conf/httpd.conf file.

2. Edit the file to add the following code:

   ```
   ...
   LoadModule weblogic_module /home/myhome/weblogic-plugins-12.2.1/lib/mod_wl.so

   <IfModule mod_weblogic.c>
     WebLogicHost wls-host
     WebLogicPort wls-port
   </IfModule>
   ```

```
<Location /mywebapp>
  WLSRequest On
</Location>
...
```

3. Include ${*PLUGIN_HOME*}/lib in the *LD_LIBRARY_PATH* by entering the following command:

```
$ export LD_LIBRARY_PATH=/home/myhome/weblogic-plugin-12.2.1/lib:...
```

> **✎ Note:**
>
> You can also update the `LD_LIBRARY_PATH` by copying the 'lib' contents to *APACHE_HOME*/lib or by editing the *APACHE_HOME*/bin/apachectl to update the `LD_LIBRARY_PATH`.

4. At the prompt, start the Apache HTTP Server by entering:

```
$ ${APACHE_HOME}/bin/apachectl start
```

5. Send a request to `http://apache-host:apache-port/mywebapp/my.jsp` from the browser and validate the response

3.3.4 Including a weblogic.conf File in the httpd.conf File

To keep several separate configuration files, you can define parameters in a separate configuration file called `weblogic.conf` file, by using the Apache HTTP Server `Include` directive in an `<IfModule>` block in the httpd.conf file.

```
<IfModule mod_weblogic.c>
 # Config file for WebLogic Server that defines the parameters
 Include conf/weblogic.conf
</IfModule>
```

The syntax of weblogic.conf files is the same as that for the httpd.conf file.

The following sections describe how to create weblogic.conf files, and include sample weblogic.conf files.

- Rules for Creating weblogic.conf Files
- Sample weblogic.conf Configuration Files
- Template for the Apache HTTP Server httpd.conf File

3.3.4.1 Rules for Creating weblogic.conf Files

Be aware of the following rules and best practices for constructing a weblogic.conf file.

- Enter each parameter on a new line. Do not put "=" between a parameter and its value. For example:

```
PARAM_1 value1
PARAM_2 value2
PARAM_3 value3
```

- If a request matches both a MIME type specified in a `MatchExpression` in an `<IfModule>` block and a path specified in a `Location` block, the behavior specified by the `<Location>` block takes precedence.

- If you use an Apache HTTP Server `<VirtualHost>` block, you must include all configuration parameters (`MatchExpression`, for example) for the virtual host within the `<VirtualHost>` block (see Apache Virtual Host documentation at `http://httpd.apache.org/docs/vhosts/`).

- Sample httpd.conf file:

```
<IfModule mod_weblogic.c>
 WebLogicCluster johndoe02:8005,johndoe:8006
 WLTempDir "c:\myTemp"
 DebugConfigInfo ON
 KeepAliveEnabled ON
 KeepAliveSecs 15
</IfModule>

<Location /jurl>
 WLSRequest On
 WebLogicCluster myCluster:7001
 WLTempDir "c:\jurl"
</Location>

<Location /web>
 WLSRequest On
 PathTrim /web
 WebLogicHost myhost
 WebLogicPort 8001
 WLTempDir "c:\web"
</Location>
 <Location /foo>
 WLSRequest On
 WebLogicHost myhost02
 WebLogicPort 8090
 WLTempDir "c:\foo"
 PathTrim /foo
</Location>
```

- All the requests that match /jurl/* will have the POST data files in c:\jurl and will reverse proxy the request to agarwalp01 and port 7001. All the requests that match /web/* will have the POST data files in c:\web and will reverse proxy the request to myhost and port 8001. All the requests that match /foo/* will have the POST data files written to c:\foo and will reverse proxy the request to myhost02 and port 8090.

- You should use the `MatchExpression` statement instead of the `<Files>` block.

3.3.4.2 Sample weblogic.conf Configuration Files

These examples of weblogic.conf files may be used as templates that you can modify to suit your environment and server. Lines beginning with # are comments.

Example Using WebLogic Clusters

```
# These parameters are common for all URLs which are
# directed to the current module. If you want to override
# these parameters for each URL, you can set them again in
# the <Location> or <Files> blocks.
<IfModule mod_weblogic.c>
```

```
 WebLogicCluster w1s1.com:7001,w1s2.com:7001,w1s3.com:7001
 ErrorPage http://myerrorpage.mydomain.com
 MatchExpression *.jsp
</IfModule>
####################################################
```

In the example, the `MatchExpression` parameter syntax for expressing the filename pattern, the WebLogic Server host to which HTTP requests should be forwarded, and various other parameters is as follows:

```
MatchExpression [filename pattern] [WebLogicHost=host] | [paramName=value]
```

The first MatchExpression parameter below specifies the filename pattern *.jsp, and then names the single `WebLogicHost`. The `paramName=value` combinations following the pipe symbol specify the port at which WebLogic Server is listening for connection requests, and also activate the Debug option. The second `MatchExpression` specifies the filename pattern `*.html` and identifies the WebLogic Cluster hosts and their ports. The `paramName=value` combination following the pipe symbol specifies the error page for the cluster.

Example Using Multiple WebLogic Clusters

```
# These parameters are common for all URLs which are
# directed to the current module. If you want to override
# these parameters for each URL, you can set them again in
# the <Location> or <Files> blocks.
<IfModule mod_weblogic.c>
 MatchExpression *.jsp WebLogicHost=myHost|WebLogicPort=7001|Debug=ON
 MatchExpression *.html WebLogicCluster=myHost1:7282,myHost2:7283|ErrorPage=
 http://www.xyz.com/error.html
</IfModule>
```

Example Without WebLogic Clusters

```
# These parameters are common for all URLs which are
# directed to the current module. If you want to override
# these parameters for each URL, you can set them again in
# the <Location> or <Files> blocks.
<IfModule mod_weblogic.c>
 WebLogicHost myweblogic.server.com
 WebLogicPort 7001
 MatchExpression *.jsp
</IfModule>
```

Example Configuring Multiple Name-Based Virtual Hosts

```
# VirtualHost1 = localhost:80
<VirtualHost 127.0.0.1:80>
DocumentRoot "C:/test/VirtualHost1"
ServerName localhost:80
<IfModule mod_weblogic.c>
#... WLS parameter ...
WebLogicCluster localhost:7101,localhost:7201
# Example: MatchExpression *.jsp <some additional parameter>
MatchExpression *.jsp PathPrepend=/test2
</IfModule>
</VirtualHost>

# VirtualHost2 = 127.0.0.2:80
<VirtualHost 127.0.0.2:80>
DocumentRoot "C:/test/VirtualHost1"
```

```
ServerName 127.0.0.2:80
<IfModule mod_weblogic.c>
#... WLS parameter ...
WebLogicCluster localhost:7101,localhost:7201
# Example: MatchExpression *.jsp <some additional parameter>
MatchExpression *.jsp PathPrepend=/test2
#... WLS parameter ...
</IfModule>
</VirtualHost>
```

You must define a unique value for `ServerName` or some plug-in parameters will not work as expected.

3.3.4.3 Template for the Apache HTTP Server httpd.conf File

This section contains a sample httpd.conf file for Apache HTTP Server. You can use this sample as a template and modify it to suit your environment and server. Lines beginning with # are comments.

> ✎ **Note:**
>
> Apache HTTP Server is not case sensitive.

Sample httpd.conf file for Apache HTTP Server

```
####################################################
# APACHE-HOME/conf/httpd.conf file
####################################################
LoadModule weblogic_module /home/myhome/weblogic-plugins-12.2.1/lib/mod_wl.so

<Location /weblogic>
 WLSRequest On
 PathTrim /weblogic
 ErrorPage http://myerrorpage1.mydomain.com
</Location>

<Location /servletimages>
 WLSRequest On
 PathTrim /something
 ErrorPage http://myerrorpage1.mydomain.com
</Location>

<IfModule mod_weblogic.c>
  MatchExpression *.jsp
  WebLogicCluster wls1.com:7001,wls2.com:7001,wls3.com:7001
  ErrorPage http://myerrorpage.mydomain.com
</IfModule>
```

3.4 Understanding DMS Metrics for Apache HTTP Server Plug-in

You can configure and view DMS performance metrics for Apache HTTP Server. The Apache HTTP server provides the same set of DMS metrics as the Oracle WebLogic Server.

The DMS metrics that can be returned are described in DMS State Metrics, DMS Event Metrics, and DMS PhaseEvent Metrics.

This section contains the following information:

- Configuring Metrics DMS Metrics for Apache HTTP Server Plug-in
- Viewing Performance Metrics for Apache HTTP Server Plug-in

3.4.1 Configuring Metrics DMS Metrics for Apache HTTP Server Plug-in

To configure DMS metrics for the Apache HTTP Server, add the following code to the `httpd.conf` file.

```
# Add the following LoadModule only if it is not already present
# Use mod_wl.so for Apache httpd 2.2
LoadModule weblogic_module $PLUGIN_HOME/mod_wl_24.so

<Location /metrics>
    SetHandler dms-handler
</Location>
```

3.4.2 Viewing Performance Metrics for Apache HTTP Server Plug-in

You can view the raw metrics using the URL:

```
http://apachehost:apacheport/metrics
```

where `apachehost` is the host name of the Apache server and `apacheport` is the port number.

The metrics that are coming from Oracle WebLogic Server plug-in can be found under the section `/WebLogicProxy [type=OHSWebLogic]`.

3.5 Deprecated Directives for Apache HTTP Server

The WebLogic Server plug-in logs are now part of the Apache HTTP Server error log. References can be identified with the prefix `weblogic:` to easily identify them.

Apache 2.4 example:

```
[Thu May 14 23:15:05.160459 2015] [weblogic:debug] [pid 6571:tid 139894556022528]
ApacheProxy.cpp(875): [client 10.184.61.77:53634] <657114316705052> =========New
Request: [GET /weblogic/index.html HTTP/1.1] ======
```

Apache 2.2 example:

```
[Thu Apr 16 04:15:37 2015] [debug] ApacheProxy.cpp(873): [client 10.184.61.77]
<2157714291829372> weblogic: ======New Request: [GET /weblogic/index.html HTTP/1.1]
======
```

The directives WLLogFile and Debug are deprecated. If the configuration uses these directives, the following note appears during startup:

```
[Thu May 14 23:22:19 2015] [warn] weblogic: The Debug directive is ignored.  The web
server log level is used instead.
```

To enable plug-in logs, set *LogLevel* to `debug`. The logs will be included in the file pointed to by the *ErrorLog* directive.

4

Configuring the Plug-In for iPlanet Web Server

This chapter describes how to install and configure the Oracle WebLogic Server Proxy Plug-In for iPlanet Web Server. In previous releases, this plug-in was referred to as the Netscape Enterprise Server plug-in.

This chapter contains the following sections:

- Overview of the Oracle WebLogic Server Proxy Plug-In for iPlanet
- Installing and Configuring the Plug-In for iPlanet
- Deprecated Directives for iPlanet Web Server

4.1 Overview of the Oracle WebLogic Server Proxy Plug-In for iPlanet

The Oracle WebLogic Server Proxy Plug-In for iPlanet Web Server enables requests to be proxied from Oracle iPlanet Web Server to Oracle WebLogic Server. The plug-in enhances an Oracle iPlanet Web Server installation by allowing Oracle WebLogic Server to handle those requests that require the dynamic functionality of the server.

The Oracle WebLogic Server Proxy Plug-In for iPlanet Web Server is designed for an environment where Oracle iPlanet Web Server serves static pages, and an Oracle WebLogic Server instance (operating in a different process, possibly on a different machine) is delegated to serve dynamic pages, such as JSPs or pages generated by HTTP Servlets. The connection between WebLogic Server and the Oracle WebLogic Server Proxy Plug-In for iPlanet Web Server is made using clear text or Secure Sockets Layer (SSL). To the end user—the browser—the HTTP requests delegated to WebLogic Server appear to come from the same source as the static pages. Additionally, the HTTP-tunneling facility of WebLogic Server can operate through the Oracle WebLogic Server Proxy Plug-In for iPlanet Web Server, providing access to all WebLogic Server services (not just dynamic pages).

The Oracle WebLogic Server Proxy Plug-In for iPlanet Web Server operates as a module within an Oracle iPlanet Web Server. The module is loaded at startup and later based on the configuration, certain HTTP requests are delegated to it.

For more information about Oracle iPlanet Web Server see, `http://docs.oracle.com/docs/cd/E18958_01/doc.70/e18789/chapter.htm`

4.2 Installing and Configuring the Plug-In for iPlanet

The following sections provide information pertaining to the installation prerequisites and configuring the Oracle WebLogic Server Proxy Plug-In for iPlanet Web Server.

- Installation Prerequisites

4.2.1 Installation Prerequisites

Before you install the Oracle WebLogic Server Proxy Plug-In for iPlanet Web Server, complete these installation and verification tasks.

- • Create a plug-in zip extract location (*PLUGIN_HOME*. For example, `/home/myhome/weblogic-plugins-12.2.1.3.0/`)

- • Download the Oracle WebLogic Server Proxy Plug-In for iPlanet Web Server, as described in Availability of Oracle WebLogic Server Proxy Plug-In.

- • Extract the plug-in zip distribution into the Web Server installation directory, *install-dir*. Before extracting the plug-in zip distribution, rename the existing README.txt within *install-dir*. This distribution contains these files:

Table 4-1   Files Included in the Oracle iPlanet Web Server Plug-in Zip

(path)/filename	Description
README.txt	information specific to the distribution, late-breaking updates, and other errata.
bin/orapki (.bat on Windows)	orapki tool for configuring Oracle wallets
jlib/*.jar	orapki helper Java libraries
lib/mod_wl.so	WebLogic proxy module
lib/*.so	Helper libraries

- • Install JDK 8 to use SSL. You must have a JDK 8 installation to use the orapki utility. The orapki utility manages public key infrastructure (PKI) elements, such as wallets and certificate revocation lists, for use with SSL.

- • Create a supported Oracle WebLogic Server Proxy Plug-In for iPlanet Web Server installation (7.0.9 or later) installed on *IPLANET_HOME*, that is, iPlanet server listening on `iplanet-host:iplanet-port`.

- • Create an iPlanet instance location (*INSTANCE-DIR*. For example, `${IPLANET_HOME}/https-foo`.

- • Create a supported version of WebLogic Server is configured and running on a target system. This server does not need to run on the system to which you extracted the plug-in zip distribution. For the supported WebLogic Server versions, see `http://www.oracle.com/technology/software/products/ias/files/fusion_certification.html`.

4.2.2 Installing the Oracle WebLogic Server Proxy Plug-In for iPlanet Web Server

The Oracle WebLogic Server Proxy Plug-In for iPlanet Web Server is distributed as a shared object (.so).

To instruct Oracle iPlanet Web Server to load the native library (mod_wl.so) as a module, add the following line to the magnus.conf file.

```
Init fn="load-modules" shlib="mod_wl.so"
```

The magnus.conf file is located in the *INSTANCE-DIR*/config directory. Where *INSTANCE-DIR* is the web server instance directory.

See http://docs.oracle.com/docs/cd/E19146-01/821-1827/821-1827.pdf

4.2.3 Configuring the Oracle WebLogic Server Proxy Plug-In for iPlanet Web Server

This section provides information about configuring the Oracle WebLogic Server Proxy Plug-In for iPlanet Web Server. To do this, you configure the obj.conf file. the obj.conf file defines which requests are proxied to WebLogic Server and other configuration information.

1. Locate and open the obj.conf file

   The default obj.conf file is located in the *INSTANCE-DIR*/config directory. Where *INSTANCE-DIR* is the web server instance directory.

   See http://docs.oracle.com/docs/cd/E19146-01/821-1827/821-1827.pdf

2. Configure the obj.conf file.

   There are different ways to configure obj.conf file. See the guidelines described in Guidelines for Modifying the obj.conf File.

