Oracle® Fusion Middleware

Using Oracle WebLogic Server Proxy Plug-Ins

12*c* (12.1.3)

E48286-08

November 2016

This document describes the use of version 12.1.3 plug-ins provided for proxying requests from web servers to Oracle WebLogic Server. This document is intended mainly for system administrators who manage the WebLogic Server application platform and its various subsystems.

ORACLE®

Oracle Fusion Middleware Using Oracle WebLogic Server Proxy Plug-Ins, 12*c* (12.1.3)

E48286-08

Contents

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 Common Configuration Tasks

7 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 12c*.

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 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-Ins 12.1.3

The following sections describe the plug-ins provided by Oracle for use with WebLogic Server:

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. For more information, see `KeepAliveEnabled` in Table 7–1.

> **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 12.1.3

Version 12.1.3 plug-ins are available for the following web servers:

Table 1–1   Availability of Version 12.1.3 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 Chapter 2, "Configuring the Plug-In for Oracle HTTP Server"	See Chapter 2, "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) 8.0 and 8.5	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. For the Windows version, DLL files are provided. - 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, Microsoft IIS, and Oracle iPlanet Web Server, see the following: - Chapter 4, "Configuring the Plug-In for iPlanet Web Server" - Chapter 3, "Configuring the Plug-In for Apache HTTP Server" - Chapter 5, "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 12.1.3 plug-ins are the recommended replacement.

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

1.3.1 Upgrade Instructions

For upgrading from 11g plug-ins to the Oracle WebLogic Server Proxy Plug-Ins 12.1.3, 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 12.1.3 plug-in for:	See:
Oracle HTTP Server	Chapter 2, "Configuring the Plug-In for Oracle HTTP Server"
Apache HTTP Server	Chapter 3, "Configuring the Plug-In for Apache HTTP Server"
iPlanet Web Server	Chapter 4, "Configuring the Plug-In for iPlanet Web Server"
Microsoft IIS Web Server	Chapter 5, "Configuring the Plug-In for Microsoft IIS Web Server"

1.3.2 Considerations for Upgrading to Oracle WebLogic Server Proxy Plug-Ins 12.1.3

The version 12.1.3 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. For more information, see the *Oracle Fusion Middleware Supported System Configurations* at:

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

- If you have been using 128-bit encryption, you must change your configuration file to reflect the new naming convention, as described in Section 1.4.1, "Standard Encryption Strength Allows Simplified Naming". For example, you must change mod_wl128_22.so to mod_wl.so.

1.4 Features of the Version 12.1.3 Plug-Ins

This section describes the additional features of the version 12.1.3 plug-ins when compared with the 1.0 plug-ins.

- Section 1.4.1, "Standard Encryption Strength Allows Simplified Naming"

- Section 1.4.2, "Version 12.1.3 Plug-Ins Use Oracle SSL/Networking"

- Section 1.4.3, "Version 12.1.3 Plug-Ins Support IPv6"

- Section 1.4.4, "Version 12.1.3 Plug-Ins Support Two-Way SSL"

- Section 1.4.5, "Version 12.1.3 Plug-Ins Support Proxying WebSocket Upgrade Requests"

1.4.1 Standard Encryption Strength Allows Simplified Naming

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 12.1.3 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.

1.4.2 Version 12.1.3 Plug-Ins Use Oracle SSL/Networking

The version 12.1.3 plug-ins use the Oracle SSL/Networking implementation (as opposed to Certicom stack that is used in Plug-in 1.0) and can therefore use Oracle wallets to store SSL configuration information.

For this reason, the version 12.1.3 plug-ins introduce an SSL configuration parameter WLSSLWallet to use Oracle wallets. For more information on creating and managing Oracle wallets, see Managing Wallets in *Oracle Fusion Middleware Administrator's Guide*.

You can configure the certificates in the Oracle wallet with a command line tool that is provided with the plug-in binary files. See Section 6.1, "Use SSL with Plug-Ins" for information about configuring SSL.

WebLogic Server Plug-Ins for third party web servers use the same SSL technology as the WebLogic Server Plug-In for Oracle HTTP Server because Oracle HTTP Server uses Oracle SSL/Networking and Oracle wallet technologies. For example, Apache HTTP Server's mod_wl module uses the same SSL as Oracle HTTP Server's mod_wl_ohs module. Therefore, SSL configuration and processing is the same for both.

1.4.3 Version 12.1.3 Plug-Ins Support IPv6

The version 12.1.3 plug-ins support IPv6. The WebLogicHost and WebLogicCluster configuration parameters (see Table 7–1) now support IPv6 addresses.

For more information, see Section 6.2, "Use IPv6 With Plug-Ins."

1.4.4 Version 12.1.3 Plug-Ins Support Two-Way SSL

The version 12.1.3 plug-ins provide two-way SSL support for verifying client identity. Two-way SSL is automatically enforced when WebLogic Server requests the client certificate during the handshake process.

For more information, see Section 6.1, "Use SSL with Plug-Ins."

1.4.5 Version 12.1.3 Plug-Ins Support Proxying WebSocket Upgrade Requests

Version 12.1.3 of both the Oracle HTTP Server and Apache HTTP Server plug-ins supports proxying WebSocket upgrade requests.

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 will look like this:

WebLogic Server Plug-in version 12.1.3 <WLSPLUGINS_XXXX_XXXX_XXXXX.XXXX>

> **Note:** On the Apache Web Server for Linux, You can also obtain the plugin 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. Be sure 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

This chapter describes how to configure the Oracle WebLogic Server Proxy Plug-In (mod_wl_ohs), 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 12.1.3 installation. You need not 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.

This chapter contains the following topics:

- Section 2.1, "Support Note"
- Section 2.2, "Prerequisites for Configuring the Plug-In"
- Section 2.3, "Configuring the Plug-In Using Fusion Middleware Control"
- Section 2.4, "Configuring the Plug-In Manually"
- Section 2.5, "Deprecated Directives for Oracle HTTP Server"
- Section 2.6, "Troubleshooting Oracle WebLogic Server Proxy Plug-In Implementations"

2.1 Support Note

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

2.2 Prerequisites for Configuring the Plug-In

Before you begin configuring the Oracle WebLogic Server Proxy Plug-In, do the following:

- 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 if you want configure the Oracle

WebLogic Server Proxy Plug-In by using the graphical interface provided by Fusion Middleware Control.

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

- If the version of the Oracle WebLogic Server instances in the back end is 10.3.4 (or later releases), you must set the WebLogic Plug-In Enabled parameter.

  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.

     The Configuration: General tab is displayed.

  4. 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. For more information, see Section 2.2.1, "Understanding the WebLogic Plug-In Enabled Parameter."

  5. 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.

  6. Click **Save**.

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

2.2.1 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.

- **Inherit**—When **Inherit** 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 **Inherit** 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 Plug-In Using Fusion Middleware Control

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

1. Ensure you have fulfilled the prerequisites listed in Section 2.2.

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.
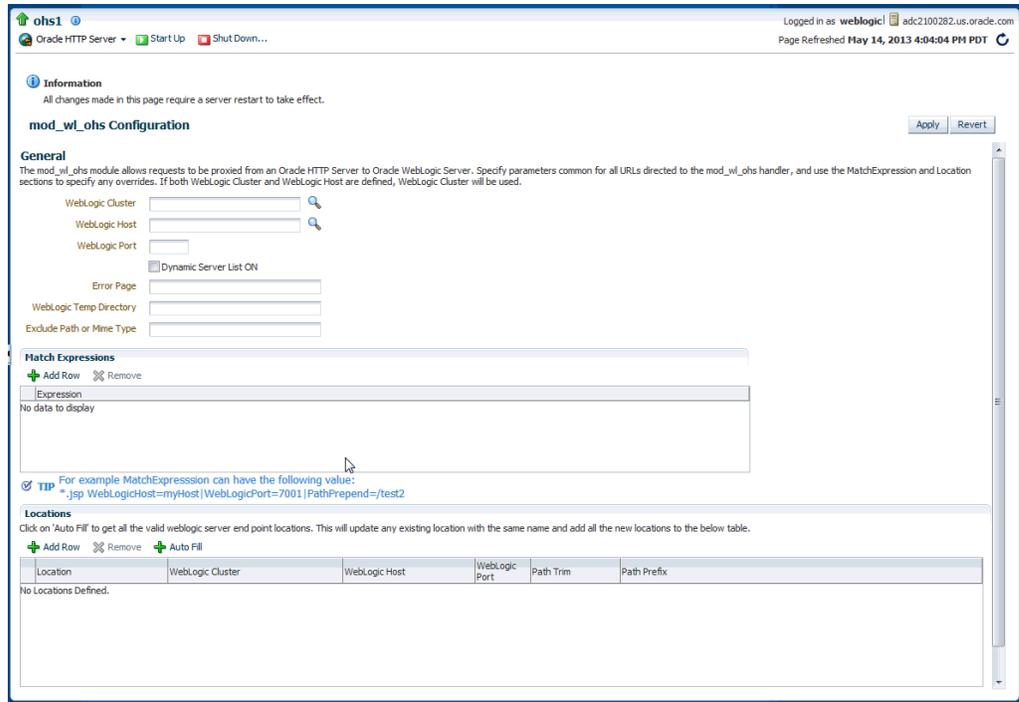
4. Specify the configuration settings as described in the following table:

Field	Description
WebLogic Cluster	List of Oracle 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.
	If you are not sure if the correct cluster, you can click the search icon to see a list of all associated clusters. For more information, see Section 2.3.1, "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 created the cookie.
	Note: WebLogic Cluster and WebLogic Host are mutually-exclusive fields; you should only specify one. If you provide a value for both fields, WebLogic Cluster takes precedence.