4.2.3.1 Proxying Requests by URL

To proxy requests by URL, (also called proxying by path.) create a separate `<Object>` tag for each URL that you want to proxy and define the `PathTrim` parameter. The following is an example of an `<Object>` tag that proxies a request containing the string `*/weblogic/*`

```
<Object ppath="*/weblogic/*">
Service fn=wl-proxy WebLogicHost=myserver.com WebLogicPort=7001 PathTrim="/weblogic"
</Object>
```

Here is an example of the object definitions for two separate `ppaths` that identify requests to be sent to different instances of WebLogic Server:

```
<Object ppath="*/weblogic/*">
Service fn=wl-proxy WebLogicHost=myserver.com WebLogicPort=7001 PathTrim="/weblogic"
</Object>
<Object name="si" ppath="*/servletimages/*">
Service fn=wl-proxy WebLogicHost=otherserver.com WebLogicPort=7008
</Object>
```

> **✎ Note:**
>
> Parameters that are not required, such as `PathTrim`, can be used to further configure the way the `ppath` is passed through the Oracle WebLogic Server Proxy Plug-In for iPlanet Web Server. For a complete list of plug-in parameters, see General Parameters for Web Server Plug-Ins.

4.2.3.2 Proxying the Request by MIME Type

If you are proxying requests by MIME type, add any new MIME types referenced in the obj.conf file to the mime.types file. You can add MIME types by using the iPlanet server console or by editing the mime.types file directly.

- To directly edit mime.types file, open the file for editing and type the following line:

  ```
  type=text/jsp exts=jsp
  ```

- To edit the mime.types file in the iPlanet Administration console, see `http://docs.oracle.com/docs/cd/E19146-01/821-1828/gdabr/index.html`.

> **✎ Note:**
>
> iPlanet Web Server 7.0.9 and above already defines the MIME type for JSPs. Change the existing MIME type from magnus-internal/jsp to text/jsp.

All requests with a designated MIME type extension (for example, .jsp) can be proxied to the WebLogic Server, regardless of the URL.

For example, to proxy all JSPs to a WebLogic Server, the following Service directive should be added:

```
Service method="(GET|HEAD|POST|PUT)" type=text/jsp fn=wl-proxy
WebLogicHost=myserver.com WebLogicPort=7001 PathPrepend=/jspfiles
```

This Service directive proxies all files with the .jsp extension to the designated WebLogic Server, where they are served with a URL like this:

```
http://myserver.com:7001/jspfiles/myfile.jsp
```

The value of the `PathPrepend` parameter should correspond to the context root of a Web Application that is deployed on the WebLogic Server or cluster to which requests are proxied.

After adding entries for the Oracle WebLogic Server Proxy Plug-In for iPlanet Web Server, the default `<Object>` definition will be similar to the following example:

```
<Object name="default">
AuthTrans fn="match-browser" browser="*MSIE*" ssl-unclean-shutdown="true"
NameTrans fn="pfx2dir" from="/mc-icons" dir="/export/home/ws/lib/icons" name="es-
internal"
PathCheck fn="uri-clean"
PathCheck fn="check-acl" acl="default"
PathCheck fn="find-pathinfo"
PathCheck fn="find-index" index-names="index.html,home.html"
```

```
ObjectType fn="type-by-extension"
ObjectType fn="force-type" type="text/plain"
Service method="(GET|HEAD|POST|PUT)" type="text/jsp" fn="wl-proxy"
WebLogicHost="myweblogic.server.com" WebLogicPort="7100"
Service method="(GET|HEAD)" type="magnus-internal/directory" fn="index-common"
Service method="(GET|HEAD|POST)" type="*~magnus-internal/*" fn="send-file"
Service method="TRACE" fn="service-trace"
AddLog fn="flex-log"
</Object>
```

You can add a similar Service statement to the default object definition for all other MIME types that you want to proxy to WebLogic Server.

For proxy-by-MIME to work properly you must disable Java from the Oracle WebLogic Server Proxy Plug-In for iPlanet Web Server. Otherwise, SUN One will try to serve all requests that end in *.jsp and will return a 404 error as it will fail to locate the resource under `$doc_root`.

To disable Java from the Oracle iPlanet Web Server, comment out the `NameTrans` line that appears under `name="default"` in the obj.conf file. Then, restart the web server.

```
#NameTrans fn="ntrans-j2ee" name="j2ee"
```

As an alternative, you can do the following:

- If you are proxying by path, enable HTTP-tunneling.

  If you are using weblogic.jar and tunneling the t3 protocol, add the following object definition to the obj.conf file, substituting the WebLogic Server host name and the WebLogic Server port number, or the name of a WebLogic Cluster that you want to handle HTTP tunneling requests.

  ```
  <Object name="tunnel" ppath="*/HTTPClnt*"
  Service fn=wl-proxy WebLogicHost=myserver.com WebLogicPort=7001
  </Object>
  ```

- If you are tunneling IIOP, which is the only protocol used by the WebLogic Server thin client, wlclient.jar, add the following object definition to the obj.conf file, substituting the WebLogic Server host name and the WebLogic Server port number, or the name of a WebLogic Cluster that you want to handle HTTP tunneling requests.

  ```
  <Object name="tunnel" ppath="*/iiop*">
  Service fn=wl-proxy WebLogicHost=myserver.com WebLogicPort=7001
  </Object>
  ```

4.2.3.3 Testing the Plug-In

To test the Oracle iPlanet Web Server plug-in, follow these steps:

1. Start WebLogic Server.

2. Start Oracle iPlanet Web Server.

   If Oracle WebLogic Server Proxy Plug-In for iPlanet Web Server is already running, you must either restart or reconfigure the server.

3. Test the Oracle WebLogic Server Proxy Plug-In for iPlanet Web Server.

   You can test the Oracle WebLogic Server Proxy Plug-In for iPlanet Web Server using the following URL. It starts the default WebLogic Server HTML page,

welcome file, or default servlet, as defined for the default Web Application as shown in this example:

```
http://webserver_host:webserver_port/weblogic/
```

For information about how to create a default Web Application, see *Developing Web Applications, Servlets, and JSPs for Oracle WebLogic Server*.

4.2.4 Example: Configuring the iPlanet Plug-in

The following example demonstrates basic instructions for quickly setting up the Oracle WebLogic Server Proxy Plug-In for iPlanet Web Server to proxy requests to a backend WebLogic Server (WLS).

1. Edit *$IPLANET_INSTANCE_HOME* /config/magnus.conf file and add the following:

    ```
    ...
    Init fn="load-modules" shlib="$PLUGIN_HOME/lib/mod_wl.so"
    ...
    ```

2. Open the *$IPLANET_INSTANCE_HOME* /config/*<vs-obj.conf>* file (the default is *$IPLANET_INSTANCE_HOME* /config/obj.conf) and add the following code:

    ```
    ...
    <Object name="weblogic" ppath="*/wls/*">
    Service fn="wl-proxy" WebLogicHost=<wls-host> WebLogicPort=<wls-port>
    DebugConfigInfo="ON"
    PathTrim="/wls"
    </Object>
    ...
    ```

    For more information on configuring the contents of obj.conf, see Sample obj.conf File (Not Using a WebLogic Cluster) and Sample obj.conf File (Using a WebLogic Cluster).

3. At the prompt, include the $*PLUGIN_HOME*/lib in the LD_LIBRARY_PATH by entering:

    ```
    set LD_LIBRARY_PATH=/home/user/weblogic-plugin-12.2.1.3.0/lib:..
    ```

    > **✎ Note:**
    >
    > You can also update the LD_LIBRARY_PATH by copying the 'lib' contents to IPLANET_HOME/lib or editing the $IPLANET_INSTANCE_HOME /bin/startserv.

4. At the prompt, start the iPlanet server by entering:

    ```
    $IPLANET_INSTANCE_HOME /bin/startserv
    ```

5. Send a request to http://iplanet-host:iplanet-port/mywebapp/my.jsp from the browser and validate the response.

4.2.5 Guidelines for Modifying the obj.conf File

To use the Oracle WebLogic Server Proxy Plug-In for iPlanet Web Server, you must make several modifications to the obj.conf file.

See http://docs.oracle.com/docs/cd/E19146-01/821-1827/821-1827.pdf.

4.2.6 Sample obj.conf File (Not Using a WebLogic Cluster)

The following code is an example of lines that should be added to the `obj.conf` file if you are not using a cluster. You can use this example as a template that you can modify to suit your environment and server. Lines beginning with # are comments.

- Proxy requests by URL

```
## --------------BEGIN SAMPLE obj.conf CONFIGURATION---------
# (no cluster)
# Configure which types of HTTP requests should be handled by the
# iPlanet NSAPI plug-In (and, in turn, by WebLogic). This is done
# with one or more "<Object>" tags as shown below.
# Here we configure the iPlanet plug-In module to pass requests for
# "/weblogic" to a WebLogic Server listening at port 7001 on
# the host myweblogic.server.com.
<Object ppath="*/weblogic/*">
Service fn=wl-proxy WebLogicHost=myweblogic.server.com WebLogicPort=7001
PathTrim="/weblogic"
</Object>
# Here we configure the plug-in so that requests that
# match "/servletimages/" is handled by the
# plug-in/WebLogic.
<Object name="si" ppath="*/servletimages/*">
Service fn=wl-proxy WebLogicHost=myweblogic.server.com WebLogicPort=7001
</Object>
## -------------END SAMPLE obj.conf CONFIGURATION-------------------
```

- Proxy requests by MIME type

```
This Object directive works by file extension rather than
# request path. To use this configuration, you must modify the existing line or
add the following line to mime.types file.

## -----------BEGIN SAMPLE mime.types CONFIGURATION---------------------
#
# type=text/jsp exts=jsp
## ------------END SAMPLE mime.types CONFIGURATION---------------------

## -------------BEGIN SAMPLE obj.conf CONFIGURATION--------------------
# This configuration means that any file with the extension
# ".jsp" are proxied to WebLogic. Then you must add the
# Service line for this extension to the Object "default",
# which should already exist in your obj.conf file:
<Object name=default>
NameTrans fn=pfx2dir from=/ns-icons dir="c:/Export/Home/ns-icons"
NameTrans fn=pfx2dir from=/mc-icons dir="c://Export/Home/ns-icons"
NameTrans fn="pfx2dir" from="/help" dir="c:/Export/Home/manual/https/ug"
NameTrans fn=document-root root="c:/Export/Home/docs"
Service method="(GET|HEAD|POST|PUT)" type=text/jsp fn=wl_proxy
WebLogicHost=myweblogic.server.com WebLogicPort=7001 PathPrepend=/jspfiles
PathCheck fn=nt-uri-clean
PathCheck fn="check-acl" acl="default"
PathCheck fn=find-pathinfo
PathCheck fn=find-index index-names="index.html,home.html"
ObjectType fn=type-by-extension
ObjectType fn=force-type type=text/plain
Service method=(GET|HEAD) type=magnus-internal/imagemap fn=imagemap
Service method=(GET|HEAD) type=magnus-internal/directory fn=index-common
Service method=(GET|HEAD) type=*~magnus-internal/* fn=send-file
```

```
AddLog fn=flex-log name="access"
</Object>
# The following directive enables HTTP-tunneling of the
# WebLogic protocol through the iPlanet plug-in.
<Object name="tunnel" ppath="*/HTTPClnt*">
Service fn=wl-proxy WebLogicHost=myweblogic.server.com WebLogicPort=7001
</Object>
#
## -------------END SAMPLE obj.conf CONFIGURATION---------------------
```

4.2.7 Sample obj.conf File (Using a WebLogic Cluster)

The following code is an example of lines that should be added to obj.conf if you are using a WebLogic Server cluster. You can use this example as a template that you can modify to suit your environment and server. Lines beginning with # are comments.

- Proxy requests by URL

```
## -------------BEGIN SAMPLE obj.conf CONFIGURATION-------------------
# (using a WebLogic Cluster)
#
# Configure which types of HTTP requests should be handled by the
# iPlanet module (and, in turn, by WebLogic). This is done
# with one or more "<Object>" tags as shown below.
# Here we configure the iPlanet module to pass requests for
# "/weblogic" to a cluster of WebLogic Servers.
<Object ppath="*/weblogic/*">
Service fn=wl-proxy WebLogicCluster="myweblogic.com:7001,yourweblogic.com:7001,
theirweblogic.com:7001" PathTrim="/weblogic"
</Object>
# Here we configure the plug-in so that requests that
# match "/servletimages/" are handled by the
# plug-in/WebLogic.
<Object name="si" ppath="*/servletimages/*">
Service fn=wl-proxy WebLogicCluster="myweblogic.com:7001,yourweblogic.com:7001,
theirweblogic.com:7001"
</Object>
## ----------------END OF SAMPLE obj.conf CONFIGURATION------------------
```

- Proxy requests by MIME types

```
# This Object directive works by file extension rather than
# request path. To use this configuration, you must modify the existing line or
add the following line to mime.types file.:
## ----------------BEGIN SAMPLE mime.types FILE ------------------------
# type=text/jsp exts=jsp
#
## -------------------END SAMPLE mime.types-----------------------------

## -------------BEGIN SAMPLE obj.conf CONFIGURATION---------------------
# This configuration means that any file with the extension
# ".jsp" is proxied to WebLogic. Then you must add the
# Service line for this extension to the Object "default",
# which should already exist in your obj.conf file:
<Object name=default>
NameTrans fn=pfx2dir from=/ns-icons dir="c:/Export/Home/ns-icons"
NameTrans fn=pfx2dir from=/mc-icons dir="c:/Export/Home/ns-icons"
NameTrans fn="pfx2dir" from="/help" dir="c://Export/Home/manual/https/ug"
NameTrans fn=document-root root="c://Export/Home/docs"
Service method="(GET|HEAD|POST|PUT)" type=text/jsp fn=wl_proxy
WebLogicCluster="myweblogic.com:7001, yourweblogic.com:7001, theirweblogic.com:
```

```
7001",PathPrepend=/jspfiles
PathCheck fn=nt-uri-clean
PathCheck fn="check-acl" acl="default"
PathCheck fn=find-pathinfo
PathCheck fn=find-index index-names="index.html,home.html"
ObjectType fn=type-by-extension
ObjectType fn=force-type type=text/plain
Service method=(GET|HEAD) type=magnus-internal/imagemap fn=imagemap
Service method=(GET|HEAD) type=magnus-internal/directory fn=index-common
Service method=(GET|HEAD) type=*~magnus-internal/* fn=send-file
AddLog fn=flex-log name="access"
</Object>
# The following directive enables HTTP-tunneling of the
# WebLogic protocol through the NES plug-in.
<Object name="tunnel" ppath="*/HTTPClnt*">
Service fn=wl-proxy WebLogicCluster="myweblogic.com:7001, yourweblogic.com:7001,
theirweblogic.com:7001"
</Object>
#
## -------------END SAMPLE obj.conf CONFIGURATION--------------------
```

4.3 Deprecated Directives for iPlanet Web Server

The WebLogic Server plug-in logs are now part of the Oracle iPlanet Server error log. References can be identified with the prefix `weblogic:` to easily identify them, for example:

```
[14/May/2015:23:18:34] fine (21904): for host 10.184.61.77 trying to GET /wls/
testcluster, wl-proxy reports: <2190414316707144> weblogic: After trimming path: '/
testcluster'
```

The directives WLLogFile and Debug are deprecated. If the configuration uses these directives, the following note will appear:

```
[14/May/2015:23:18:34] warning (21904): wl-proxy reports: weblogic: Debug &
WLLogFile directives are deprecated for the current plugin release. Please Refer to
the plugin documentation.
```

To enable plug-in logs, set log-level to `fine`.

5
Configuring the Plug-In for Microsoft IIS Web Server

To install and configure the Oracle WebLogic Server Proxy Plug-In for Microsoft IIS Web Server, download the Oracle WebLogic Server Proxy Plug-In for IIS Web Server, as described in Availability of Oracle WebLogic Server Proxy Plug-In. The zip file contains the following files:

Table 5-1   Files Included in the Microsoft IIS Plug-In Zip

(path)/filename	Description
README.txt	Information specific to the distribution, late-breaking updates, and other errata.
bin/orapki.bat	orapki tool for configuring Oracle wallets
jlib/*.jar	orapki helper Java libraries
iisproxy.dll	WebLogic proxy module
lib/*.dll	Helper libraries

WebLogic Server Proxy Plug-in 12.2.1.0.0 and later version builds are moved from Intel compiler to MSVC Compiler. When Microsoft IIS Web Server is used as a front end with WebLogic Server Proxy Plug-in, the plug-in library depends on the two dlls — `msvcp110.dll` and `msvcr110.dll`, provided by Microsoft. These dlls are available with Microsoft Visual C ++ Redistributable Package for x64.