Field	Description
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 if the correct server, you can click the search icon to see a list of all associated clusters. For more information, see Section 2.3.1, "Using the Search Function".
	Note: WebLogic Host and **WebLogic Cluster** are mutually-exclusive fields; you should only specify on. If you provide a value for both fields, WebLogic Cluster takes precedence.
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.
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	This region is used 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`
Location	This table is used to specify any location overrides. See step 6, below.

5. If necessary, add any expression overrides in the **Match Expression** field.

6. If necessary, add any location overrides in the **Location** table. To do so:

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 Section 2.3.2, "Using the AutoFill Function").

d. For 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. For 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 must append a File Name, use the `DefaultFileName` module parameter instead of Path Prepend.

---

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

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

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

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

2.3.1 Using the Search Function

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.

2.3.2 Using the AutoFill Function

You can easily add valid WebLogic Server and endpoint locations for a specified Base UIRL 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. Type 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 Plug-In Manually

You can configure the Oracle WebLogic Server Proxy Plug-In manually by specifying directives in the mod_wl_ohs.conf file.

1. Ensure you have fulfilled the prerequisites listed in Section 2.2.

2. Open the mod_wl_ohs.conf file, which is located in the following directory, in a text editor:

   DOMAIN_ HOME/config/fmwconfig/components/OHS/instances/*componentName*

3. Look for the `<IfModule weblogic_module>` element.

4. Add directives within the `<IfModule weblogic_module>` element in the configuration file, as follows:

   > **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 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 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 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—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 of 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 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 mod_wl_ohs.conf 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 of parameter to use, see Section 7.2, "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.

For more information, 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.

- 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 Use SSL with Plug-Ins.

  For information about the other directives that you can specify in the mod_wl_ohs.conf file, see Chapter 7, "Parameters for Web Server Plug-Ins."

5. Restart Oracle HTTP Server by using one of the techniques described in "Starting Oracle HTTP Server", in *Administering Oracle HTTP Server*.

2.5 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 are prefixed with `weblogic:` to easily identify them; for example:

```
[Fri Dec 27 12:09:14 2013] [debug] ap_proxy.cpp(143): [client 192.168.1.123]
weblogic: INFO: SSL is configured, referer:@
https://example.com/app/fileUploadAction.do
```

The directives WLLogFile and Debug are deprecated. If the configuration still uses any of these directives, the following note will appear in the console log file:

```
The WLLogFile directive is ignored. The web server log file is used instead. The
Debug directive is ignored. The web server log level is used instead.
```

To enable plug-in logs:

- If OraLogMode is set to ODL-text, set OraLogSeverity to TRACE:32. 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.

For more details on Managing Oracle HTTP Server Logs, See "Managing Oracle HTTP Server Logs".

2.6 Troubleshooting Oracle WebLogic Server Proxy Plug-In Implementations

This section describes common problems that you might encounter when using the Oracle WebLogic Server Proxy Plug-In and explains how to solve them. It includes the following topics:

- Oracle WebLogic Server Session Issues

- CONNECTION_REFUSED Errors

- NO_RESOURCES Errors

2.6.1 Oracle WebLogic Server Session Issues

The Oracle WebLogic Server Proxy Plug-In routes the requests to backend WebLogic Server server/cluster. WebLogic Server maintains sessions so that subsequent requests

from the same client are routed to the same WLS server. However, due to various reasons, if the Oracle WebLogic Server Proxy Plug-In cannot communicate with the WLS server:

- 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 WebLogic Server server is marked as bad, and the request is routed to the next available WebLogic Server 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.
```

2.6.2 CONNECTION_REFUSED Errors

Occasionally, under stress conditions, 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 Section 6.4.2, "Tips for reducing Connection_Refused Errors", WLS server might have reached the maximum allowed backlog connections.

To resolve, follow the steps mentioned in Section 6.4.2, "Tips for reducing Connection_ Refused Errors".

2.6.3 NO_RESOURCES Errors

Occasionally, under stress conditions, 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 WLS 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 by the WLS server.

3

Configuring the Plug-In for Apache HTTP Server

This chapter describes how to install and configure the Oracle WebLogic Server Proxy Plug-In for Apache HTTP Server. It contains the following sections:

> **Note:** For proxying requests from Oracle HTTP Server to Oracle WebLogic Server, use the mod_wl_ohs plug-in, which is similar to the plug-in for Apache HTTP Server, but need not be downloaded and installed separately. For information about configuring mod_wl_ohs, see Chapter 2, "Configuring the Plug-In for Oracle HTTP Server.".

3.1 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 Plug-In for Apache HTTP Server

After you download the Oracle WebLogic Server Proxy Plug-In for Apache HTTP Server, as described in Section 1.2, "Availability of Oracle WebLogic Server Proxy Plug-In 12.1.3,", you can install it as an Apache HTTP Server module in your Apache HTTP Server installation.

3.2.1 Installation Prerequisites

Before you install the Oracle WebLogic Server Proxy Plug-In for Apache HTTP Server, do the following:

- Download the Oracle WebLogic Server Proxy Plug-In for Apache HTTP Server, as described in Section 1.2, "Availability of Oracle WebLogic Server Proxy Plug-In 12.1.3."

- Extract the plug-ins zip distribution to PLUGIN_HOME; for example, `/home/myhome/weblogic-plugins-12.1.3/`. 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 6 to use SSL. The JDK 6 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.

  For more information, see:

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

  ---
  Note: If you intend to use Apache HTTP Server 2.4, then please upgrade to 12.1.*x* plug-ins.

- 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). You can obtain the plug-in from here:

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

To install the Apache HTTP Server plug-in:

1. Ensure the weblogic-plugins-12.1.3/lib folder is included in LD_LIBRARY_PATH on UNIX systems (and PATH on Windows systems). If the folder is not included, 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-12.1.3/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-12.1.3/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-12.1.3/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.

3.3.1 Editing the httpd.conf File

Edit the httpd.conf file in your Apache HTTP Server installation to configure the Apache HTTP Server plug-in.

1. Open the httpd.conf file, if it is not already open.

2. To proxy requests by MIME type, add an IfModule block that defines one of the following:

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

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

Example:

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

3. To proxy requests by MIME type, add a MatchExpression line to the `<IfModule>` block. If both MIME type and proxying by path are enabled, then proxying by path takes precedence over proxying by MIME type.

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>
```

4. To proxy requests by path, use the `<Location>` block and the `WLSRequest` statement. `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>
```

The PathTrim parameter specifies a string trimmed from the beginning of the URL before the request is passed to the WebLogic Server instance (see Section 7.1, "General Parameters for Web Server Plug-Ins").

5. The PathTrim parameter must be configured inside the `<Location>` tag. 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 the **correct** setup:

```
<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.

6. Optionally, enable HTTP tunneling for t3 or IIOP.

   a. 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>
   ```

   b. 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>
   ```

7. 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 Section 7.1, "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.

8. 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
   ```

9. Start the Apache HTTP Server.

- UNIX/Linux

  ```
  > APACHE_HOME/bin/apachectl start
  ```

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

3.3.1.1 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>
```

3.3.1.2 Example: Configuring the Apache Plug-In

The following 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-12.1.3/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 is in the *LD_LIBRARY_PATH* by entering the following command:

```
$ export LD_LIBRARY_PATH=/home/myhome/weblogic-plugin-12.1.3/lib:...
```

> **Note:** You can also update the 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

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

If you want 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.

This section describes how to create weblogic.conf files, and includes sample weblogic.conf files.

3.3.2.1 Creating weblogic.conf Files

Be aware of the following when 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 agarwalp01:7001
 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.

3.3.2.2 Sample weblogic.conf Configuration Files

The following 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 3–1   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. (Except WebLogicHost,
# WebLogicPort, WebLogicCluster, and CookieName.)

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

In Example 3–2, 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 3–2   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 (Except WebLogicHost,
# WebLogicPort, WebLogicCluster, and CookieName.)

<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 3–3 shows an example without WebLogic clusters.