For information about the specific versions of Microsoft IIS Web Server that are supported, see the Oracle Fusion Middleware Supported System Configurations page on the Oracle Technology Network.

Learn about how to install and configure the plug-in for Microsoft IIS Web Server in the following sections:

- Installing and Configuring the Plug-In for Microsoft Internet Information Server
- Serving Static Files with IIS
- Serving Static Files and Dynamic Content From the Same Request with IIS
- Using Wildcard Application Mappings to Proxy by Path
- Proxying Requests from Multiple Virtual Web Sites to WebLogic Server
- Creating ACLs Through IIS
- Testing the Installation

5.1 Installing and Configuring the Plug-In for Microsoft Internet Information Server

Follow these steps to install and configure the Plug-In for Microsoft Internet Information Server (IIS):

1. Install Microsoft IIS.

2. Ensure that all of the necessary features of Microsoft IIS are enabled.

   a. In the **Start** menu, select **Control Panel**, and then select **Programs and Features**.

   b. Select **Internet Information Services (IIS) Manager**.

   c. Click **Turn Windows features on or off**.

   d. Expand the entire tree beneath Internet Information Services and ensure **all** of the sub-features are selected.

Figure 5-1   Windows Features for Internet Information Services
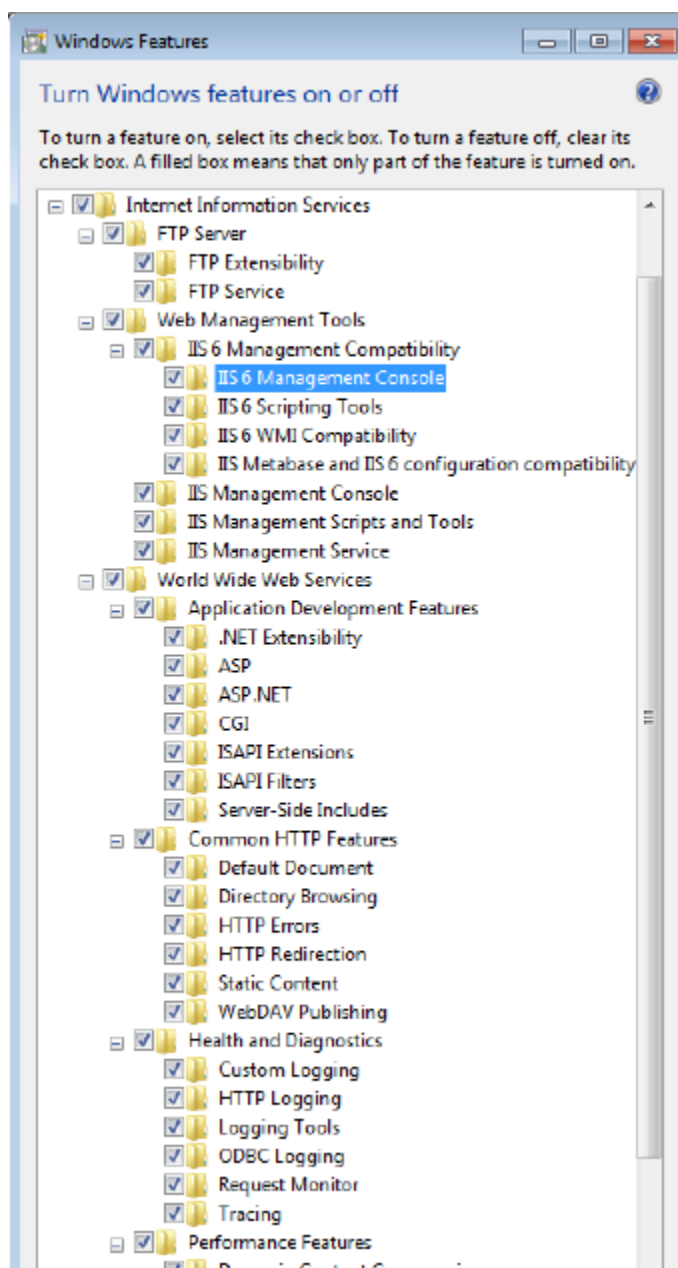
3. Download and install the latest Oracle WebLogic Server Proxy Plug-In zip file.

4. Create an `iisproxy.ini` file with the following content in the `%PLUGIN_HOME%\lib\` folder:

```
WebLogicHost=URL_of_WebLogic_Host
WebLogicPort=WebLogic_Port
Debug=ALL
DebugConfigInfo=ON
WLLogFile=C:\Temp\wl-proxy.log
```

5. Ensure that the `%PLUGIN_HOME%\lib` folder is included in the system `PATH`. (Select **Control Panel**, select **System**, select **System Properties**, select **Environment Variables**, select **System Properties**, and then select **PATH**.

6. Open IIS Manager. Select **Start**, select **All Programs**, select **Administrative Tools**, and then select **Internet Information Services Manager**.

7. Create a new Web Site in IIS. See the IIS Help system for more information.

8. Click the site name, open **Handler Mappings** and add a script map (set the **Extension** to a value such as `.jsp`, set **Executable** to `%PLUGIN_HOME%\lib \iisproxy.dll`, and assign a value to **Name**).

9. Click **MIME Types** and ensure a MIME type has been defined for the extension. Add the MIME type and its definition if it is not present.

10. Click the site name, open **Directory browsing** and enable the feature.

11. Start IIS. Enter `services.msc` at the command prompt and go to "World Wide Web Publishing Services" at the bottom and restart it. Also restart the web site.

Figure 5-2   Windows Services Window
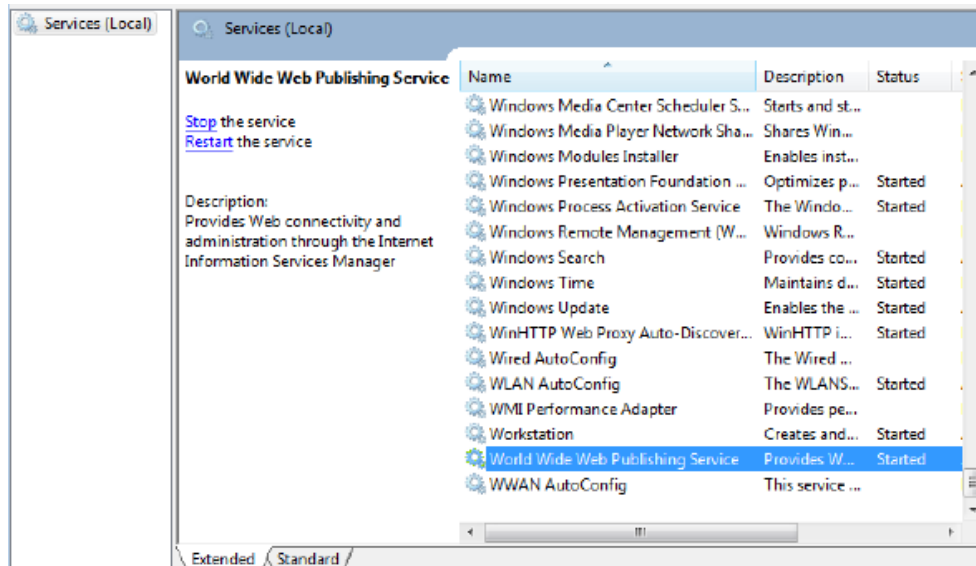
12. Test your configuration by sending a request to `http://`*iishost*`:`*iisport*`/` *application_name*`/` from the browser and validate the response.

    Here, `iishost` is the URL of the IIS server and `iisport` is the port number. Note that the `iisport` number should be different from the port number of the WebLogic Server.

5.2 Serving Static Files with IIS

After configuring the WLS plug-in and confirming it works (see Installing and Configuring the Plug-In for Microsoft Internet Information Server) follow these steps to serve static files.

1. Right click **Default web site** and then click **Add Virtual Directory**.

2. In the **Alias** field enter `static` and set the physical path to the location of the static files, for example `c:\inetpub\wwwroot\static`. Click **OK**. A **static** folder will appear under **Default Web Site**.

    Note: The physical path may be different in your case if the files are in a different location. Modify the path accordingly.
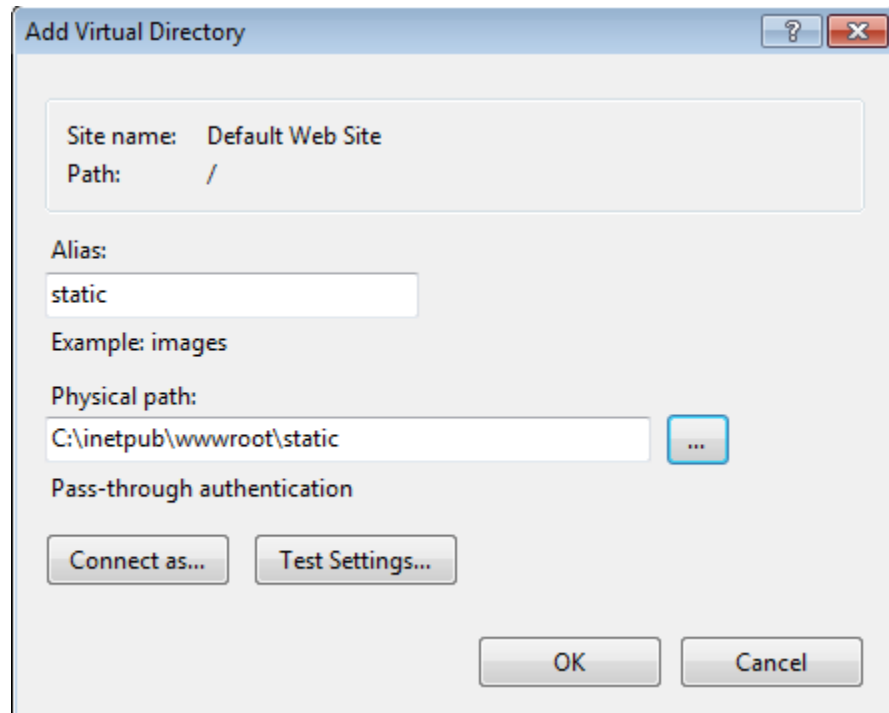
3. Click **static** under **Default Web Site** to open the **static Home** page.

4. On the **static Home** page click **Handler Mappings** and then click **View Ordered List** on the right-side pane. You will see an ordered list of **Handler Mappings**.

5. Click Add Script Map. Set **Request Path** to "*", set **Executable** to `%PLUGIN_HOME%\lib\iiproxy.dll`, and assign the value `proxy` to **Name**). Click OK.

6. Click **View Ordered List** to re-order the list of handlers.

7. Click the **proxy** script map and move it down below the **StaticFile** handler mapping. (That means the **StaticFile** handler mapping should appear above the **proxy** handler mapping.)

8. Create a `static` folder under `c:\inetpub\wwwroot` and copy an HTML file into it, for example `index.html`.

9. Restart IIS by restarting the "World Wide Web Publishing Service" under services.

10. Test your work. Access the `index.html` file by accessing: `http://localhost:80/static/index.html`

5.3 Serving Static Files and Dynamic Content From the Same Request with IIS

Suppose you want to serve the static files such as `*.gif` , `*.png` images for the request `http://localhost:80/console` from the IIS and other dynamic content from the backend WebLogic Server. Follow these steps in addition to Serving Static Files with IIS.

1. Complete the steps described in the Serving Static Files with IIS.

2. Right-click the **Default web site**, then click **Add Virtual Directory**, and then enter the following in the **Add Virtual Directory** dialog box.

- **Alias**—`console`
- **Physical Path**—`C:\`*`path_to_the_wls_plug-in`*`\console`

In this example, the physical path to the console is `C:\OHS_Plugin_IIS\console`.
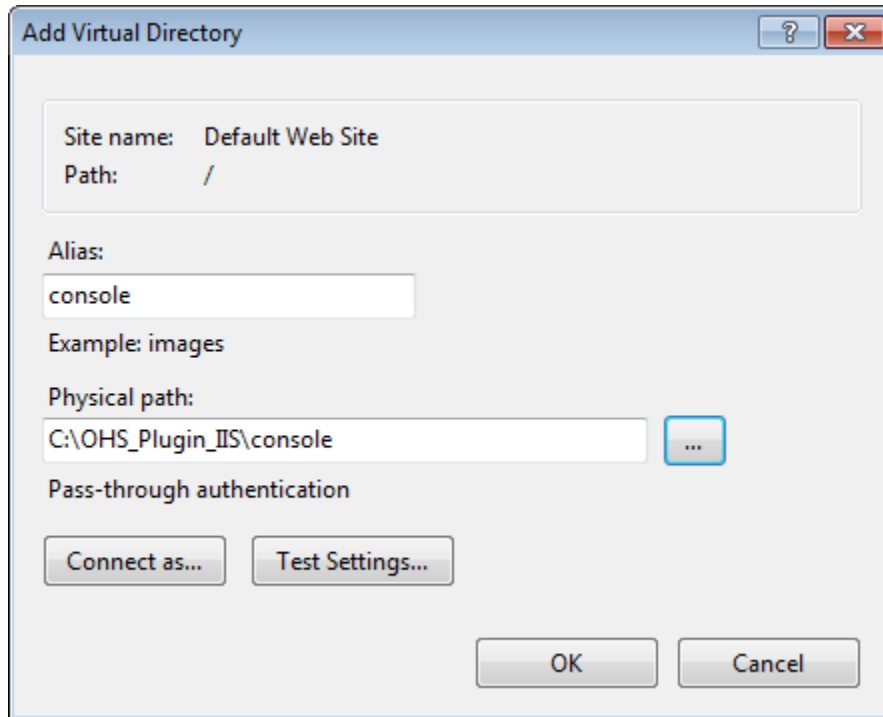
Click **OK**. You will see a `console`entry in the left pane under the Default Web Site.

3. Click **console** under **Default Web Site**. A **console Home** opens on the right side.

4. Click **Handler Mappings** on the **console Home** pane.

5. Right-click **StaticFile** and select **Edit**. Update the fields as follows in the **Edit Module Mapping** dialog box:

   - Request Path—*.png (that is, change "*" to "*.png")

   - Module— StaticFileModule, DefaultDocumentModule, DirectoryListingModule (should be the default)

   - Executable—not required. Leave it blank.

   - Name—StaticFile (it is not possible to change Name here)
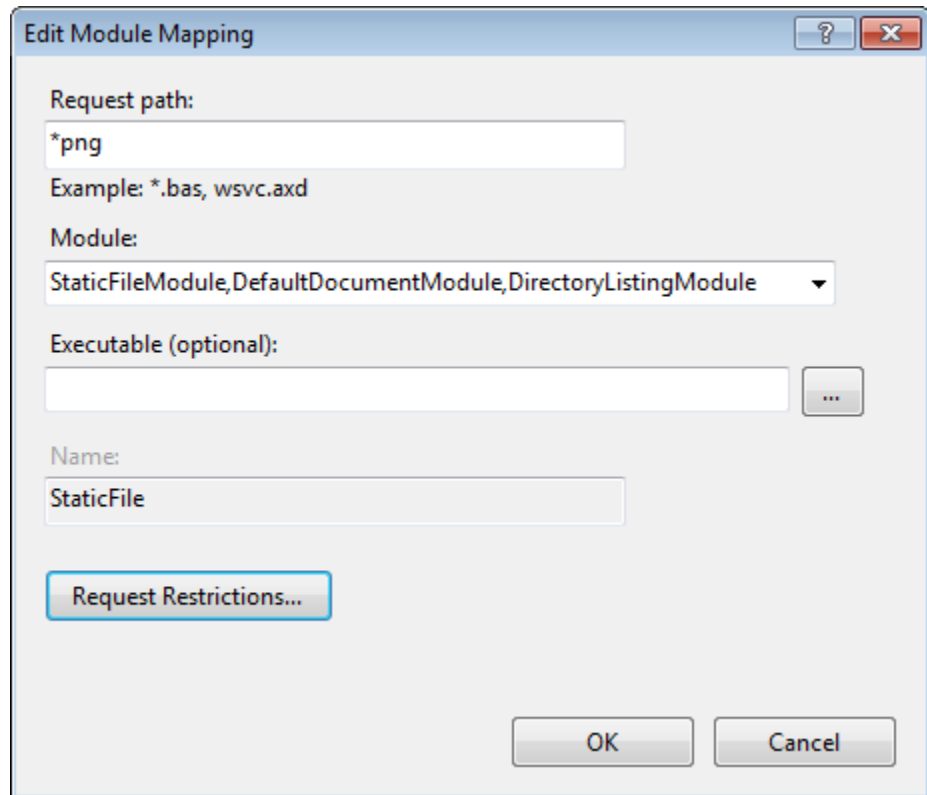
6. Click **Request Restrictions**. In the **Mapping** tab of the Request Restrictions dialog box, ensure that **Invoke Handler only if the request is mapped to** is selected, then select **File or Folder**. Click **OK** and **OK** to dismiss the dialog boxes.
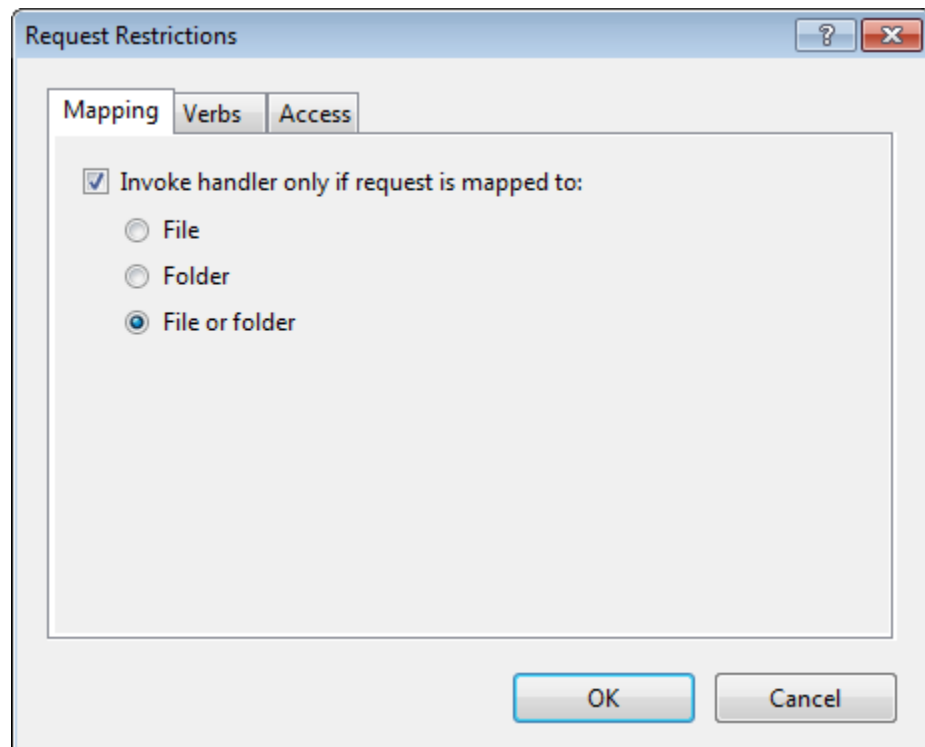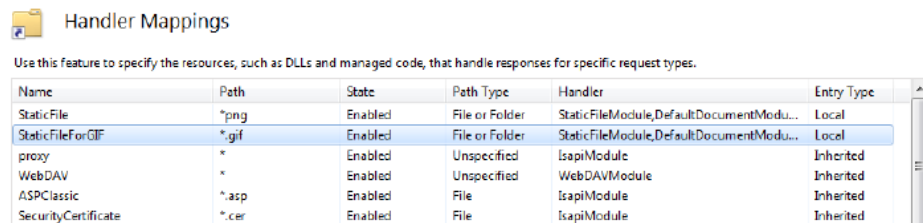
The above step is to serve the *.png from the IIS server.

7. To serve other image files, such as *.gif files, do the following.

    a. Under "Console Home" click "Handler Mappings", then click on "Add Module Mapping" on the right side and then enter the following.

       Request Path—*.gif

       Module—StaticFileModule, DefaultDocumentModule, DirectoryListingModule

       Executable—not required. Keep it blank

       Name—StaticFileForGIF

    b. Click "Request Restrictions". Under "Mapping" make sure the "Invoke Handler only if the request is mapped to" is selected and then select the "File or Folder" and then click "OK" and "OK"

8. Arrange the order of the handlers.

    a. Click on "console" under "Default web site" and then click on "Handler Mappings" and then "View Ordered List" on the right side

    b. Select "proxy" and move it down till the "proxy" is below the "StaticFile" and "StaticFileForGIF. That is, the order should be like below.

```
StaticFile
StaticFileForGIF
proxy
```

Handler Mappings

Use this feature to specify the resources, such as DLLs and managed code, that handle responses for specific request types.

Name	Path	State	Path Type	Handler	Entry Type
StaticFile	*.png	Enabled	File or Folder	StaticFileModule,DefaultDocumentModu...	Local
StaticFileForGIF	*.gif	Enabled	File or Folder	StaticFileModule,DefaultDocumentModu...	Local
proxy	*	Enabled	Unspecified	IsapiModule	Inherited
WebDAV	*	Enabled	Unspecified	WebDAVModule	Inherited
ASPClassic	*.asp	Enabled	File	IsapiModule	Inherited
SecurityCertificate	*.cer	Enabled	File	IsapiModule	Inherited

9. Copy all of the static files that belong to `http://loclhost:<iis-port>/console` request from WebLogic Server to IIS.

10. Restart the service, then restart the web site.

The images (`*.png` and `*.gif`) are now served by IIS and dynamic content by Weblogic server.

For example, for the request `http://localhost:80/console` the images for console are served by the IIS and all other requests other than `*.png` and `*.gif` are served by Weblogic Server.

5.4 Using Wildcard Application Mappings to Proxy by Path

You can configure a website or virtual directory to run an Internet Server API (ISAPI) application at the beginning of every request to that website or virtual directory, regardless of the extension of the requested file. You can use this feature to insert a mapping to iisproxy.dll and thereby proxy requests by path to WebLogic Server.

5.4.1 Adding a Wildcard Script Map for IIS

The following steps summarize the instructions available at "Add a Wildcard Script Map" for IIS (`http://technet.microsoft.com/en-us/library/cc754606(WS.10).aspx`) to add a wildcard script map to do proxy-by-path with ISAPI in IIS:

1. Open IIS Manager and navigate to the level you want to manage.

    For information about opening IIS Manager, see *Open IIS Manager* at `http://technet.microsoft.com/en-us/library/cc770472(WS.10).aspx`.

    For information about navigating to locations in the UI, see *Navigation in IIS Manager* at `http://technet.microsoft.com/en-us/library/cc732920(WS.10).aspx`.

2. In Features View, on the server, site, or application Home page, double-click **Handler Mappings**.

3. On the Handler Mappings page, in the Actions pane, click Add Wildcard Script Map.

4. In the **Executable** box, type the full path or browse to the `iisproxy.dll` that processes the request. For example, type `systemroot\system32\inetsrv\iisproxy.dll`.

5. In the **Name** box, type a friendly name for the handler mapping.

6. Click **OK**.

7. Optionally, on the Handler Mappings page, select a handler to lock or unlock. When you lock a handler mapping, it cannot be overridden at lower levels in the configuration. Select a handler mapping in the list, and then in the Actions pane, click **Lock** or **Unlock**.

8. After you add a wildcard script map, you must add the executable to the ISAPI and CGI Restrictions list to enable it to run. For more information about ISAPI and CGI restrictions, see *Configuring ISAPI and CGI Restrictions* at `http://technet.microsoft.com/en-us/library/cc730912(WS.10).aspx`.

> **Note:**
>
> If you are proxying a request to multiple IIS applications within the same IIS site, to prevent the subsequent request from proxying to the first website only, create each IIS application and assign a unique application pool to each IIS application.
>
> With IIS 7.x, you cannot assign application pools to virtual directories.

5.5 Proxying Requests from Multiple Virtual Web Sites to WebLogic Server

To proxy requests from multiple websites (defined as virtual directories in IIS) to WebLogic Server:

1. Create a new directory for the virtual directories. This directory will contain .dll and .ini files used to define the proxy.

2. Extract the contents of the plug-in .zip file to a directory.

3. For each virtual directory you configured, copy the contents of the plug-in \lib folder to the directory you created in Step 1.

4. Create an iisproxy.ini file for the virtual websites, as described in Sample iisproxy.ini File. Copy this iispoxy.ini file to the directory you created in Step 1.

5. Copy iisproxy.dll to the directory you created in Step 1.

5.5.1 Sample iisproxy.ini File

The following sample iisproxy.ini file can be used with a single, non-clustered WebLogic Server. Comment lines are denoted with the "#" character.

```
# This file contains initialization name/value pairs
# for the IIS/WebLogic plug-in.
WebLogicHost=localhost
WebLogicPort=7001
ConnectTimeoutSecs=20
ConnectRetrySecs=2
```

The following sample iisproxy.ini file can be used with clustered WebLogic Servers. Comment lines are denoted with the "#" character.

```
# This file contains initialization name/value pairs
# for the IIS/WebLogic plug-in.
WebLogicCluster=myweblogic.com:7001,yourweblogic.com:7001
ConnectTimeoutSecs=20
ConnectRetrySecs=2
```

> **Note:**
>
> If you are using SSL between the plug-in and WebLogic Server, the port number should be defined as the SSL listen port.

5.6 Creating ACLs Through IIS

ACLs will not work through the Oracle WebLogic Server Proxy Plug-In 12*c* (12.2.1.3.0) for Microsoft IIS Web Server if the Authorization header is not passed by IIS. Use the following information to ensure that the Authorization header is passed by IIS.

1. Ensure that the user is logged on with local log-on rights when using Basic Authentication.

2. Grant each user account the Log On Locally user right on the IIS server. To enable the use of Basic Authentication, Two problems may result from Basic Authentication's use of local logon:

   - If the user does not have local logon rights, Basic Authentication does not work even if the FrontPage, IIS, and Windows NT configurations appear to be correct.

- A user who has local log-on rights and who can obtain physical access to the host computer running IIS will be permitted to start an interactive session at the console.

3. To enable Basic Authentication, in the Directory Security tab of the console, ensure that the Allow Anonymous option is "on" and all other options are "off".

5.7 Testing the Installation

Follow these steps to ensure that the Microsoft IIS plug-in has been installed, configured, and deployed successfully.

1. Make sure WebLogic Server and IIS are running.

2. Save a JSP file into the document root of the default Web Application.

3. Open a browser and set the URL to the IIS plus *filename*.jsp, as shown in this example:

```
http://myiis.server.com/filename.jsp
```

If *filename*.jsp is displayed in your browser, the plug-in is functioning.

6

Configuring Security

This chapter describes how to work with security for plug-ins. It contains the following sections:

- Using SSL with Plug-Ins
- Configuring SSL with the Plug-In for iPlanet Web Server
- Configuring Perimeter Authentication

6.1 Using SSL with Plug-Ins

You can use the Secure Sockets Layer (SSL) protocol to protect the connection between the plug-in and Oracle WebLogic Server. The SSL protocol provides confidentiality and integrity to the data passed between the plug-in and WebLogic Server.

The plug-in does not use the transport protocol (HTTP or HTTPS) specified in the HTTP request (usually by the browser) to determine whether to use SSL to protect the connection between the plug-in and WebLogic Server; that is, the plug-in is in no way dependent on whether the HTTP request (again, usually from the browser) uses HTTPS (SSL).

Instead, the plug-in uses SSL parameters that you configure for the plug-in, as described in SSL Parameters for Web Server Plug-Ins, to determine when to use SSL:

- `WebLogicSSLVersion`—Specifies the SSL protocol version to use for communication between the plug-in and the WebLogic Server.

- `WLSSLWallet`—The version 12*c* (12.2.1.2.0) plug-ins use Oracle wallets to store SSL configuration information. Use the `WLSSLWallet` SSL configuration parameter to configure the wallets. The orapki utility is provided in the plug-in distribution for this purpose.

  The orapki utility manages public key infrastructure (PKI) elements, such as wallets and certificate revocation lists, on the command line so the tasks it performs can be incorporated into scripts. This enables you to automate many of the routine tasks of maintaining a PKI. See Using the orapki Utility for Certificate Validation and CRL Management.

- `SecureProxy`—The `SecureProxy` parameter determines whether SSL is enabled.

> ✏ **Note:**
>
> For more information on valid security protocols and ciphers for the current release, see SSLCipherSuite and SSLProtocol in *Administering Oracle HTTP Server*.

In the case of two-way SSL, the plug-in (the SSL client) automatically uses two-way SSL when Oracle WebLogic Server is configured for two-way SSL and requests a client certificate.

If a client certificate is not requested, the plug-ins default to one-way SSL.

> **Note:**
>
> If an Oracle Fusion Middleware 12*c* (12.2.1.2.0) product is installed on the same system as the Apache (including Oracle HTTP Server) plug-in, the *ORACLE_HOME* variable must point to a valid installation; otherwise, the plug-in fails to initialize SSL.
>
> For example, if *ORACLE_HOME* is invalid because the product was not cleanly removed, the plug-in fails to initialize SSL.

This section contains the following information:

- Configuring Libraries for SSL
- Configuring a Plug-In for One-Way SSL
- Configuring Two-Way SSL Between the Plug-In and Oracle WebLogic Server
- Replacing Certificates Signed Using the MD5 Algorithm
- Enabling Support of Certificate Signed with MD5 Algorithm

6.1.1 Configuring Libraries for SSL

Plug-ins use Oracle libraries (NZ) to provide SSL support. Because the libraries are large, they are loaded only when SSL is needed. You must ensure that the library files, located in lib/*.so*, are available at the proper locations so that they can be dynamically loaded by the plug-in.

To configure the libraries for the plug-ins for Apache HTTP Server, you have a few options:

- Windows: Specify the `lib` directory that contains the .dll files in the *PATH* variable or copy the *.dll files in the `bin` directory.
- UNIX: Configure *LD_LIBRARY_PATH* to point to the folder containing the libraries or copy the libraries to the `lib` directory.

If you copy the libraries instead of updating the `PATH` (Windows) or *LD_LIBRARY_PATH* (UNIX) variables, you must copy the libraries afresh each time you install a new version of the plug-in.

6.1.2 Configuring a Plug-In for One-Way SSL

Perform the following steps to configure one-way SSL.

In these steps, you run the keytool commands on the system on which WebLogic Server is installed, and you run the orapki commands on the system on which the version 12*c* (12.2.1.2.0) plug-ins are installed.

> **Note:**
>
> The examples in this section use the WebLogic Server demo CA. If you are using the plug-in a production environment, ensure that trusted CAs are properly configured for the plug-in and for Oracle WebLogic Server.

1. Configure Oracle WebLogic Server for SSL. See Configuring SSL in *Administering Security for Oracle WebLogic Server*.

2. Create an Oracle Wallet, by using the orapki utility.

   ```
   orapki wallet create -wallet mywallet -auto_login_only
   ```

   See Using the orapki Utility for Certificate Validation and CRL Management in the *Administering Oracle Fusion Middleware*.