Example 3–3   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 (Except WebLogicHost,
# WebLogicPort, WebLogicCluster, and CookieName.)
<IfModule mod_weblogic.c>
 WebLogicHost myweblogic.server.com
 WebLogicPort 7001
 MatchExpression *.jsp
</IfModule>
```

Example 3–4 shows an example of configuring multiple name-based virtual hosts.

Example 3–4   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.

3.3.2.3 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.

Apache HTTP Server is not case sensitive.

Example 3–5   Sample httpd.conf file for Apache HTTP Server

```
####################################################
APACHE-HOME/conf/httpd.conf file
####################################################
LoadModule weblogic_module lhome/myhome/weblogic-plugins-12.1.3/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 w1s1.com:7001,w1s2.com:7001,w1s3.com:7001
  ErrorPage http://myerrorpage.mydomain.com
</IfModule>
```

3.4 Deprecated Directives for Apache HTTP Server

The WebLogic Server plug-in logs are now part of the Apache HTTP Server error log and are prefixed with `weblogic:` to easily identify them. Hence the directives WLLogFile and Debug are deprecated. If the configuration still uses any of these directives, the following note will appear during startup::

```
The WLLogFile directive is ignored. The web server log file is used instead.
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 ErrorLog.

4

Configuring the Plug-In for iPlanet Web Server

This chapter describes how to install and configure the Oracle WebLogic Server Proxy Plug-In 12.1.3 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:

- Section 4.1, "Overview of the Plug-In for iPlanet"
- Section 4.2, "Installing and Configuring the Plug-In for iPlanet"
- Section 4.3, "Deprecated Directives for iPlanet Web Server"

4.1 Overview of the Plug-In for iPlanet

The Oracle WebLogic Server Proxy Plug-In 12.1.3 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 WebLogic Server to handle those requests that require the dynamic functionality of WebLogic Server.

The Oracle WebLogic Server Proxy Plug-In 12.1.3 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 system) 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 12.1.3 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 12.1.3 for iPlanet Web Server, providing access to all WebLogic Server services (not just dynamic pages).

The Oracle WebLogic Server Proxy Plug-In 12.1.3 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/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 12.1.3 for iPlanet Web Server.

4.2.1 Installation Prerequisites

Before you install the Oracle iPlanet Web Server plug-in, do the following:

- Create a plug-in zip extract location (*PLUGIN_HOME*; for example, /home/myhome/weblogic-plugins-12.1.3/)

- Download the Oracle WebLogic Server Proxy Plug-In 12.1.3 for iPlanet Web Server, as described in Section 1.2, "Availability of Oracle WebLogic Server Proxy Plug-In 12.1.3."

- 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 (.dll on Windows)	WebLogic proxy module
lib/*.so (.dll on Windows)	Helper libraries

- Install JDK 6 or later to use SSL. You must have a JDK 6 or later 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 12.1.3 for iPlanet Web Server installation (7.0.9 or later) installed on *IPLANET_HOME*; that is, iPlanet server listening on iplanet-host:iplanet-port.

  The version 12.1.3 plug-in is supported on the Oracle WebLogic Server Proxy Plug-In 12.1.3 for iPlanet Web Server platforms described in: http://www.oracle.com/technetwork/middleware/ias/downloads/fusion-certification-100350.html

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

- Ensure 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/technetwork/middleware/ias/downloads/fusion-certification-100350.html

4.2.2 Installing the Plug-In for iPlanet Web Server

The Oracle WebLogic Server Proxy Plug-In 12.1.3 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. For more information, see:

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

4.2.3 Configuring the Plug-In for iPlanet Web Server

This section provides information about configuring the Oracle WebLogic Server Proxy Plug-In 12.1.3 for iPlanet Web Server.

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.
For more information, see

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

There are different ways to configure obj.conf file.

Read guidelines for Section 4.2.5, "Guidelines for Modifying the obj.conf File". The obj.conf file defines which requests are proxied to WebLogic Server and other configuration information.

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 12.1.3 for iPlanet Web Server. For a complete list of plug-in parameters, see Section 7.1, "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/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 12.1.3 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 12.1.3 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 following in the obj.conf file under the `name="default"`
`#NameTrans fn="ntrans-j2ee" name="j2ee"` and restart the web server. Optionally,

- 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:

1. Start WebLogic Server.

2. Start Oracle iPlanet Web Server. If Oracle WebLogic Server Proxy Plug-In 12.1.3 for iPlanet Web Server is already running, you must either restart or reconfigure the server.

3. You can test the Oracle WebLogic Server Proxy Plug-In 12.1.3 for iPlanet Web Server plug-in using the following URL. It should bring up 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 on 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 12.1.3 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 Section 4.2.6, "Sample obj.conf File (Not Using a WebLogic Cluster)" and Section 4.2.7, "Sample obj.conf File (Using a WebLogic Cluster)".

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

    ```
    set PATH=/home/user/weblogic-plugin-12.1.3/lib:..
    ```

    > **Note:** You can also update the 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 12.1.3 for iPlanet Web Server, you must make several modifications to the obj.conf file. For more information, see

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

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

Below 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)

Below 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 and are prefixed with `weblogic:` to easily identify them. Hence the directives