   > **Note:**
   >
   > Only the user who creates the wallet (or for Windows, the account SYSTEM) has access to the wallet.
   >
   > This is typically sufficient for the Oracle WebLogic Server Proxy Plug-In for Apache HTTP Server because Apache runs as the account SYSTEM on Windows, and as the user who creates it on UNIX. However, for IIS the wallet will not work because the default user is IUSR_<*Machine_Name*>(IIS6.0 and below) or IUSR (IIS7.0 and above).
   >
   > If the user who runs the Oracle WebLogic Server Proxy Plug-In for Apache HTTP Server or Oracle WebLogic Server Proxy Plug-In 12*c* (12.2.1.2.0) for Microsoft IIS Web Server is different from the user who creates the wallet (or for Windows, the account SYSTEM), you need to grant the user access to the wallet by running the command `cacls` (Windows) or `chmod` (UNIX) after you create the wallet. For example:
   >
   > ```
   > cacls <wallet_path>\cwallet.sso /e /g IUSR:R
   > ```

3. Import the WLS trust certificate into the Oracle Wallet.

   ```
   orapki wallet add -wallet mywallet -trusted_cert -cert <cert_file_name> -
   auto_login_only
   ```

4. Configure the web server configuration files as follows:

   - For Oracle HTTP Server, edit the `mod_wl_ohs.conf` file as follows:

     ```
     <IfModule mod_weblogic.c>
      WebLogicHost host
      WebLogicPort port
      SecureProxy ON
      WLSSLWallet path_to_wallet
     </IfModule>
     ```

   - For Microsoft IIS, edit the `iisproxy.ini` file as follows:

     ```
     WebLogicHost=host
     WebLogicPort=port
     ```

ORACLE

```
SecureProxy=ON
WLSSLWallet=path_to_wallet
```

For more information about the parameters in these examples, see Parameters for Web Server Plug-Ins.

5. Complete these steps if the version of the Oracle WebLogic Server instances in the back end is 10.3.4 (or a later release).

   a. Log in to the Oracle WebLogic Server administration console.

   b. In the Domain Structure pane, expand the **Environment** node.

      • If the server instances to which you want to proxy requests from Oracle HTTP Server are in a cluster, select **Clusters**.

      • Otherwise, select **Servers**.

   c. Select the server or cluster to which you want to proxy requests from Oracle HTTP Server.

   d. In the Configuration: General tab, scroll down to the Advanced section, then expand it.

   e. Do one of the following:

To...	Select...
Enable one-way SSL	**WebLogic Plug-In Enabled**
Enable two-way SSL where client certificates are used to authenticate	**Client Cert Proxy Enabled**
Enable two-way SSL with client certificates.	Both

   f. If you selected **Servers** in Step 2, repeat steps Step 3 and Step 4 for the other servers to which you want to proxy requests from Oracle HTTP Servers.

   g. Click **Save**.

   For the change to take effect, you must restart the server instances.

6. Send a request to `http://host:port/mywebapp/my`.jsp from the browser and validate the response.

6.1.3 Configuring Two-Way SSL Between the Plug-In and Oracle WebLogic Server

When Oracle WebLogic Server is configured for two-way SSL, the plug-in forwards the user certificate to WebLogic Server. As long as WebLogic Server can validate the user certificate, two-way SSL can be established.

In addition to the steps described in Configuring a Plug-In for One-Way SSL, perform the following steps:

In these steps, you run the keytool commands on the system on which WebLogic Server is installed. You run the orapki commands on the system on which the version 12*c* (12.2.1.2.0) plug-ins are installed.

1. From the Oracle wallet, generate a certificate request.

2. Use this certificate request to create a certificate by using a CA or some other mechanism.

3. Import the user certificate as a trusted certificate in the WebLogic trust store. Oracle WebLogic Server needs to trust the certificate.

```
keytool -file user.crt -importcert -trustcacerts -keystore DemoTrust.jks -
storepass <passphrase>
```

4. Set the WebLogic Server SSL configuration options that require the presentation of client certificates (for two-way SSL). See *Configure two-way SSL* in *Oracle WebLogic Server Administration Console Online Help*.

6.1.4 Replacing Certificates Signed Using the MD5 Algorithm

When using SSL to connect to WebLogic Server, ensure that any certificate request or certificates signed with MD5 are replaced by SHA-2 signed certificates in the wallet; otherwise, the server will fail to start.

Checking the Certificate Singing Algorithm

To check the certificate singing algorithm :

1. To search the certificate with it's distinguished name, using the following command

```
${PLUGIN_HOME}/bin/orapki wallet display -wallet
<wallet__location>
```

2. Export certificate available in wallet

```
${PLUGIN_HOME}/bin/orapki  wallet export -wallet
<wallet_Location> -dn 'DN_string' -cert <certificate_file>
```

3. Check the signature algorithm used to sign <certificate_file> using the keytool

```
$JAVA_HOME/bin/keytool -printcert -file <certificate_file>
```

Removing a Certificate Request or Certificate Signed with MD5 algorithm

- To remove a user certificate signed using MD5 algorithm

```
    ${PLUGIN_HOME}/bin/orapki wallet remove -wallet <wallet_location> -dn
'DN_string' -user_cert [-pwd <pwd>] | [-auto_login_only]
```

- To remove a self-signed certificate available in the trusted and requested certificate list:

```
${PLUGIN_HOME}/bin/orapki wallet remove -wallet < wallet_location > -dn
'DN_string' -trusted_cert [-pwd <pwd>] | [-auto_login_only]
 ${PLUGIN_HOME}/bin/orapki wallet remove -wallet < wallet_location > -dn
'DN_string' -cert_req [-pwd <pwd>] | [-auto_login_only]
```

- To remove a trusted certificate signed using MD5 algorithm

```
${PLUGIN_HOME}/bin/orapki wallet remove -wallet < wallet_location > -dn
'DN_string' -trusted_cert [-pwd < pwd >] | [-auto_login_only]
```

- To remove a certificate request signed using MD5 algorithm

```
${PLUGIN_HOME}/bin/orapki wallet remove -wallet < wallet_location > -dn
'DN_string' -cert_req [-pwd <pwd>] | [-auto_login_only]
```

Adding a Self-Signed User Certificate Signed with SHA-2 Algorithm

Use the following command to add a self-signed user certificate, signed using MD5 algorithm with a self-signed certificate signed using a SHA-2 algorithm in the wallet:

```
${PLUGIN_HOME}
/bin/orapki wallet add -wallet <wallet_Location>
-dn 'DN_String'
keysize 2048 -sign_alg sha256 -self_signed
-validity 9125 [-pwd <pwd>] | [-auto_login_only]
```

Updating the Existing Certificate Authority Signed User Certificate Using MD5 Algorithm

Contact the certificate authority to get a user certificate signed using SHA-2 signature algorithm and replace it with existing user certificate.

```
${PLUGIN_HOME}/bin/orapki -wallet add -wallet <wallet_Location> -user_cert -cert
<certificate_file> [-pwd <pwd> ] | [-auto_login_only]
```

Updating the Existing Trusted Certificates Signed Using MD5 Algorithm

If you have any trusted certificate that is signed using MD5 signature algorithm imported in your wallet, update the certificate of the corresponding backend WebLogic Server to use the SHA-2 signature algorithm. Once updated, replace the MD5 trusted certificate in your wallet with the updated certificate.

```
${PLUGIN_HOME}/bin/orapki -wallet add -wallet <wallet_Location> -trusted_cert -cert
<certificate_file> [-pwd <pwd> ] | [-auto_login_only]
```

6.1.5 Enabling Support of Certificate Signed with MD5 Algorithm

> **Note:**
>
> Certificates signed using MD5 algorithm are not recommended, due to compromised security. To continue using certificates signed using MD5 algorithm by setting `ORACLE_SSL_ALLOW_MD5_CERT_SIGNATURES=1` environment variable.

Set the `environment. variable` in the plugin:

*   Oracle HTTP Server Plugin: Add `environment.ORACLE_SSL_ALLOW_MD5_CERT_SIGNATURES=1` in `DOMAIN_HOME/config/fmwconfig/components/OHS/instances/ instanceName /ohs.plugin.nodemanger.properties`

*   Apache plugin : Add `export ORACLE_SSL_ALLOW_MD5_CERT_SIGNATURES=1` in `$APACHE_HOME/bin/envvars`.

*   iPlanet plugin : Add `export ORACLE_SSL_ALLOW_MD5_CERT_SIGNATURES=1` in `$IPLANET_INSTANCE_HOME/bin/startserv`.

*   IIS Plugin : Add `ORACLE_SSL_ALLOW_MD5_CERT_SIGNATURES=1` in System environment variable.

6.2 Configuring SSL with the Plug-In for iPlanet Web Server

You can use the Secure Sockets Layer (SSL) protocol to protect the connection between the Oracle WebLogic Server Proxy Plug-In 12*c* (12.2.1.2.0) for iPlanet Web Server plug-in and Oracle WebLogic Server. The SSL protocol provides confidentiality and integrity to the data passed between the Oracle iPlanet Web Server plug-in and Oracle WebLogic Server.

The Oracle WebLogic Server Proxy Plug-In 12*c* (12.2.1.2.0) for iPlanet Web Server plug-in does not use the transport protocol (http or https) specified in the HTTP request (usually by the browser) to determine whether the SSL protocol will be used to protect the connection between the Oracle WebLogic Server Proxy Plug-In 12*c* (12.2.1.2.0) for iPlanet Web Server and Oracle WebLogic Server.

To use the SSL protocol between Oracle iPlanet Web Server plug-in and Oracle WebLogic Server:

1. Configure Oracle WebLogic Server for SSL. See Configuring SSL in *Administering Security for Oracle WebLogic Server*.

2. Set the `WebLogicPort` parameter in the `Service` directive in the obj.conf file to the listen port configured in Step 1.

3. Set the `SecureProxy` parameter in the `Service` directive in the obj.conf file to `ON`.

4. Set additional parameters, as required, in the `Service` directive in the `obj.conf` file that define information about the SSL connection. For the list of parameters, see SSL Parameters for Web Server Plug-Ins.

6.3 Configuring Perimeter Authentication

Use perimeter authentication to secure WebLogic Server applications that are accessed by using the plug-in.

A WebLogic Identity Assertion Provider authenticates tokens from outside systems that access your WebLogic Server application, including users who access your WebLogic Server application through the plug-in. Create an Identity Assertion Provider that will safely secure your plug-in as follows:

1. Create a custom Identity Assertion Provider on your WebLogic Server application. See How to Develop a Custom Identity Assertion Provider in *Developing Security Providers for Oracle WebLogic Server*.

2. Configure the custom Identity Assertion Provider to support the Cert token type and make Cert the active token type. See How to Create New Token Types in *Developing Security Providers for Oracle WebLogic Server*.

3. Set clientCertProxy to True in the web.xml deployment descriptor file for the Web application (or, if using a cluster, optionally set the `Client Cert Proxy Enabled` attribute to true for the whole cluster on the Administration Console **Cluster** then **Configuration** then **General** tab).

   The `clientCertProxy` attribute can be used with a third party proxy server, such as a load balancer or an SSL accelerator, to enable 2-way SSL authentication. For more information about the `clientCertProxy` attribute, see context-param in *Developing Web Applications, Servlets, and JSPs for Oracle WebLogic Server*.

4. Once you have set `clientCertProxy`, be sure to use a connection filter to ensure that WebLogic Server accepts connections only from the machine on which the plug-in is running. See Using Network Connection Filters in *Developing Applications with the WebLogic Security Service*.

5. Web server plug-ins require a trusted Certificate Authority file to use SSL between the plug-in and WebLogic Server. See Using SSL with Plug-Ins for the steps you need to perform to configure SSL.

See Identity Assertion Providers in *Developing Security Providers for Oracle WebLogic Server*.

7

Common Configuration Tasks

This chapter describes tasks that are common across all the web servers for configuring the plug-ins provided by Oracle. It contains the following sections:

- Configuring IPv6 With Plug-Ins
- Understanding Connection Errors and Clustering Failover
- Tuning Oracle HTTP Server and Apache HTTP Server for High Throughput for WebSocket Upgrade Requests
- Working with Partitions

7.1 Configuring IPv6 With Plug-Ins

The version 12*c* (12.2.1.2.0) plug-ins support IPv6. Specifically, the `WebLogicHost` and `WebLogicCluster` configuration parameters (see WebLogicCluster and WebLogicHost) now support IPv6 addresses. For example:

```
<IfModule mod_weblogic.c>
 WebLogicHost [a:b:c:d:e:f]
 WebLogicPort 7002
 ...
</IfModule>
```

or

```
<IfModule mod_weblogic.c>
 WebLogicCluster [a:b:c:d:e:f]:<port>, [g:h:i:j:k:l]:<port>

 ....
</IfModule>
```

You can also use the IPv6 address mapped host name.

> **Note:**
>
> As of Windows 2008, the DNS server returns the IPv6 address in preference to the IPv4 address. If you are connecting to a Windows 2008 (or later) system using IPv4, the link-local IPv6 address format is tried first, which may result in a noticeable delay and reduced performance. To use the IPv4 address format, configure your system to instead use IP addresses in the configuration files or add the IPv4 addresses to the `etc/hosts` file.
>
> In addition, you may find that setting the `DynamicServerList` property to OFF in the `mod_wl_ohs.conf/mod_wl.conf/iisproxy.ini` file also improves performance with IPv6. When set to OFF, the plug-in ignores the dynamic cluster list used for load balancing requests proxied from the plug-in and uses the static list specified with the WebLogicCluster parameter.

7.2 Understanding Connection Errors and Clustering Failover

When the plug-in attempts to connect to WebLogic Server, the plug-in uses several configuration parameters to determine how long to wait for connections to the WebLogic Server host and, after a connection is established, how long the plug-in waits for a response. If the plug-in cannot connect or does not receive a response, the plug-in attempts to connect and send the request to other WebLogic Server instances in the cluster. If the connection fails or there is no response from any WebLogic Server in the cluster, an error message is sent.

Figure 7-1 illustrates how the plug-in handles failover.

This section contains the following information:

- Possible Causes of Connection Failures
- Tips for Reducing CONNECTION_REFUSED Errors
- Failover with a Single, Non-Clustered WebLogic Server
- The Dynamic Server List
- Failover, Cookies, and HTTP Sessions
- Failover Behavior When Using Firewalls and Load Directors

7.2.1 Possible Causes of Connection Failures

Failure of the WebLogic Server host to respond to a connection request could indicate the following problems:

- Physical problems with the host machine
- Network problems
- Other server failures

Failure of all WebLogic Server instances to respond could indicate the following problems:

- WebLogic Server is not running or is unavailable
- A hung server
- A database problem
- An application-specific failure

7.2.2 Tips for Reducing CONNECTION_REFUSED Errors

Under load, a plug-in may receive CONNECTION_REFUSED errors from a back-end WebLogic Server instance. Follow these tuning tips to reduce CONNECTION_REFUSED errors:

- Increase the `AcceptBackLog` setting in the configuration of your WebLogic Server domain.

- Decrease the time wait interval. This setting varies according to the operating system you are using. For example:
    - On Windows NT, set the `TcpTimedWaitDelay` on the proxy and WebLogic Server servers to a lower value. Set the TIME_WAIT interval in Windows NT by editing the registry key under HKEY_LOCAL_MACHINE:

      ```
      SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\TcpTimedWaitDelay
      ```

      If this key does not exist you can create it as a DWORD value. The numeric value is the number of seconds to wait and may be set to any value between 30 and 240. If not set, Windows NT defaults to 240 seconds for TIME_WAIT.

    - On Windows 2000, lower the value of the `TcpTimedWaitDelay` by editing the registry key under HKEY_LOCAL_MACHINE:

      ```
      SYSTEM\CurrentControlSet\Services\Tcpip\Parameters
      ```

    - On Solaris, reduce the setting `tcp_time_wait_interval` to one second (for both the WebLogic Server machine and the Apache machine, if possible):

      ```
      $ndd /dev/tcp
       param name to set - tcp_time_wait_interval
       value=1000
      ```

- Increase the open file descriptor limit on your machine. This limit varies by operating system. Using the limit (.csh) or ulimit (.sh) directives, you can make a script to increase the limit. For example:

  ```
  #!/bin/sh
  ulimit -S -n 100
  exec httpd
  ```

- On Solaris, increase the values of the following tunables on the WebLogic Server machine:

  ```
  tcp_conn_req_max_q tcp_conn_req_max_q0
  ```

7.2.3 Failover with a Single, Non-Clustered WebLogic Server

If you run only a single WebLogic Server instance the plug-in only attempts to connect to the server defined with the `WebLogicHost` parameter. If the attempt fails, an HTTP 503 error message is returned. The plug-in continues trying to connect to that same WebLogic Server instance for the maximum number of retries as specified by the ratio of `ConnectTimeoutSecs` and `ConnectRetrySecs`.