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

```
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

The following sections describe how to install and configure the Oracle WebLogic Server Proxy Plug-In 12.1.3 for Microsoft IIS Web Server:

- Section 5.1, "Working with the Microsoft IIS 7.0 Plug-In"

- Section 5.2, "Working with the Microsoft IIS 7.5 Plug-In"

Download the Oracle WebLogic Server Proxy Plug-In for IIS Web Server, as described in Section 1.2, "Availability of Oracle WebLogic Server Proxy Plug-In 12.1.3." The zip file contains these 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

5.1 Working with the Microsoft IIS 7.0 Plug-In

This section describes how to install and configure the plug-in for Microsoft IIS 7.0.

- Section 5.1.1, "Installing and Configuring the Microsoft IIS 7.0 Plug-In"

- Section 5.1.2, "Serving Static Files from the Web Server"

- Section 5.1.3, "Using Wildcard Application Mappings to Proxy by Path"

- Section 5.1.4, "Proxying Requests from Multiple Virtual Websites to WebLogic Server"

- Section 5.1.5, "Creating ACLs Through IIS 7.0"

- Section 5.1.6, "Testing the Installation"

5.1.1 Installing and Configuring the Microsoft IIS 7.0 Plug-In

This section describes differences in how you set up the Microsoft Internet Information Server plug-in for IIS 7.0.

To set up the Microsoft Internet Information Server plug-in for IIS 7.0, follow these steps:

1. Download and install Microsoft IIS 7.0.

2. Ensure that all of the necessary features of Microsoft IIS 7.0 are enabled.

3. Download and install the latest Oracle WebLogic Server Proxy Plug-In zip file.

4. Create a web application in IIS Manager by right clicking **Web Sites** then **Add Web Site**.

   Fill in the **Web Site Name** with the name you want to give to your web application; for example, MyApp. Select the physical path of your web application Port (any valid port number not currently in use).

   Click **OK** to create the web application.

   If you can see the name of your application under **Web Sites** it means that your application has been created and started running. Click the MyApp node under **Web Sites** to see all of the settings related to the MyApp application, which you can change, as shown in Figure 5–1.

Figure 5–1   Application Home Page
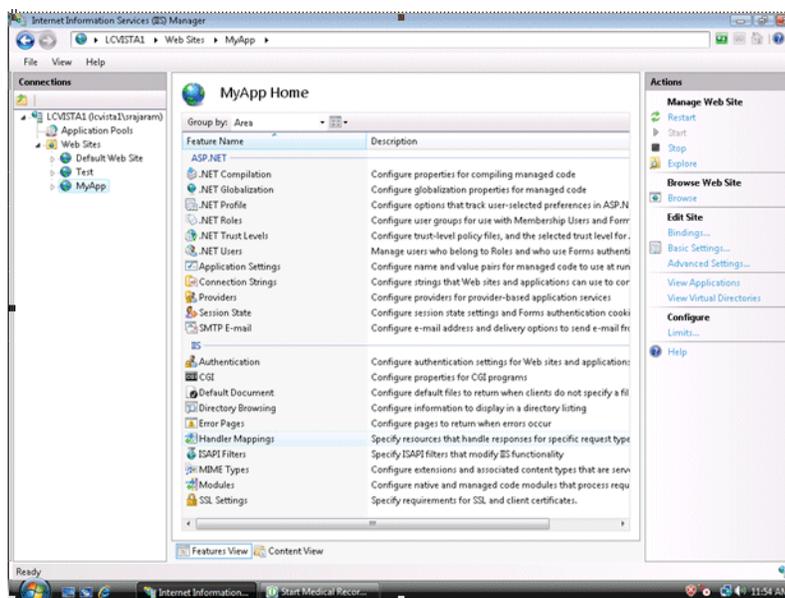
5. Click **Handler Mappings** to set the mappings to the handler for a particular MIME type.

Figure 5–2   Setting the Handler Mappings
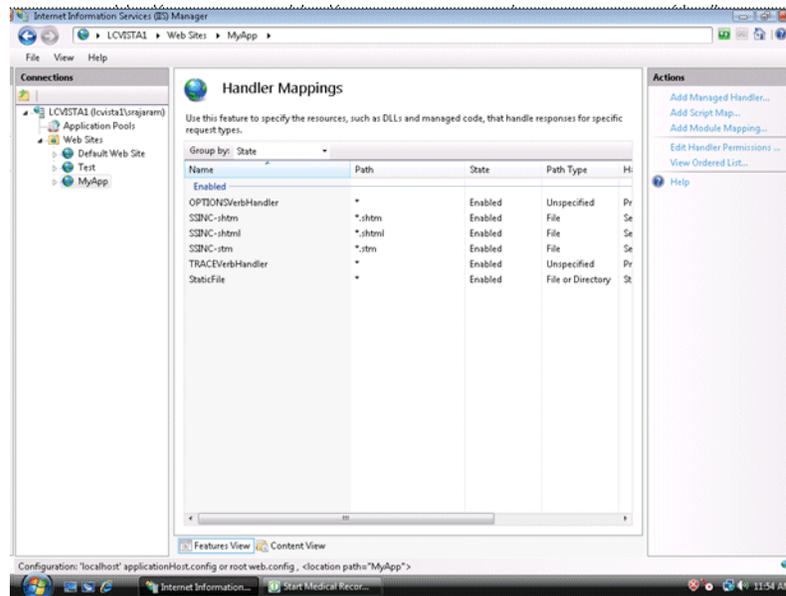
6. Click the StaticFile and change the Request path from * to *.*. Click **OK**.

Figure 5–3   Editing the Request Path for Module
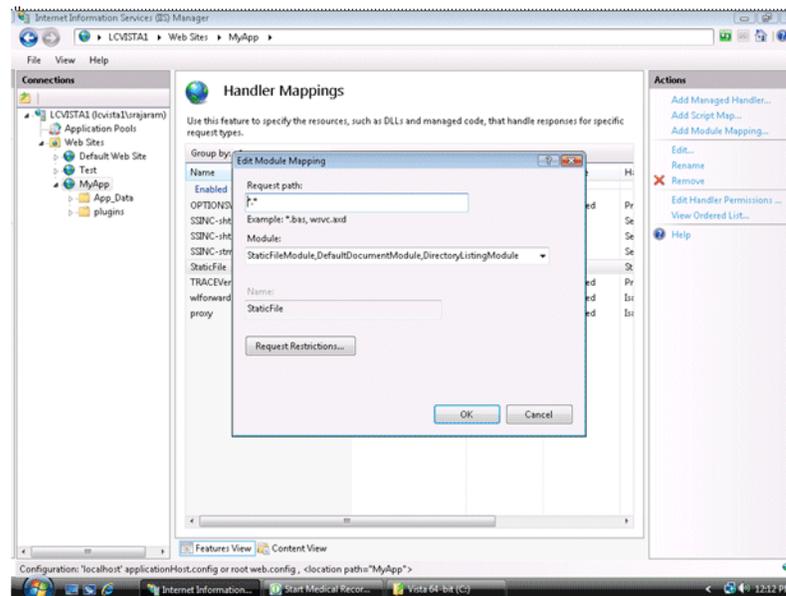
7. Click **MyApp** and then click **Add Script Map** on the right-hand side menu options. Enter * for the Request path.

    Browse to the iisproxy.dll file and add it as the executable. Name it `proxy`.

Figure 5–4 Editing the Request Path for Script
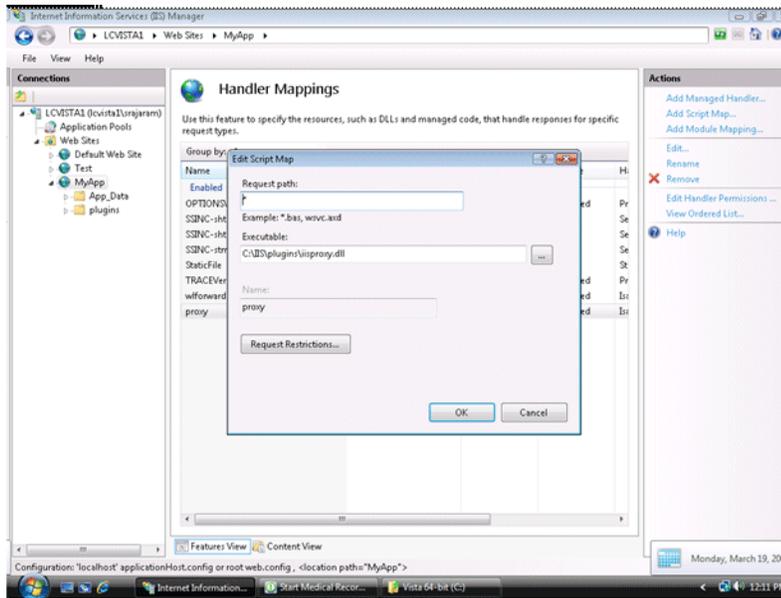
8. Click **Request Restrictions** and deselect **Invoke handler only if the request is mapped to**.

Figure 5–5 Editing the Request Restrictions
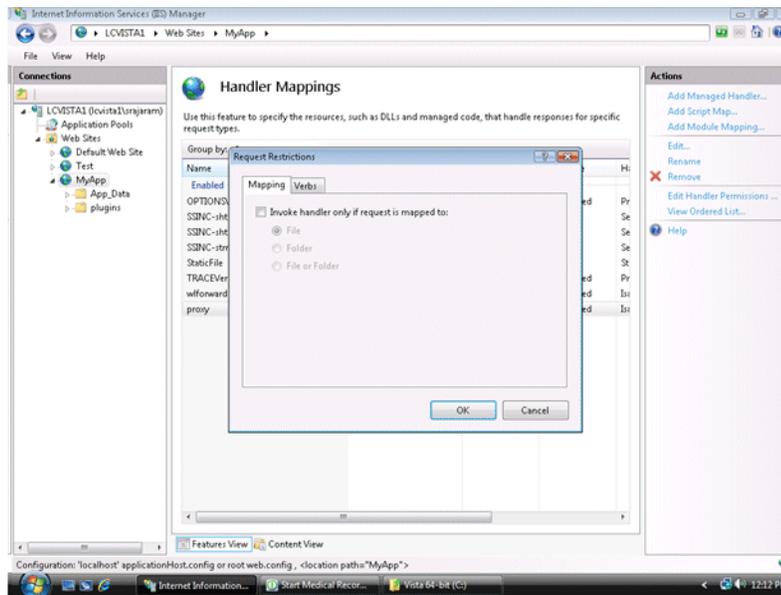
9. Click **OK** to add this Handler mapping. Click **Yes** on the Add Script Map dialog box.

Figure 5–6   Adding the Script Map
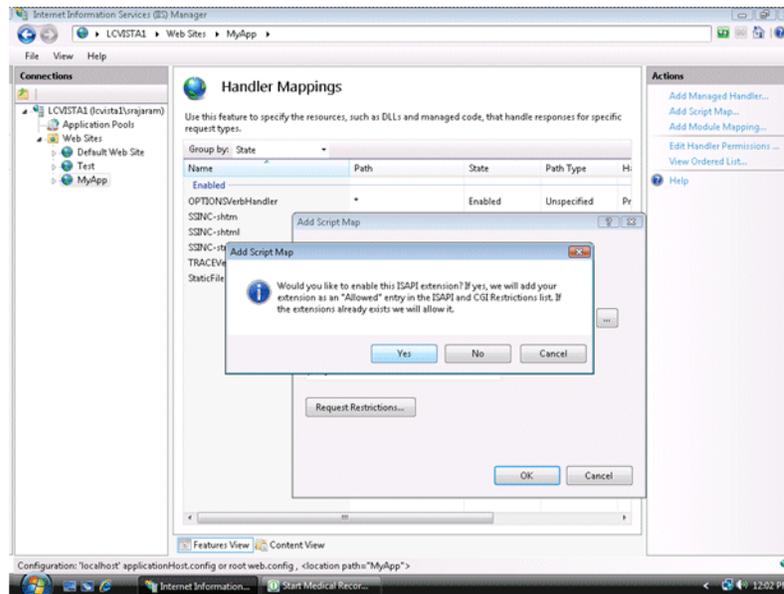
10. To configure proxying by path, see Section 5.1.3, "Using Wildcard Application Mappings to Proxy by Path".

11. Click the Root node of the IIS Manager tree and click the ISAPI and CGI Restrictions. Make sure to check **Allow unspecified ISAPI modules**.

Figure 5–7   Editing ISAPI and CGI Restrictions
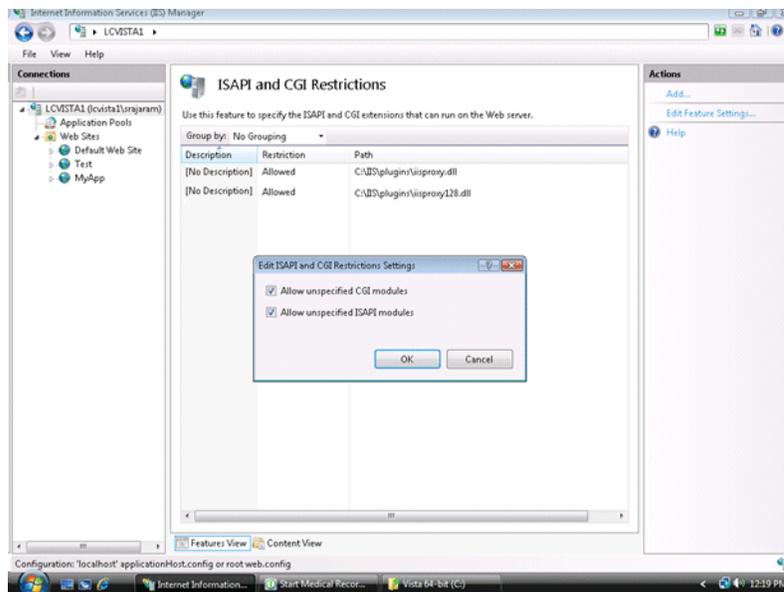
12. Create a file called iisproxy.ini with the following contents and place it in the directory with the plug-in:

```
WebLogicHost= @hostname@
WebLogicPort= @port@
ConnectRetrySecs=5
ConnectTimeoutSecs=25
Debug=ALL
```

```
DebugConfigInfo=ON
KeepAliveEnabled=true

WLLogFile=@Log file name@
SecureProxy=OFF
```

13. Open the Internet Explorer browser and enter `http://<hostname>:<port>`. You should be able to see the Medrec Sample Application from your Oracle WebLogic Server.

    To run the plug-in SSL mode, change the value of WeblogicPort to the SSL port of your application, change the `SecureProxy` value to ON, and set `WLSSLWallet` to the location of the wallet. For more information on SSL parameters, see Section 7.2, "SSL Parameters for Web Server Plug-Ins".

Figure 5–8   Medrec Sample Application
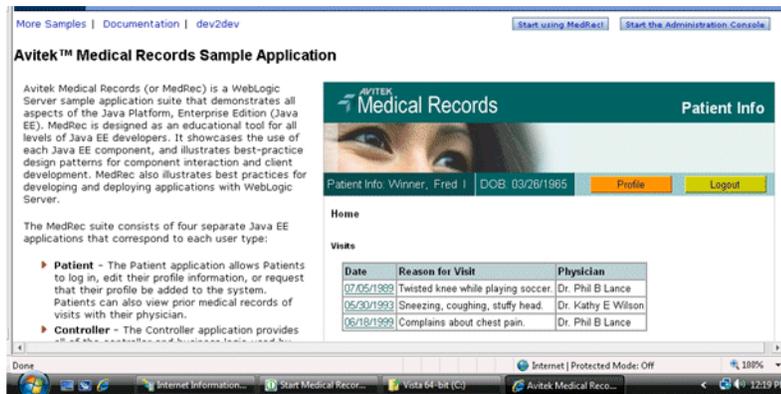
5.1.2 Serving Static Files from the Web Server

In order to have IIS serve all static content that could be included on a web application that is to be served by WebLogic Server, do the following:

1. Configure your application by setting up Oracle WebLogic Server Proxy Plug-In 12.1.3 for Microsoft IIS Web Server on IIS Web Server as described in Section 5.1.1, "Installing and Configuring the Microsoft IIS 7.0 Plug-In."

   Assume that you created a Handler Mapper named proxy as described on the Oracle documentation.

   > **Important:**   Do not use `WLEXCLUDEPATHORMIMETYPE` property inside your proxy setup. It is not required neither useful here and can only confuse the understanding of the flow.

2. On IIS Manager, display the home page by clicking the Virtual Directory or Application created on step 1.

3. Double-click the Handling Mappers and then click **View Ordered List** on the right side pane. An ordered list of Handler Mappings appears.

4. Select **proxy** and drag it below **StaticFile** handler mapping (in other words the StaticFile handler mapping should be above the proxy handler mapping.)

5. Edit the Static File and change the request path to: *.jpg. Save the file.

6. Configure IIS to serve types of static files, for example, PNGs, GIFs, or CSS, do the following:

    a. On IIS Manager, display the home page by clicking the Virtual Directory or Application created on step 1.

    b. Double click the Handling Mappers and then click **Add Module Mapping** on the right side pane.

    c. Choose a Request Path of desired type: for PNGs use *.png, for GIFs use *.gif and so on. For Module, choose StaticFileModule, enter a name, and click **OK**.

    d. Ensure that as stated on step 4, the newly created HandlerMapping is ordered before the proxy Handler Mapping defined on step 1.

5.1.3 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.

For more information, see "Add a Wildcard Script Map" for IIS 7.5 (http://technet.microsoft.com/en-us/library/cc754606(WS.10).aspx).

5.1.3.1 Adding a Wildcard Script Map for IIS 7.0 and Above

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 in IIS 7" 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.

5.1.4 Proxying Requests from Multiple Virtual Websites 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 Section 1.1.2, "Proxying Requests". 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.

6. Create a separate application pool for each virtual directory.

   As described in "Creating Application Pools (IIS 6.)" (http://www.microsoft.com/technet/prodtechnol/WindowsServer2003/Library/IIS/93275ef2-2f85-4eb1-8b92-a67545be11b4.mspx?mfr=true), you can isolate different Web applications or websites in pools, which are called application pools. In an application pool, process boundaries separate each worker process from other worker processes so that when an application is routed to one application pool, applications in other application pools do not affect that application.

   > **Note:** Step 6 only applies to IIS 6.0 as there is no Application Protection option with IIS 7.0.
   >
   > The above step will work with IIS 7.0, except that exactly one IIS virtual directory exists within a particular IIS site. The limitation exists because, with IIS 7.0, it is impossible to assign a unique application pool to each virtual directory, thus the subsequent requests are always proxied to the first website(virtual directory).

5.1.4.1 Sample iisproxy.ini File

Here is a sample iisproxy.ini file for use 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
```

Here is a sample iisproxy.ini file 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.

5.1.5 Creating ACLs Through IIS 7.0

ACLs will not work through the Oracle WebLogic Server Proxy Plug-In 12.1.3 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.

When using Basic Authentication, the user is logged on with local log-on rights. To enable the use of Basic Authentication, grant each user account the Log On Locally user right on the IIS server. 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 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.

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".

5.1.6 Testing the Installation

After you install and configure the Microsoft IIS plug-in, follow these steps for deployment and testing:

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.

5.2 Working with the Microsoft IIS 7.5 Plug-In

This section describes how to install and configure the plug-in for Microsoft IIS 7.5.

- Section 5.2.1, "Installing and Configuring the Microsoft IIS 7.5 Plug-In"

- Section 5.2.2, "Serving Static Files with IIS 7.5"

- Section 5.2.3, "Serving Static Files and Dynamic Content From the Same Request with IIS 7.5"

5.2.1 Installing and Configuring the Microsoft IIS 7.5 Plug-In

Follow these steps to install and configure the Microsoft IIS 7.5 Plug-In.

1. Install Microsoft IIS 7.5.

2. Ensure that all of the necessary features of Microsoft IIS 7.5 are enabled.

    a. From the **Start** menu, choose **Control Panel**, then **Programs and Features**. Select **Internet Information Services (IIS) 7+ Manager**. Click **Turn Windows features on or off**.

    b. Expand the entire tree beneath Internet Information Services and ensure **all** of the subfeatures are selected.

Figure 5–9   Windows Features for Internet Information Services
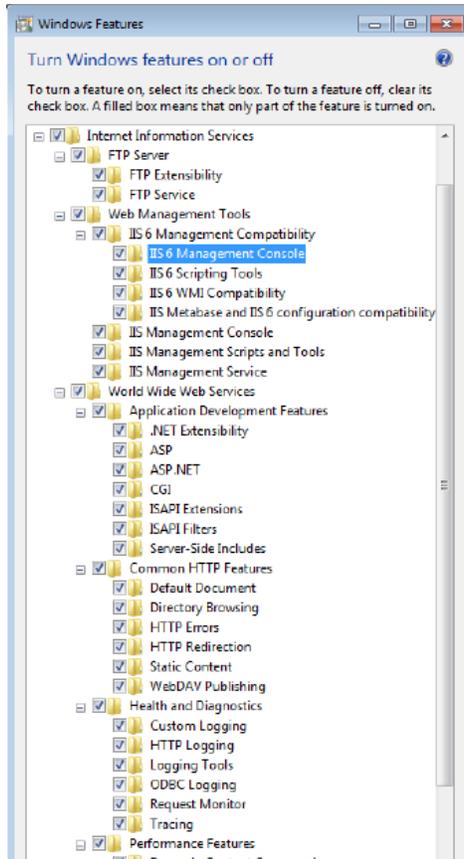
3. Download and install the latest Oracle WebLogic Server Proxy Plug-In zip file.