7.2.4 The Dynamic Server List

The `WebLogicCluster` parameter is required to proxy to a list of back-end servers that are clustered, or to perform load balancing among non-clustered managed server instances.

In the case of proxying to clustered managed servers, when you use the `WebLogicCluster` parameter to specify a list of WebLogic Servers, the plug-in uses that list as a starting point for load balancing among the members of the cluster. After the first request is routed to one of these servers, a dynamic server list is returned containing an updated list of servers in the cluster.

The updated list adds any new servers in the cluster and deletes any that have been shut down, or are being suspended, or are no longer part of the cluster or that have

failed to respond to requests. This feature can be controlled by using `DynamicServerList`. For example, to disable this feature, set `DynamicServerList` to `OFF`.

`DynamicServerList ON` is a preferred performance tuning parameter. It is useful, for example, if a member of a cluster is temporarily down for maintenance or if administrators decide they want to add another member, and not need to restart the web server.

> **Note:**
>
> If `DynamicServerList` is set to `ON`, and the list of backend WebLogic Servers specified in `WebLogicCluster` is not in a cluster, then the behavior would be undefined.

7.2.5 Failover, Cookies, and HTTP Sessions

When a request contains session information stored in a cookie or in the POST data, or encoded in a URL, the session ID contains a reference to the specific server instance in which the session was originally established (called the primary server). A request containing a cookie attempts to connect to the primary server. If that attempt fails, the plug-in attempts to make a connection to the next available server in the list in a round-robin fashion. That server retrieves the session from the original secondary server and makes itself the new primary server for that same session. See Figure 7-1.

> **Note:**
>
> If the POST data is larger than 64K, the plug-in will not parse the POST data to obtain the session ID. Therefore, if you store the session ID in the POST data, the plug-in cannot route the request to the correct primary or secondary server, resulting in possible loss of session data.

Figure 7-1   Connection Failover
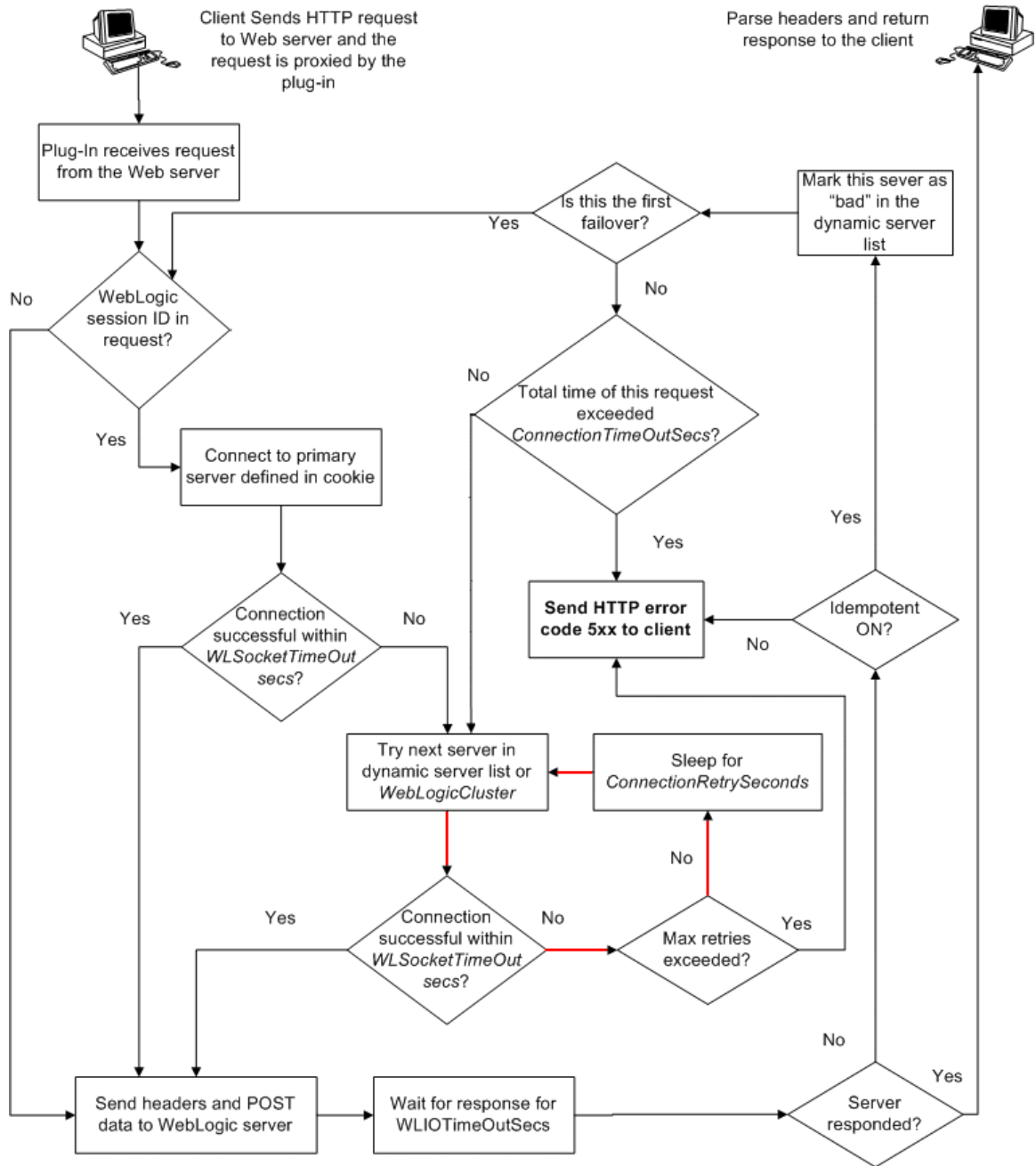
In this figure, the Maximum number of retries allowed in the red loop is equal to `ConnectTimeoutSecs/ConnectRetrySecs`.

7.2.6 Failover Behavior When Using Firewalls and Load Directors

In most configurations, the Oracle WebLogic Server Proxy Plug-In 12*c* (12.2.1.2.0) for iPlanet Web Server sends a request to the primary instance of a cluster. When that instance is unavailable, the request fails over to the secondary instance. However, in some configurations that use combinations of firewalls and load-directors, any one of

the servers (firewall or load-directors) can accept the request and return a successful connection while the primary instance of WebLogic Server is unavailable. After attempting to direct the request to the primary instance of WebLogic Server (which is unavailable), the request is returned to the plug-in as "connection reset."

Requests running through combinations of firewalls (with or without load-directors) are handled by WebLogic Server. In other words, responses of connection reset fail over to a secondary instance of WebLogic Server. Because responses of connection reset fail over in these configurations, servlets must be idempotent. Otherwise duplicate processing of transactions may result.

7.3 Tuning Oracle HTTP Server and Apache HTTP Server for High Throughput for WebSocket Upgrade Requests

WebLogic Server 12*c* (12.2.1.2.0) supports deploying WebSocket applications. Oracle WebLogic Server Proxy Plug-In 12*c* (12.2.1.2.0) for Oracle HTTP Server 12*c* (12.2.1.2.0) and Apache HTTP Server 2.2.x and 2.4.x can now handle such WebSocket connection upgrade requests and effectively proxy to WebSocket applications hosted within WebLogic Server 12*c* (12.2.1.2.0) and later. As a result of adding this support, a new configuration parameter WLMaxWebSocketClients is introduced.

The WLMaxWebSocketClients parameter limits the number of active WebSocket connections at any instant of time. The maximum value you can set for this parameter is 75 percent of ThreadsPerChild (Windows) or 75 percent of MaxRequestWorkers (non-Windows). Hence, to tune your HTTP Server for maximum WebSocket connection upgrade requests, set MaxRequestWorkers/ThreadsPerChild to a value that can accommodate WebSocket connections as well. Also, ensure that WLMaxWebSocketClients is set to 75 percent of MaxRequestWorkers/ThreadsPerChild.

7.4 Working with Partitions

By doing some manual configuration, Oracle WebLogic Server Proxy Plug-Ins can front-end Oracle WebLogic Server MT (Multi-Tenency).

This section contains the following information:

> **Note:**
>
> This section describes partitions and multi-tenancy only as far as they apply to Oracle WebLogic Server Proxy Plug-Ins. For more information about Oracle WebLogic Server MT, partitions, and multi-tenancy, see *Oracle Fusion Middleware Using WebLogic Server MT*.

- Adding a Partition
- Modifying a Partition and Partition Migration
- Configuring SSL Between the Web Server and Oracle WebLogic Server
- Dynamic Discovery of Cluster Changes

7.4.1 Adding a Partition

Whenever a partition is added on the Oracle WebLogic Server MT side, you must make corresponding changes to the web server configuration to front the new partition. The details of adding a partition itself is documented by Oracle WebLogic Server MT, (see "Configuring Domain Partitions" in *Oracle Fusion Middleware Using WebLogic Server MT*).

The following sections describe how to make corresponding changes to web server configuration to front-end a newly added partition in Oracle WebLogic Server MT.

> **Note:**
>
> These sections assume that you have already created one or more domain partitions in Oracle WebLogic Server. For more information on creating domain partitions, see Configuring Domain Partitions in *Oracle Fusion Middleware Using WebLogic Server MT*.

This section includes the following topics:

- Apache Server and Oracle HTTP Server Configuration Changes
- iPlanet Server Configuration Changes
- IIS Server Configuration Changes

7.4.1.1 Apache Server and Oracle HTTP Server Configuration Changes

The following is a sample configuration that must be added to the Oracle HTTP Server or Apache plug-in configuration file (`httpd.conf`) for each new partition being added. The configuration identifies the hostname and port of the partition, the server and the WebLogic cluster it belongs to, and any optional URI that is configured for the partition.

1. Install and configure the plug-in for Oracle HTTP Server or Apache Server. See Preparing for Configuring the Oracle WebLogic Server Proxy Plug-In or Install the Oracle WebLogic Server Proxy Plug-In for Apache HTTP Server.

2. Edit the `httpd.conf` file and add the following section for every new partition:

```
# For OHS: Uncomment the following line
# LoadModule weblogic_module modules/mod_wl_ohs.so
# For Apache 2.2: Uncomment the following line
# LoadModule weblogic_module modules/mod_wl.so
# For Apache 2.4: Uncomment the following line
LoadModule weblogic_module modules/mod_wl_24.so

<VirtualHost hostname_and_port_of_the_partition>
<IfModule mod_weblogic.c>
ServerName server_URL
WebLogicCluster comma_separated_list_of_WebLogic_clusters
SetHandler weblogic-handler
PathPrepend optional_uri_of_the_partition
</IfModule>
</VirtualHost>
```

3. After making the configuration change, you must restart the Apache or Oracle HTTP Server.

7.4.1.1.1 Oracle HTTP Server/Apache Plug-in Examples

The following examples provide sample configurations for different use cases.

Example 1: Client may not configure the partition path

Assume there are there are two partitions in the domain:

- Host Name of the `VirtualTarget1` is `www.myCompany1.com` and the URI Prefix (partition path) is null as configured in Oracle WebLogic Server.

- Host Name of the `VirtualTarget2` is `www.myCompany2.com` and the URI Prefix (partition path) is null as configured in Oracle WebLogic Server.

In this case, you do not need to configure the partition path

```
LoadModule weblogic_module modules/mod_wl_24.so

<VirtualHost *:8080>
<IfModule mod_weblogic.c>
ServerName www.myCompany1.com
WebLogicCluster wls1:7001,wls2:7001,wls3:7001
SetHandler weblogic-handler
</IfModule>
</VirtualHost>

<VirtualHost *:8080>
<IfModule mod_weblogic.c>
ServerName www.myCompany2.com
WebLogicCluster wls1:7002,wls2:7002,wls3:7002
SetHandler weblogic-handler
</IfModule>
</VirtualHost>
```

This configuration proxies all of the requests coming to `myCompany1.com:8080` or `myCompany1.com` to the managed servers `wls1:7001`, `wls2:7001` and `wls3:7001`. Similarly, all of the requests coming to `myCompany2.com:8080` or `myCompany2.com` will be proxied to the managed servers `wls1:7002`, `wls2:7002` and `wls3:7002`.

Example 2: Client may divide the web site by the partition path

Assume there are there are two partitions in the domain:

- Host Name of the `VirtualTarget1` is `www.foo.com` and URI Prefix (partition path) is `/myCompany1` as configured in Oracle WebLogic Server.

- Host Name of the `VirtualTarget2` is `www.foo.com` and URI Prefix (partition path) is `/myCompany2` as configured in Oracle WebLogic Server.

In this case, you do not need to configure the partition path:

```
<VirtualHost *:8080>
<IfModule mod_weblogic.c>
ServerName server1
WebLogicCluster wls1:7001,wls2:7001,wls3:7001
SetHandler weblogic-handler
</IfModule>
</VirtualHost>
```

The available urls will be:

```
www.foo.com:8080/myCompany1
```

```
www.foo.com:8080/myCompany2
```

7.4.1.2 iPlanet Server Configuration Changes

The following is a sample configuration that must be added to the iPlanet Server plug-in configuration file for each new partition being added.

1. Install the plug-in for iPlanet Server. See Installing and Configuring the Plug-In for iPlanet .

2. Edit the `magnus.conf` file to add the following section:

    ```
    ServerName server_URL
    Init fn="load-modules" shlib="path to the /plugin/mod_wl.so file"
    ```

    For example:

    ```
    ServerName www.myCompany.com
    Init fn="load-modules" shlib="/scratch/plugin/mod_wl.so"/"
    ```

3. Edit the `obj.conf` file to add the following section for every new partition:

    ```
    <Object name="weblogic" ppath="*">
    Service ChunkedRequestBufferSize="0" fn="wl-proxy"
    WebLogicCluster="comma_separated_list_of_WebLogic_clusters"
    PathPrepend="optional_uri_of_the_partition" SecureProxy="OFF"
    WLIOTimeoutSecs="10"
    </Object>
    ```

    For example:

    ```
    <Object name="weblogic" ppath="*">
    Service ChunkedRequestBufferSize="0" fn="wl-proxy"
    WebLogicCluster="wls1:7001,wls2:7001,wls3:7001" PathPrepend="/partition1"
    SecureProxy="OFF" WLIOTimeoutSecs="10"
    </Object>
    ```

4. After making the configuration change, you must restart the iPlanet Server.

7.4.1.3 IIS Server Configuration Changes

The following is a sample configuration that must be added to the Microsoft IIS plug-in configuration file for each new partition being added.

1. Install the plug-in for Microsoft IIS Server. See Installing and Configuring the Plug-In for Microsoft Internet Information Server.

2. Open IIS Manager and create a new Web Site.

    Set **Host name**. For example, set it to `www.myCompany.com`.

3. Add the plug-in to the IIS server.

    Click **Handler Mappings** in IIS Manager to set the mappings to the handler for a particular MIME type.

4. Create a plug-in configuration file:

    Create a file named `iisproxy.ini` with the following content and place it in the directory with the plug-in. Create similar content in the file for every new partition:

```
WebLogicCluster=comma_separated_list_of_WebLogic_clusters
Debug=ALL
WLLogFile=path to the wlproxy.log file
PathPrepend=optional_uri_of_the_partition
SecureProxy=OFF
WLIOTimeoutSecs=10
```

For example:

```
WebLogicCluster=wls1:7001,wls2:7001,wls3:7001
Debug=ALL
WLLogFile=C:/pp_iis_home/myCompany/logs/iis7.0_wlproxy.log
PathPrepend=/partition1
SecureProxy=OFF
WLIOTimeoutSecs=10
```

5. After making the configuration change, you must restart the Microsoft IIS Server.

7.4.2 Modifying a Partition and Partition Migration

Partition migration is an Oracle WebLogic Server MT feature. You cannot migrate partitions on the Oracle WebLogic Server Proxy Plug-In side. Once the partition is migrated, the Oracle WebLogic Server Proxy Plug-In configuration must be updated manually to use the new partition information.