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**, then **System**, then **System Properties**, then **Environment Variables**, then **System Properties**, then **PATH**).

6. Open IIS Manager (select **Start**, then **All Programs**, then **Administrative tools**, then **Internet Information Services Manager**).

Figure 5–10   Internet Information Services Manager Window
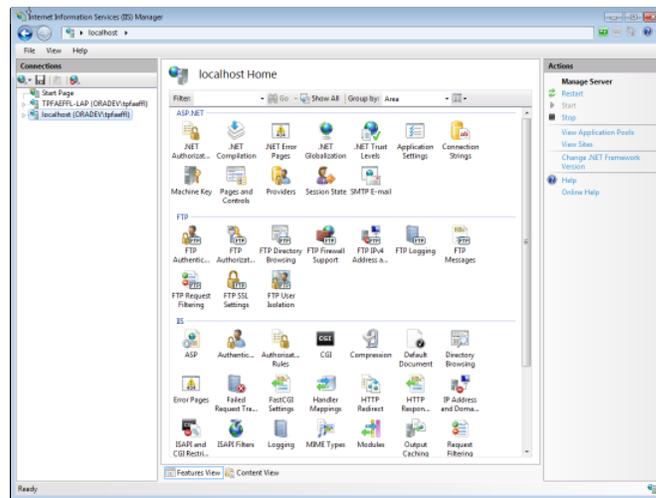
7. Create a new Web Site in IIS. See the IIS Help system for more information.

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\iiproxy.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 5–11   Windows Services Window
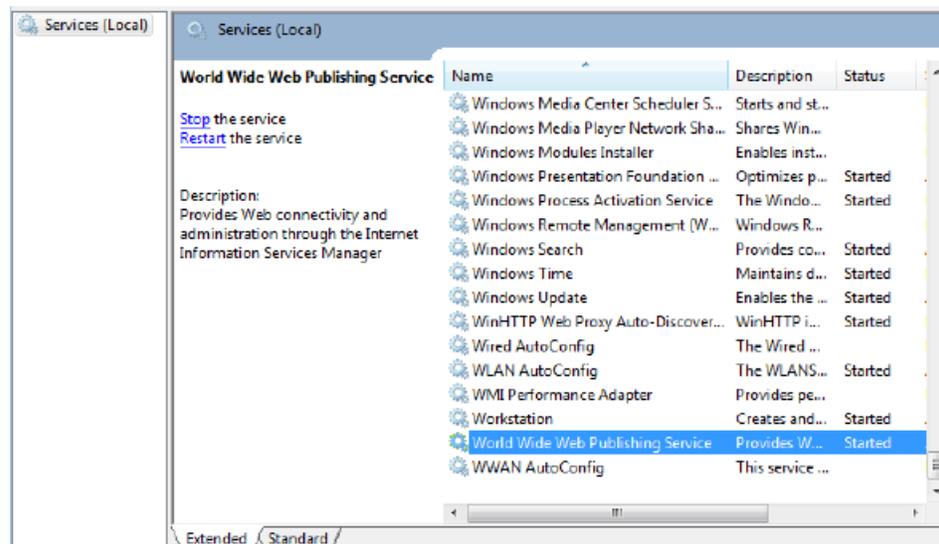
12. Test your configuration by sending a request to `http://iishost:iisport/application_name/` from the browser and validate the response.

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.

5.2.2 Serving Static Files with IIS 7.5

After configuring the WLS plug-in and confirming it works (see Section 5.2.1, "Installing and Configuring the Microsoft IIS 7.5 Plug-In") 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.
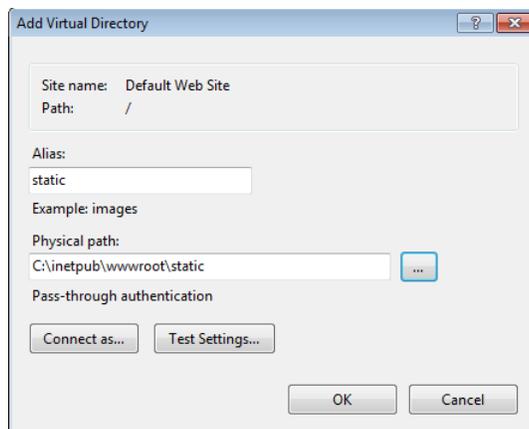
3. Click **static** under **Default Web Site** to open the **static Home** page.

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 7.5 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`

5.2.3 Serving Static Files and Dynamic Content From the Same Request with IIS 7.5

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 Section 5.2.2, "Serving Static Files with IIS 7.5."

1. Complete the steps described in the Section 5.2.2, "Serving Static Files with IIS 7.5."

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`.
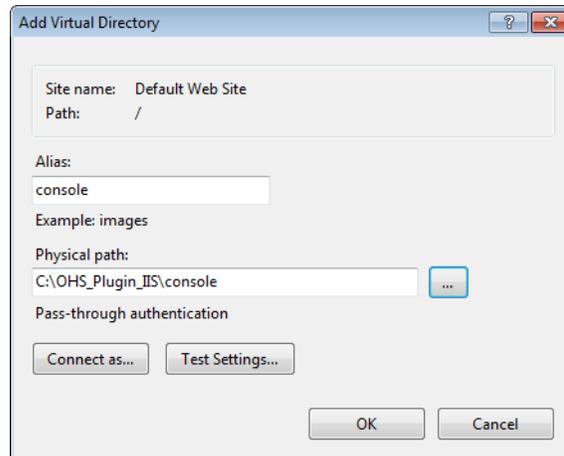
   Click **OK**. You will see a "`console`" entry in the left pane under the Default Web Site.

3. Click "console" under "Default Web Site". A "console Home" will open 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)
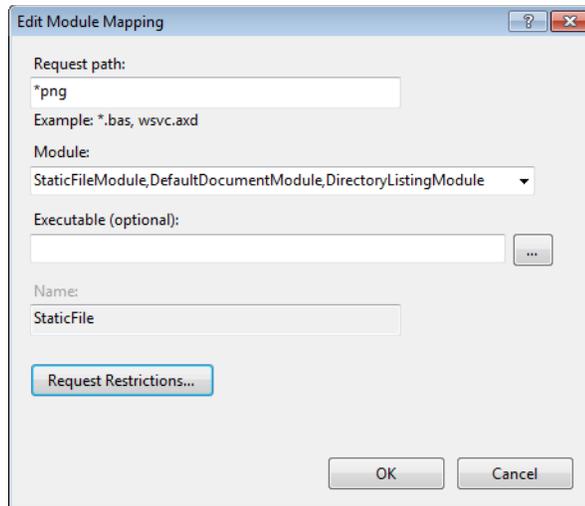
6. Click **Request Restrictions**. In the **Mapping** tab of the Request Restrictions dialog box, ensure that **Invoke Handler only if the request is mapped to** is selected, then select **File or Folder**. Click **OK** and **OK** to dismiss the dialog boxes.
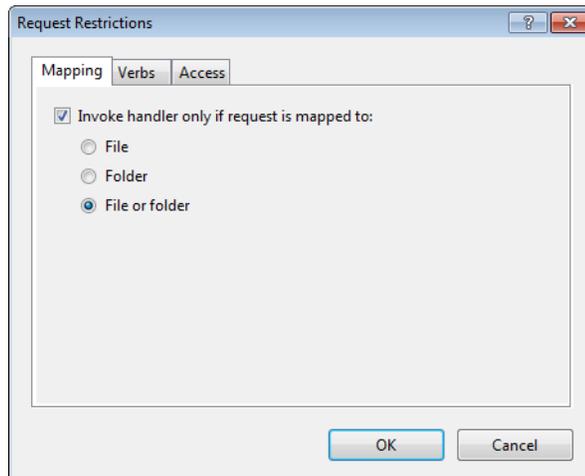
The above step is to serve the *.png from the IIS server.

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.

6

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:

- Section 6.1, "Use SSL with Plug-Ins"
- Section 6.2, "Use IPv6 With Plug-Ins"
- Section 6.3, "Set Up Perimeter Authentication"
- Section 6.4, "Understanding Connection Errors and Clustering Failover"

6.1 Use 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 Section 7.2, "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 12.1.3 plug-ins use Oracle wallets to store SSL configuration information. The plug-ins introduce a new SSL configuration parameter `WLSSLWallet` to use Oracle 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.

  For more information, 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 *Oracle Fusion Middleware 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* (12.1.3) 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.

6.1.1 Configure Libraries for SSL

The 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 12.1.3 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. For more information, see "Configuring SSL" in *Securing Oracle WebLogic Server*.

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

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

For more information, see "Using the orapki Utility for Certificate Validation and CRL Management" in the *Oracle Fusion Middleware Administrator's Guide*.

> **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 Microsoft IIS Web Server is different from the user who creates the wallet (or for Windows, the account SYSTEM), you must 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
       SecureProxy=ON
       WLSSLWallet=path_to_wallet
      ```

    - For iPlanet Web Server, edit the `config/obj.conf` or `config/<vs>-obj.conf` file as follows:

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

    For more information about the parameters in these examples, see Chapter 7, "Parameters for Web Server Plug-Ins."

5. If the version of the Oracle WebLogic Server instances in the back end is 10.3.4 (or a later release), do the following:

   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.

      The Configuration: General tab is displayed.

   d. Scroll down to the Advanced section, 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 b, repeat steps c and d 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 Configure 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 Section 6.1.2, "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 12.1.3 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). For more information, see "Configure two-way SSL" in the *Oracle WebLogic Server Administration Console Help*.

6.2 Use IPv6 With Plug-Ins

The version 12.1.3 plug-ins support IPv6. Specifically, the `WebLogicHost` and `WebLogicCluster` configuration parameters (see Table 7–1) 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.

6.3 Set Up 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 *Programming Security for Oracle WebLogic Server*.

5. Web server plug-ins require a trusted Certificate Authority file to use SSL between the plug-in and WebLogic Server. See Section 6.1, "Use SSL with Plug-Ins" for the steps you must perform to configure SSL.

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

6.4 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 6–1 demonstrates how the plug-in handles failover.

6.4.1 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

6.4.2 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, set the `TcpTimedWaitDelay` on the proxy and WebLogic Server servers to a lower value. Set the TIME_WAIT interval in Windows 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 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
  ```

6.4.3 Failover with a Single, Non-Clustered WebLogic Server

If you are running 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`.

6.4.4 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. 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. 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 decided they want to add another member on the fly and not need to restart Oracle HTTP Server.

> **Note:** If `DynamicServerList` only works when set to `OFF` when using clustered WebLogic Servers using WebLogicCluster, WebLogic Server is encountering a problem obtaining or delivering the dynamic list.

6.4.5 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 6–1.

> **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 6–1   Connection Failover
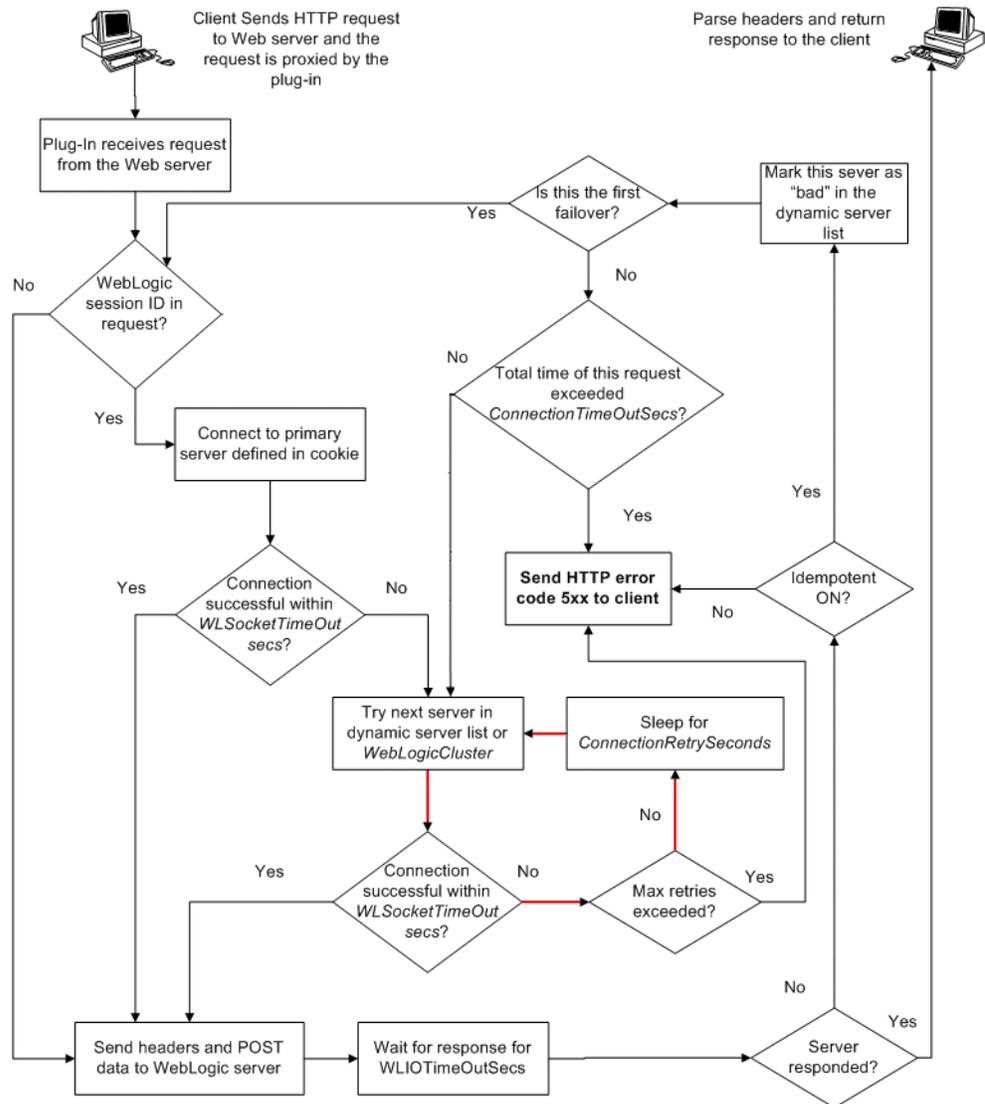
In this figure, the Maximum number of retries allowed in the red loop is equal to `ConnectTimeoutSecs/ConnectRetrySecs`.

6.4.6 Using 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 12.1.3 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 12.1.3 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 12.1.3 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. For more information, see "Configuring SSL" in *Securing 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 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 Section 7.2, "SSL Parameters for Web Server Plug-Ins."

6.4.7 Failover Behavior When Using Firewalls and Load Directors

In most configurations, the Oracle WebLogic Server Proxy Plug-In 12.1.3 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.

6.5 Tuning OHS and Apache HTTP Server for High Throughput for WebSocket Upgrade Requests

WebLogic Server 12.1.3 supports deploying WebSocket applications. WebLogic Web Server Plug-In 12.1.3 for Oracle HTTP Server 12.1.3 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 12.1.3 and later. As a result of adding this support, a new configuration parameter WLMaxWebSocketClients is introduced.

WLMaxWebSocketClients 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 MaxClients (non-Windows). Hence, to tune your HTTP Server for maximum WebSocket connection upgrade requests, set MaxClients/ThreadsPerChild to a value that can accommodate WebSocket connections as well. Also, ensure that WLMaxWebSocketClients is set to 75 percent of MaxClients/ThreadsPerChild.

7

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:

- Section 7.1, "General Parameters for Web Server Plug-Ins"
- Section 7.2, "SSL Parameters for Web Server Plug-Ins"

> **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:
>
> - Chapter 3, "Configuring the Plug-In for Apache HTTP Server"
> - Chapter 2, "Configuring the Plug-In for Oracle HTTP Server"
> - Chapter 5, "Configuring the Plug-In for Microsoft IIS Web Server"
> - Chapter 4, "Configuring the Plug-In for iPlanet Web Server"

7.1 General Parameters for Web Server Plug-Ins

The general parameters for Web server plug-ins are shown in Table 7–1. The parameters are case sensitive.

Table 7–1   General Parameters for Web Server Plug-Ins

Parameter Name	Default	Description	Applicable to
WLIOTimeoutSecs (new name for HungServerRecoverSecs)	300	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 terminated, 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: 2147483647	Oracle HTTP Server Oracle iPlanet Web Server Apache HTTP Server Microsoft IIS
WebLogicCluster (Required when proxying to a cluster of WebLogic Servers, or to multiple non-clustered servers.)	none	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. For more information, see the following: ■ Chapter 3, "Configuring the Plug-In for Apache HTTP Server" ■ Chapter 2, "Configuring the Plug-In for Oracle HTTP Server" ■ Chapter 5, "Configuring the Plug-In for Microsoft IIS Web Server" ■ Chapter 4, "Configuring the Plug-In for iPlanet 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.	Oracle HTTP Server Oracle iPlanet Web Server Apache HTTP Server Microsoft IIS

Table 7–1   (Cont.) General Parameters for Web Server Plug-Ins

Parameter Name	Default	Description	Applicable to
WebLogicHost (Required when proxying to a single WebLogic Server.)	none	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.	Oracle HTTP Server Oracle iPlanet Web Server Apache HTTP Server Microsoft IIS
WebLogicPort (Required when proxying to a single WebLogic Server.)	none	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.	Oracle HTTP Server Oracle iPlanet Web Server Apache HTTP Server Microsoft IIS
WLCookieName (CookieName is deprecated.)	JSESSIONID	If you change the name of the WebLogic Server session cookie in the WebLogic Server Web application, 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.	Oracle HTTP Server Oracle iPlanet Web Server Apache HTTP Server Microsoft IIS
ConnectRetrySecs	2	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.	Oracle HTTP Server Oracle iPlanet Web Server Apache HTTP Server Microsoft IIS
ConnectTimeoutSecs	10	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.	Oracle HTTP Server Oracle iPlanet Web Server Apache HTTP Server Microsoft IIS

Table 7–1   (Cont.) General Parameters for Web Server Plug-Ins

Parameter Name	Default	Description	Applicable to
Debug **DEPRECATED**	OFF	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 TEMP directory has write permission assigned to the user who is logged in to the server. 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 Section 3.4, "Deprecated Directives for Apache HTTP Server."	Microsoft IIS

Table 7–1   (Cont.) General Parameters for Web Server Plug-Ins

Parameter Name	Default	Description	Applicable to
DebugConfigInfo	OFF	Enables the special query parameter "__WebLogicBridgeConfig". Use it to get details about configuration parameters from the plug-in.	Oracle HTTP Server
			Oracle iPlanet Web Server
			Apache HTTP Server
		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.	Microsoft IIS
		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.	
DefaultFileName	none	If the URI is "/" then the plug-in performs the following steps:	Oracle HTTP Server
			Oracle iPlanet Web Server
		Trims the path specified with the PathTrim parameter.	Apache HTTP Server
		Appends the value of DefaultFileName.	Microsoft IIS
		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. For more information, 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.	

Table 7–1 (Cont.) General Parameters for Web Server Plug-Ins

Parameter Name	Default	Description	Applicable to
DynamicServerList	ON	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.	Oracle HTTP Server Oracle iPlanet Web Server Apache HTTP Server Microsoft IIS
ErrorPage	none	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.	Oracle HTTP Server Oracle iPlanet Web Server Apache HTTP Server Microsoft IIS

Table 7–1   (Cont.) General Parameters for Web Server Plug-Ins

Parameter Name	Default	Description	Applicable to
FileCaching	ON	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.	Oracle HTTP Server Oracle iPlanet Web Server Apache HTTP Server Microsoft IIS
Idempotent	ON	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 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 has been successfully sent to WebLogic Server and the plugin is now waiting for a response from backend server.	Oracle HTTP Server Oracle iPlanet Web Server Apache HTTP Server Microsoft IIS

Table 7–1 (Cont.) General Parameters for Web Server Plug-Ins

Parameter Name	Default	Description	Applicable to
KeepAliveEnabled	true (Microsoft IIS plug-in) ON (Oracle HTTP Server and Apache HTTP Server) ON (Oracle iPlanet Web Server)	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. Turn KeepAliveEnabled to OFF when using prefork mpm or use worker mpm in Apache. Valid values for Oracle iPlanet Webserver are ON and OFF	Oracle HTTP Server Oracle iPlanet Web Server Apache HTTP Server Microsoft IIS
KeepAliveSecs	20	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.	Oracle HTTP Server Oracle iPlanet Web Server Apache HTTP Server Microsoft IIS

Table 7–1 (Cont.) General Parameters for Web Server Plug-Ins

Parameter Name	Default	Description	Applicable to

MatchExpression	none	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 myHost:8080.
		
		The URIs that match the type jpg will be proxied to `myHost:8080/images` and others will be proxied to myHost:8080.
			Oracle HTTP Server
			
			Apache HTTP Server

Table 7–1   (Cont.) General Parameters for Web Server Plug-Ins

Parameter Name	Default	Description	Applicable to
		``` <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 Chapter 3, "Configuring the Plug-In for Apache HTTP Server."	
`MaxPostSize`	0	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.	Oracle HTTP Server Oracle iPlanet Web Server Apache HTTP Server Microsoft IIS
`MaxSkipTime`	10	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).	Oracle HTTP Server Oracle iPlanet Web Server Apache HTTP Server Microsoft IIS
`PathPrepend`	null	According to 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`.	Oracle HTTP Server Oracle iPlanet Web Server Apache HTTP Server Microsoft IIS