For more information about partition migration, see Exporting and Importing Partitions in *Oracle Fusion Middleware Using WebLogic Server MT*.

If you make any changes in the partition on the Oracle WebLogic Server side, you must make corresponding changes in the plug-in configuration. On the Oracle WebLogic Server side you can typically change the VirtualHost parameters (host and port) and the optional URI. In this case, you must edit the ServerName and PathPrepend parameters in the plug-in configuration.

If you add, delete, or migrate managed servers on the Oracle WebLogic Server side, you must also make corresponding changes in the plug-in configuration.

For example, assume you have the following Oracle HTTP Server (or Apache) plug-in configuration:

```
<VirtualHost *:8080>
<IfModule mod_weblogic.c>
ServerName server1
WebLogicCluster wls1:7001,wls2:7001,wls3:7001
SetHandler weblogic-handler
</IfModule>
</VirtualHost>
```

If you add managed server wls4 to the partition, Then add wls4 to the WebLogicCluster parameter:

```
<VirtualHost *:8080>
<IfModule mod_weblogic.c>
ServerName server1
WebLogicCluster wls1:7001,wls2:7001,wls3:7001,wls4:7001
SetHandler weblogic-handler
</IfModule>
</VirtualHost>
```

If you delete managed server `wls1` from the partition, Then delete `wls1` from the `WebLogicCluster` parameter:

```
<VirtualHost *:8080>
<IfModule mod_weblogic.c>
ServerName server1
WebLogicCluster wls2:7001,wls3:7001,wls4:7001
SetHandler weblogic-handler
</IfModule>
</VirtualHost>
```

If you migrate managed server `wls2` to `wls5` then remove `wls2` and add `wls5` to the `WebLogicCluster` parameter:

```
<VirtualHost *:8080>
<IfModule mod_weblogic.c>
ServerName server1
WebLogicCluster wls3:7001,wls4:7001,wls5:7001
SetHandler weblogic-handler
</IfModule>
```

For information about modifying a partition of the Oracle WebLogic Server side, see Configuring Domain Partitions, Configuring Virtual Targets, and Exporting and Importing Partitions in *Oracle Fusion Middleware Using WebLogic Server MT*.

7.4.3 Configuring SSL Between the Web Server and Oracle WebLogic Server

You configure SSL on Oracle WebLogic Server MT in the same way as with Oracle WebLogic Server, however, the partitions that are targeted to the same host name should have same certificate.

7.4.4 Dynamic Discovery of Cluster Changes

The dynamic discovery of cluster changes is a feature of both Oracle WebLogic Server Proxy Plug-In and Oracle WebLogic Server. It is not affected by Multitenancy. This feature can be controlled by the `DynamicServerList` parameter. See DynamicServerList.

8

Parameters for Web Server Plug-Ins

This chapter describes the parameters that you can use to configure the Oracle HTTP Server, Apache HTTP Server, Microsoft IIS, and Oracle iPlanet Web Server plug-ins. It contains the following sections:

- General Parameters for Web Server Plug-Ins
- SSL Parameters for Web Server Plug-Ins
- Location of POST Data Files

> ✎ **Note:**
>
> The parameters for the web-server plug-ins should be specified in special configuration files, which are named and formatted uniquely for each web server. For information about the configuration files specific to the plug-ins for Apache HTTP Server, Oracle HTTP Server, Microsoft IIS, and Oracle iPlanet Web Server, see the following chapters:
>
> - Configuring the Plug-In for Oracle HTTP Server
> - Configuring the Plug-In for Apache HTTP Server
> - Configuring the Plug-In for iPlanet Web Server
> - Configuring the Plug-In for Microsoft IIS Web Server

8.1 General Parameters for Web Server Plug-Ins

The general parameters for Web server plug-ins are described in the following sections. The parameter names are case sensitive.

- ConnectRetrySecs
- ConnectTimeoutSecs
- Debug
- DebugConfigInfo
- DefaultFileName
- DynamicServerList
- ErrorPage
- FileCaching
- Idempotent
- KeepAliveEnabled
- KeepAliveSecs

- MatchExpression
- MaxPostSize
- MaxSkipTime
- PathPrepend
- PathTrim
- QueryFromRequest
- WebLogicCluster
- WebLogicHost
- WebLogicPort
- WLCookieName
- WLDNSRefreshInterval
- WLExcludePathOrMimeType
- WLFlushChunks
- WLForwardUriUnparsed
- WLIOTimeoutSecs
- WLLocalIP
- WLLogFile
- WLMaxWebSocketClients
- WLProxyPassThrough
- WLProxySSL
- WLProxySSLPassThrough
- WLRetryOnTimeout
- WLRetryAfterDroppedConnection
- WLSendHdrSeparately
- WLServerInitiatedFailover
- WLSocketTimeoutSecs
- WLSRequest
- WLTempDir

8.1.1 ConnectRetrySecs

Default: 2

Applies to: Oracle HTTP Server, Oracle iPlanet Web Server, Apache HTTP Server, Microsoft IIS

Interval in seconds that the plug-in should sleep between attempts to connect to the WebLogic Server host (or all of the servers in a cluster). Make this number less than the `ConnectTimeoutSecs`. The number of times the plug-in tries to connect before returning an `HTTP 503/Service Unavailable` response to the client is calculated by dividing `ConnectTimeoutSecs` by `ConnectRetrySecs`.

To specify no retries, set `ConnectRetrySecs` equal to `ConnectTimeoutSecs`. However, the plug-in attempts to connect at least twice.

You can customize the error response by using the `ErrorPage` parameter.

8.1.2 ConnectTimeoutSecs

Default: 10

Applies to: Oracle HTTP Server, Oracle iPlanet Web Server, Apache HTTP Server, Microsoft IIS

Maximum time in seconds that the plug-in should attempt to connect to the WebLogic Server host. Make the value greater than `ConnectRetrySecs`. If `ConnectTimeoutSecs` expires without a successful connection, even after the appropriate retries (see `ConnectRetrySecs`), an `HTTP 503/Service Unavailable` response is sent to the client.

You can customize the error response by using the `ErrorPage` parameter.

8.1.3 Debug

Default: OFF

Applies to: Microsoft IIS

Sets the type of logging performed for debugging operations. The debugging information is written to c:\TEMP\wlproxy.log on Windows NT/2000 systems.

Override this location and filename by setting the `WLLogFile` parameter to a different directory and file. (See the `WLTempDir` parameter for an additional way to change this location.)

Ensure that the directory of the log file has write permission. Set any of the following logging options (HFC,HTW,HFW, and HTC options may be set in combination by entering them separated by commas, for example "HFC,HTW"):

- `ON`: The plug-in logs informational and error messages.

- `OFF`: No debugging information is logged.

- `HFC`: The plug-in logs headers from the client, informational, and error messages.

- `HTW`: The plug-in logs headers sent to WebLogic Server, and informational and error messages.

- `HFW`: The plug-in logs headers sent from WebLogic Server, and informational and error messages.

- `HTC`: The plug-in logs headers sent to the client, informational messages, and error messages.

- `ERR`: Prints only the Error messages in the plug-in.

- `ALL`: The plug-in logs headers sent to and from the client, headers sent to and from WebLogic Server, information messages, and error messages.

For information on setting logging without using the deprecated parameter, see Deprecated Directives for Apache HTTP Server.

8.1.4 DebugConfigInfo

Default: OFF

Applies to: Oracle HTTP Server, Oracle iPlanet Web Server, Apache HTTP Server, Microsoft IIS

Enables the special query parameter "__WebLogicBridgeConfig". Use it to get details about configuration parameters from the plug-in.

For example, if you enable "__WebLogicBridgeConfig" by setting `DebugConfigInfo` and then send a request that includes the query string ?__WebLogicBridgeConfig, then the plug-in gathers the configuration information and run-time statistics and returns the information to the browser. The plug-in does not connect to WebLogic Server in this case.

This parameter is strictly for debugging and the format of the output message can change with releases. For security purposes, keep this parameter turned OFF in production systems.

8.1.5 DefaultFileName

Default: none

Applies to: Oracle HTTP Server, Oracle iPlanet Web Server, Apache HTTP Server, Microsoft IIS

If the URI is "/" then the plug-in performs the following steps:

1. Trims the path specified with the `PathTrim` parameter.

2. Appends the value of `DefaultFileName`.

3. Prepends the value specified with `PathPrepend`.

This procedure prevents redirects from WebLogic Server.

Set the DefaultFileName to the default welcome page of the Web Application in WebLogic Server to which requests are being proxied. For example, If the `DefaultFileName` is set to welcome.html, an HTTP request like "http://somehost/weblogic" becomes "http://somehost/weblogic/welcome.html". For this parameter to function, the same file must be specified as a welcome file in all the Web Applications to which requests are directed. See Configuring Welcome Pages.

Note for Apache users: If you are using Stronghold or Raven versions, define this parameter inside of a `Location` block, and not in an `IfModule` block.

8.1.6 DynamicServerList

Default: ON

Applies to: Oracle HTTP Server, Oracle iPlanet Web Server, Apache HTTP Server, Microsoft IIS

When set to `OFF,` the plug-in ignores the dynamic cluster list used for load balancing requests proxied from the plug-in and only uses the static list specified with the `WebLogicCluster` parameter. Normally this parameter should remain set to `ON`.

There are some implications for setting this parameter to `OFF`:

- If one or more servers in the static list fails, the plug-in could waste time trying to connect to a terminated server, resulting in decreased performance.

- If you add a new server to the cluster, the plug-in cannot proxy requests to the new server unless you redefine this parameter. WebLogic Server automatically adds new servers to the dynamic server list when they become part of the cluster.

8.1.7 ErrorPage

Default: none

Applies to: Oracle HTTP Server, Oracle iPlanet Web Server, Apache HTTP Server, Microsoft IIS

You can create your own error page that is displayed when your Web server cannot forward requests to WebLogic Server.

The plug-in redirects to an error page when the back-end server returns an HTTP 503/Service Unavailable response and there are no servers for failover.

8.1.8 FileCaching

Default: ON

Applies to: Oracle HTTP Server, Oracle iPlanet Web Server, Apache HTTP Server, Microsoft IIS

When set to `ON`, and the size of the POST data in a request is greater than 2048 bytes, the POST data is first read into a temporary file on disk and then forwarded to the WebLogic Server in chunks of 8192 bytes. This preserves the POST data during failover, allowing all necessary data to be repeated to the secondary if the primary goes down.

When `FileCaching` is `ON`, any client that tracks the progress of the POST will see that the transfer has completed even though the data is still being transferred between the WebServer and WebLogic. So, if you want the progress bar displayed by a browser during the upload to reflect when the data is actually available on the WebLogic Server, you might not want to have `FileCaching ON`.

When set to `OFF` and the size of the POST data in a request is greater than 2048 bytes, the reading of the POST data is postponed until a WebLogic Server cluster member is identified to serve the request. Then the plug-in reads and immediately sends the POST data to the WebLogic Server in chunks of 8192 bytes.

Turning `FileCaching OFF` limits failover. If the WebLogic Server primary server goes down while processing the request, the POST data already sent to the primary cannot be repeated to the secondary.

Finally, regardless of how `FileCaching` is set, if the size of the POST data is 2048 bytes or less the plug-in will read the data into memory and use it if needed during failover to repeat to the secondary.

8.1.9 Idempotent

Default: ON

Applies to: Oracle HTTP Server, Oracle iPlanet Web Server, Apache HTTP Server, Microsoft IIS

When set to ON and if the servers do not respond within WLIOTimeoutSecs, the plug-ins fail over if the method is Idempotent.

The plug-ins also fail over if Idempotent is set to ON and the servers respond with an error such as `READ_ERROR_FROM_SERVER`.

If Idempotent is set to OFF, the plug-ins do not fail over. If you are using the Apache HTTP Server, you can set this parameter differently for different URLs or MIME types.

Idempotent only takes effect if the request is successfully sent to the WebLogic Server and the plug-in is now waiting for a response from the back end server.

POST requests are not retried even if marked as Idempotent.

8.1.10 KeepAliveEnabled

Default: true (Microsoft IIS plug-in), ON (Oracle HTTP Server and Apache HTTP Server), ON (Oracle iPlanet Web Server)

Applies to: Oracle HTTP Server, Oracle iPlanet Web Server, Apache HTTP Server, Microsoft IIS

Enables pooling of connections between the plug-in and WebLogic Server.

- Valid values for the Microsoft IIS plug-ins are true and false.
- Valid values for the Apache HTTP Server are `ON` and `OFF`.

  While using Apache prefork mpm, Apache web server might fail. Set `KeepAliveEnabled` to OFF when using prefork mpm or use worker mpm in Apache.

- Valid values for Oracle iPlanet Webserver are `ON` and `OFF`

8.1.11 KeepAliveSecs

Default: 20

Applies to: Oracle HTTP Server, Oracle iPlanet Web Server, Apache HTTP Server, Microsoft IIS

The length of time after which an inactive connection between the plug-in and WebLogic Server is closed. You must set `KeepAliveEnabled` to true (ON when using the Apache HTTP Server) for this parameter to be effective.

The value of this parameter must be less than or equal to the value of the Duration field set in the Administration Console on the Server/HTTP tab, or the value set on the server Mbean with the `KeepAliveSecs` attribute.

8.1.12 MatchExpression

Default: none

Applies to: Oracle HTTP Server, Apache HTTP Server

Use this parameter to modify the values of existing parameters or add a new parameter for a particular configuration.

The MatchExpression parameter supports only the `*` and `?` regular expressions

- `*` which matches 0 or more characters
- `?` which matches exactly one character

This parameter can be configured for two scenarios.

Proxying by MIME type:

You can use this parameter in the following format to set other parameters for a particular MIME type.

Syntax:

```
MatchExpression <file_extension> <param=value>|<param-value>|…
```

For example, the following configuration proxies `*.jsp` to `myHost:8080`:

```
<IfModule weblogic_module>
MatchExpression *.jsp WebLogicHost=myHost|WebLogicPort=8080
</IfModule>
```

Proxying by path:

You can also use this parameter in the following format to set other parameters for a particular path.

Syntax:

```
MatchExpression <path> <param=value>|<param-value>|…
```

For example, the following configuration proxies the URIs beginning with `/weblogic` to `myHost:9090`:

```
<IfModule weblogic_module>
MatchExpression /weblogic WebLogicHost=myHost|WebLogicPort=9090
</IfModule>
```

You can also use `MatchExpression` to override the parameter values, as shown above. It can also be used to define new parameters (this is, those that have not been used in the configuration).

For example, the configuration below proxies all the requests to my`Host:8080`. The URIs that match the type `jpg` will be proxied to `myHost:8080/images` and others will be proxied to my`Host:8080`.

```
<IfModule weblogic_module>
SetHandler weblogic-handler
WebLogicHost myHost
WebLogicPort 8080
MatchExpression *.jpg PathPrepend=/images
</IfModule>
```

You can find more examples of how to use MatchExpression in Configuring the Plug-In for Apache HTTP Server.

8.1.13 MaxPostSize

Default: 0

Applies to: Oracle HTTP Server, Oracle iPlanet Web Server, Apache HTTP Server, Microsoft IIS

Maximum allowable size of POST data, in bytes. If the content-length exceeds `MaxPostSize`, the plug-in returns an error message. If set to 0, the size of POST data is not checked. This is useful for preventing denial-of-service attacks that attempt to overload the server with POST data.

8.1.14 MaxSkipTime

Default: 10

Applies to: Oracle HTTP Server, Oracle iPlanet Web Server, Apache HTTP Server, Microsoft IIS

If a WebLogic Server listed in either the `WebLogicCluster` parameter or a dynamic cluster list returned from WebLogic Server fails, the failed server is marked as "bad" and the plug-in attempts to connect to the next server in the list.