Table 7–1 (Cont.) General Parameters for Web Server Plug-Ins

Parameter Name	Default	Description	Applicable to
PathTrim	null	According to 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 the 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.	Oracle HTTP Server Oracle iPlanet Web Server Apache HTTP Server Microsoft IIS
QueryFromRequest	OFF	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.	Oracle HTTP Server Apache HTTP Server
WLDNSRefreshInterval	0 (Lookup once, during startup)	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 when 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.	Oracle HTTP Server Oracle iPlanet Web Server Apache HTTP Server
WLExcludePathOrMimeType	none	This parameter allows you make 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.	Oracle HTTP Server Oracle iPlanet Web Server Apache HTTP Server Microsoft IIS

Table 7–1   (Cont.) General Parameters for Web Server Plug-Ins

Parameter Name	Default	Description	Applicable to
WLFlushChunks	False	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.	Microsoft IIS
WLForwardUriUnparsed	OFF	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.	Oracle HTTP Server Apache HTTP Server
WLLocalIP	none	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.	Oracle HTTP Server Oracle iPlanet Web Server Apache HTTP Server
		If `WLLocalIP` is not set, If WLLocalIP is not set, the TCP/IP stack will choose the source IP address.	Microsoft IIS
WLLogFile **DEPRECATED**	See the `Debug` parameter	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.	Microsoft IIS
		For information on setting logging without using the deprecated parameter, see Section 3.4, "Deprecated Directives for Apache HTTP Server."	
WLMaxWebSocketClients	**Windows:** Half of ThreadsPerChild **Non-Windows:** Half of MaxClients	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 MaxClients (non-Windows). If the value specified for this parameter is greater than the maximum allowed, it will be automatically lowered to that maximum.	Oracle HTTP Server Apache HTTP Server
WLProxyPassThrough	OFF	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.	Oracle HTTP Server Oracle iPlanet Web Server Apache HTTP Server Microsoft IIS

Table 7–1   (Cont.) General Parameters for Web Server Plug-Ins

Parameter Name	Default	Description	Applicable to
WLProxySSL	OFF	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.	Oracle HTTP Server Oracle iPlanet Web Server Apache HTTP Server Microsoft IIS
WLProxySSLPassThrough	OFF	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).	Oracle HTTP Server Oracle iPlanet Web Server Apache HTTP Server Microsoft IIS
WLRetryAfterDroppedConnection	ALL	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.	Apache HTTP Server Oracle HTTP Server Oracle iPlanet Web Server Microsoft IIS
WLSendHdrSeparately	ON	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.	Microsoft IIS
WLServerInitiatedFailover	ON	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.	Oracle HTTP Server Oracle iPlanet Web Server Apache HTTP Server Microsoft IIS

Table 7–1   (Cont.) General Parameters for Web Server Plug-Ins

Parameter Name	Default	Description	Applicable to
WLSocketTimeoutSecs	2 (must be greater than 0)	Set the timeout for the socket while connecting, in seconds. See `ConnectTimeoutSecs` and `ConnectRetrySecs` for additional details.	Oracle HTTP Server Oracle iPlanet Web Server Apache HTTP Server Microsoft IIS

WLSRequest	OFF	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
			Oracle HTTP Server
			Apache HTTP Server

WLTempDir	See the Debug parameter	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.	Oracle HTTP Server Oracle iPlanet Web Server Apache HTTP Server
		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.	Microsoft IIS

7.1.1 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.

7.2 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 shown in Table 7–2. Parameters are case sensitive.

Table 7–2   SSL Parameters for Web Server Plug-Ins

Parameter	Default	Description	Applicable to
SecureProxy	OFF	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.	Oracle HTTP Server Oracle iPlanet Web Server Apache HTTP Server Microsoft IIS
WebLogicSSLV ersion	The best protocol supported by both the plug-in and WebLogic Server.	Selects 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 webserver'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.	Oracle HTTP Server Apache HTTP Server
WLSSLWallet	none	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 "${ORACLE_ INSTANCE}/config/fmwconfig/components/${COMP ONENT_TYPE}/instances/${COMPONENT_ NAME}/keystores/default"`	Oracle HTTP Server Oracle iPlanet Web Server Apache HTTP Server Microsoft IIS