`MaxSkipTime` sets the amount of time after which the plug-in will retry the server marked as "bad." The plug-in attempts to connect to a new server in the list each time a unique request is received (that is, a request without a cookie).

8.1.15 PathPrepend

Default: null

Applies to: Oracle HTTP Server, Oracle iPlanet Web Server, Apache HTTP Server, Microsoft IIS

As per the RFC specification, generic syntax for URL is:

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

`PathPrepend` specifies the path that the plug-in prepends to the `{PATH}` portion of the original URL, after PathTrim is trimmed and before the request is forwarded to WebLogic Server.

If you must append a File Name, use `DefaultFileName` parameter instead of `PathPrepend`.

8.1.16 PathTrim

Default: null

Applies to: Oracle HTTP Server, Oracle iPlanet Web Server, Apache HTTP Server, Microsoft IIS

As per the RFC specification, generic syntax for URL is:

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

`PathTrim` specifies the string trimmed by the plug-in from the `{PATH}/{FILENAME}` portion of the original URL, before the request is forwarded to WebLogic Server. For example, if this URL:

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

is passed to the plug-in for parsing and if `PathTrim` 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:7001/foo

If you are newly converting an existing third-party server to proxy requests to WebLogic Server using the plug-in, you will need to change application paths to `/foo` to include `weblogic/foo`. You can use `PathTrim` and `PathPrepend` in combination to change this path.

8.1.17 QueryFromRequest

Default: OFF

Applies to: Oracle HTTP Server, Apache HTTP Server

When set to `ON`, specifies that the Apache HTTP Server use

```
(request_rec *)r->the_request
```

to pass the query string to WebLogic Server. (For more information, see the Apache documentation.) This behavior is desirable when a Netscape version 4.x browser makes requests that contain spaces in the query string

When set to `OFF`, the Apache HTTP Server uses `(request_rec *)r->args` to pass the query string to WebLogic Server.

8.1.18 WebLogicCluster

Required when proxying to a cluster of WebLogic Servers, or to multiple non-clustered servers.

Default: none

Applies to: Oracle HTTP Server, Oracle iPlanet Web Server, Apache HTTP Server, Microsoft IIS

The `WebLogicCluster` parameter is required to proxy a list of back-end servers that are clustered, or to perform load balancing among non-clustered managed server instances.

List of WebLogic Servers that can be used for load balancing. 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 syntax for specifying the value of this parameter varies depending on the web server for which you are configuring the plug-in. See the following topics:

- Configuring the Plug-In for Oracle HTTP Server
- Configuring the Plug-In for Apache HTTP Server
- Configuring the Plug-In for iPlanet Web Server
- Configuring the Plug-In for Microsoft IIS Web Server

If you are using SSL between the plug-in and WebLogic Server, set the port number to the SSL listen port and set the SecureProxy parameter to ON.

The plug-in 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 plug-in maintain. WebLogic Server and the plug-in 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 setting the DynamicServerList parameter to `OFF`.

The plug-in directs HTTP requests containing a cookie, URL-encoded session, or a session stored in the POST data to the server in the cluster that created the cookie.

8.1.19 WebLogicHost

Required when proxying to a single WebLogic Server.

Default: none

Applies to: Oracle HTTP Server, Oracle iPlanet Web Server, Apache HTTP Server, Microsoft IIS

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 `WebLogicCluster` parameter instead of `WebLogicHost`.

8.1.20 WebLogicPort

Required when proxying to a single WebLogic Server.

Default: none

Applies to: Oracle HTTP Server, Oracle iPlanet Web Server, Apache HTTP Server, Microsoft IIS

Port at which the WebLogic Server host is listening for connection requests from the plug-in (or from other servers). (If you are using SSL between the plug-in and WebLogic Server, set this parameter to the SSL listen port and set the `SecureProxy` parameter to ON).

If you are using a WebLogic Cluster, use the `WebLogicCluster` parameter instead of `WebLogicPort`.

8.1.21 WLCookieName

Default: JSESSIONID

Applies to: Oracle HTTP Server, Oracle iPlanet Web Server, Apache HTTP Server, Microsoft IIS

If you change the name of the WebLogic Server session cookie in the WebLogic Server Web application, then you must change the `WLCookieName` parameter in the plug-in to the same value. The name of the WebLogic session cookie is set in the WebLogic-specific deployment descriptor, in the `<session-descriptor>` element in `weblogic.xml`.

8.1.22 WLDNSRefreshInterval

Default: 0 (Lookup once, during startup)

Applies to: Oracle HTTP Server, Oracle iPlanet Web Server, Apache HTTP Server

If defined in the proxy configuration, specifies number of seconds interval at which WebLogic Server refreshes DNS name to IP mapping for a server. This can be used if a WebLogic Server instance is migrated to a different IP address, but the DNS name for that server's IP remains the same. In this case, at the specified refresh interval the DNS<->IP mapping will be updated.

8.1.23 WLExcludePathOrMimeType

Default: none

Applies to: Oracle HTTP Server, Oracle iPlanet Web Server, Apache HTTP Server, Microsoft IIS

This parameter allows you to exclude certain requests from proxying.

This parameter can be defined locally at the Location tag level and globally. When the property is defined locally, it does not override the global property but defines a union of the two parameters.

8.1.24 WLFlushChunks

Default: False

Applies to: Microsoft IIS

By default, IIS plug-in buffers chunked transfer encoding responses instead of streaming the chunks as they are received. When the flag `WLFlushChunks` is set to true, the plug-in flushes chunks immediately as they are received from WebLogic Server.

8.1.25 WLForwardUriUnparsed

Default: OFF

Applies to: Oracle HTTP Server, Apache HTTP Server

When set to ON, the WLS plug-in will forward the original URI from the client to WebLogic Server. When set to OFF (default), the URI sent to WebLogic Server is subject to modification by mod_rewrite or other web server plug-in modules.

8.1.26 WLIOTimeoutSecs

New name for `HungServerRecoverSecs`.

Default: 300

Applies to: Oracle HTTP Server, Oracle iPlanet Web Server, Apache HTTP Server, Microsoft IIS

Defines the amount of time the plug-in waits for a response to a request from WebLogic Server. The plug-in waits for `WLIOTimeoutSecs` for the server to respond and then declares that server dead, and fails over to the next server. The value should be set to a very large value. If the value is less than the time the servlets take to process, then you may see unexpected results.

Minimum value: 10

Maximum value: Unlimited

8.1.27 WLLocalIP

Default: none

Applies to: Oracle HTTP Server, Oracle iPlanet Web Server, Apache HTTP Server, Microsoft IIS

Defines the IP address (on the plug-in's system) to bind to when the plug-in connects to a WebLogic Server instance running on a multihomed machine.

If `WLLocalIP` is not set, then the TCP/IP stack will choose the source IP address.

8.1.28 WLLogFile

Default: See the `Debug` parameter

Applies to: Microsoft IIS

Specifies 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.

For information on setting logging without using the deprecated parameter, see Deprecated Directives for Apache HTTP Server.

8.1.29 WLMaxWebSocketClients

Default: Windows: Half of **ThreadsPerChild**, **Non-Windows:** Half of **MaxRequestWorkers** (or **MaxClients** for Apache 2.2)

Applies to: Oracle HTTP Server, Apache HTTP Server

Limits the number of active WebSocket connections at any instant of time.

> **✎ Note:**
>
> The maximum value you can set for this parameter is 75 percent of `ThreadsPerChild` (Windows) or 75 percent of `MaxRequestWorkers` (or `MaxClients` for Apache 2.2, non-Windows). If the value specified for this parameter is greater than the maximum allowed, it will be automatically lowered to that maximum.

8.1.30 WLProxyPassThrough

Default: OFF

Applies to: Oracle HTTP Server, Oracle iPlanet Web Server, Apache HTTP Server, Microsoft IIS

If you have a chained proxy setup, where a proxy plug-in or `HttpClusterServlet` is running behind some other proxy or load balancer, you must explicitly enable the `WLProxyPassThrough` parameter. This parameter allows the header to be passed through the chain of proxies.

8.1.31 WLProxySSL

Default: OFF

Applies to: Oracle HTTP Server, Oracle iPlanet Web Server, Apache HTTP Server, Microsoft IIS

Set this parameter to `ON` to maintain SSL communication between the plug-in and WebLogic Server when the following conditions exist:

- An HTTP client request specifies the HTTPS protocol
- The request is passed through one or more proxy servers (including the Oracle WebLogic Server Proxy Plug-In)
- The connection between the plug-in and WebLogic Server uses the HTTP protocol

When `WLProxySSL` is set to `ON`, the location header returned to the client from WebLogic Server specifies the HTTPS protocol.

8.1.32 WLProxySSLPassThrough

Default: OFF

Applies to: Oracle HTTP Server, Oracle iPlanet Web Server, Apache HTTP Server, Microsoft IIS

If a load balancer or other software deployed in front of the web server and plug-in is the SSL termination point, and that product sets the WL-Proxy-SSL request header to true or false based on whether the client connected to it over SSL, set `WLProxySSLPassThrough` to ON so that the use of SSL is passed on to the Oracle WebLogic Server.

If the SSL termination point is in the web server where the plug-in operates, or the load balancer does not set WL-Proxy-SSL, set `WLProxySSLPassThrough` to OFF (default).

8.1.33 WLRetryOnTimeout

Default: None

Applies to: Oracle HTTP Server, Apache HTTP Server

Tells the WebLogic plug-in whether to retry requests (including POST requests) when a time-out occurs before the WebLogic server sends the status line. Valid arguments are:

- `ALL`: All requests are retried.
- `IDEMPOTENT`: Only requests that use idempotent methods are retried.
- `NONE`: No requests are retried.

8.1.34 WLRetryAfterDroppedConnection

Default: ALL

Applies to: Oracle HTTP Server, Oracle iPlanet Web Server, Apache HTTP Server, Microsoft IIS

Tells the Apache plug-in which requests to retry when a connection is lost before WLS sends the status line. Valid arguments are:

- `ALL`: All requests will be retried.

- `IDEMPOTENT`: Only requests using idempotent methods will be retried.

- `NONE`: No requests will be retried.

8.1.35 WLSendHdrSeparately

Default: ON

Applies to: Microsoft IIS

When this parameter is set to ON, the header and body of the response are sent in separate packets.

> **✎ Note:**
>
> If you must send the header and body of the response in two calls, for example, in cases where you have other ISAPI filters or programmatic clients that expect headers before the body, set this parameter to ON.

8.1.36 WLSServerInitiatedFailover

Default: ON

Applies to: Oracle HTTP Server, Oracle iPlanet Web Server, Apache HTTP Server, Microsoft IIS

This controls whether a 503 error response from Oracle WebLogic Server triggers a failover to another server. Normally, the plug-in will attempt to failover to another server when a 503 error response is received. When `WLServerInitiatedFailover` is set to OFF, the 503 error response will be returned to the client immediately.

8.1.37 WLSocketTimeoutSecs

Default: 2 (must be greater than 0)

Applies to: Oracle HTTP Server, Oracle iPlanet Web Server, Apache HTTP Server, Microsoft IIS

Set the timeout for the socket while connecting, in seconds. See `ConnectTimeoutSecs` and `ConnectRetrySecs` for additional details.

8.1.38 WLSRequest

Default: OFF

Applies to: Oracle HTTP Server, Apache HTTP Server

This is an alternative to the WLSRequest On mechanism of identifying requests to be forwarded to Oracle WebLogic Server. For example,

```
<Location /weblogic>
 WLSRequest ON
 PathTrim /weblogic
</Location>
```

The use of `WLSRequest` ON instead of `SetHandler` weblogic-handler has the following advantages:

- Lower web server processing overhead in general

- Resolves substantial performance degradation when the web server `DocumentRoot` is on a slow filesystem

- Resolves 403 errors for URIs which cannot be mapped to the filesystem due to the filesystem length restrictions

8.1.39 WLTempDir

Default: See the `Debug` parameter

Applies to: Oracle HTTP Server, Oracle iPlanet Web Server, Apache HTTP Server, Microsoft IIS

For Oracle HTTP Server, Apache HTTP Server and Oracle iPlanet HTTP Server, this directive specifies the location of the _wl_proxy directory for POST data files.

For Microsoft IIS, this directive specifies the directory where a wlproxy.log will be created. If the location fails, the Plug-In resorts to creating the log file under C:/temp. It also specifies the location of the _wl_proxy directory for POST data files. When both `WLTempDir` and `WLLogFile` are set, `WLLogFile` will override the location of wlproxy.log. `WLTempDir` will still determine the location of _wl_proxy directory.

8.2 SSL Parameters for Web Server Plug-Ins

> **Note:**
>
> SCG Certificates are not supported for use with Oracle WebLogic Server Proxy Plug-In. Non-SCG certificates work appropriately and allow SSL communication between WebLogic Server and the plug-in.
>
> KeyStore-related initialization parameters are not supported for use with Oracle WebLogic Server Proxy Plug-In.

The SSL parameters for Web Server plug-ins are described in the following sections. Parameters are case sensitive.

- SecureProxy
- WebLogicSSLVersion
- WLSSLWallet

8.2.1 SecureProxy

Default: OFF

Applies to: Oracle HTTP Server, Oracle iPlanet Web Server, Apache HTTP Server, Microsoft IIS

Set this parameter to ON to enable the use of the SSL protocol for all communication between the plug-in and WebLogic Server. Remember to configure a port on the corresponding WebLogic Server for the SSL protocol before defining this parameter.

This parameter may be set at two levels: in the configuration for the main server and—if you have defined any virtual hosts—in the configuration for the virtual host. The configuration for the virtual host inherits the SSL configuration from the configuration of the main server if the setting is not overridden in the configuration for the virtual host.

8.2.2 WebLogicSSLVersion

Default: The best protocol supported by both the plug-in and WebLogic Server.

Applies to: Oracle HTTP Server, Apache HTTP Server

Specifies the SSL protocol version to use for communication between the plug-in and the WebLogic Server. This setting does not need to match that of the web server's `ssl.conf` file. Plug-in can have its own SSL version to communicate with WebLogic Server.

The following values are accepted:

- `TLSv1`: Uses TLS v1.0

- `TLSv1_1`: Uses TLS v1.1

- `TLSv1_2`: Uses TLS v1.2

For example:

```
WebLogicSSLVersion TLSv1_1 TLSv1_2
```

You can define multiple protocols by using a space-separated list. The SSL protocol version chosen is used for all the connections from the plug-in to WebLogic Server. Hence define this parameter at the global scope.

If not configured, the plug-in uses the best protocol supported by both the plug-in and WebLogic Server.

8.2.3 WLSSLWallet

Default: none

Applies to: Oracle HTTP Server, Oracle iPlanet Web Server, Apache HTTP Server, Microsoft IIS

`WLSSLWallet` performs one-way or two-way SSL based on how SSL is configured for Oracle WebLogic Server.

Requires the path of an Oracle Wallet (containing an SSO wallet file) as an argument. For example:

```
WLSSLWallet "${ORACLE_INSTANCE}/config/fmwconfig/components/${COMPONENT_TYPE}/
instances/${COMPONENT_NAME}/keystores/default"
```

8.3 Location of POST Data Files

When the FileCaching parameter is set to ON, and the size of the POST data in a request is greater than 2048 bytes, the POST data is first read into a temporary file on disk and then forwarded to the WebLogic Server in chunks of 8192 bytes. This preserves the POST data during failover.

The temporary POST file is located under `/tmp/_wl_proxy` for UNIX. For Windows it is located as follows (if `WLTempDir` is not specified):

1. Environment variable `TMP`

2. Environment variable `TEMP`

3. `C:\Temp`

`/tmp/_wl_proxy` is a fixed directory and is owned by the HTTP Server user. When there are multiple HTTP Servers installed by different users, some HTTP Servers might not be able to write to this directory. This condition results in an error.

To correct this condition, use the `WLTempDir` parameter to specify a different location for the `_wl_proxy` directory for POST data files.