Oracle® Fusion Middleware

Using Web Server 1.1 Plug-Ins with Oracle WebLogic Server

11*g* Release 1 (11.1.1)

E37889-03

December 2016

This document describes the use of version 1.1 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 Web Server 1.1 Plug-Ins with Oracle WebLogic Server, 11*g* Release 1 (11.1.1)

E37889-03

Primary Author: Srinivas Sudhindra

Contributors: Seema Alevoor, Jeff Trawick, Yulong Shi, Edwin Spear

Contents

7 Parameters for Web Server Plug-Ins

Preface

This preface describes the document accessibility features and conventions used in this guide—*Using Web Server 1.1 Plug-Ins with Oracle WebLogic Server*.

Documentation Accessibility

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

Access to Oracle Support

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

Conventions

The following text conventions are used in this document:

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

1 Overview of Web Server 1.1 Plug-Ins

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

1.1 What are Web Server 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: Client connections are managed by the web server.

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 Version 1.1 Plug-Ins

Version 1.1 plug-ins are available for the following web servers:

Table 1–1 Availability of Version 1.1 Plug-Ins

Web Server	Plug-In Availability	More Information
Oracle HTTP Server 11gR1	The plug-in is included in the Oracle HTTP Server installation. For information about configuring this plug-in, see Chapter 2, "Configuring the mod_wl_ohs Plug-In for Oracle HTTP Server."	See Chapter 2, "Configuring the mod_wl_ohs 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) 6.0 through 7.5	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` or `lib\*.dll` (native libraries) ■ `bin/orapki` or `bin\orapki.cmd` (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 3, "Installing and Configuring the Oracle iPlanet Web Server Plug-In" ■ Chapter 4, "Installing and Configuring the Apache HTTP Server Plug-In" ■ Chapter 5, "Installing and Configuring the Microsoft IIS Plug-In"

1.3 Upgrading from 1.0 Plug-Ins

The version 1.0 plug-ins described in *Using Web Server Plug-Ins with Oracle WebLogic Server* are deprecated and are not guaranteed to be available for future versions of Oracle WebLogic Server.

The version 1.1 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.

The version 1.1 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-certification-100350.html`

- If you have been using 128-bit encryption, you need to 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 need to change `mod_wl128_22.so` to `mod_wl.so`.

1.4 Features of the Version 1.1 Plug-Ins

This section describes the additional features of the version 1.1 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 1.1 Plug-Ins Use Oracle Security Framework"
- Section 1.4.3, "Version 1.1 Plug-Ins Support IPv6"
- Section 1.4.4, "Version 1.1 Plug-Ins Support Two-Way SSL"

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 1.1 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 need to change your configuration file to reflect the new naming convention. For example, you need to change `mod_wl128_22.so` to `mod_wl.so`.

1.4.2 Version 1.1 Plug-Ins Use Oracle Security Framework

The version 1.1 plug-ins use the Oracle certified security framework, and can therefore use Oracle wallets to store SSL configuration information.

For this reason, the version 1.1 plug-ins introduce an SSL configuration parameter `WLSSLWallet` to use Oracle wallets.

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.

1.4.3 Version 1.1 Plug-Ins Support IPv6

The version 1.1 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 1.1 Plug-Ins Support Two-Way SSL

The version 1.1 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."

1.5 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 Plugin version 1.1, <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 mod_wl_ohs Plug-In for Oracle HTTP Server

This chapter describes how to configure mod_wl_ohs, which is the plug-in for proxying requests from Oracle HTTP Server to Oracle WebLogic server. The mod_wl_ohs module is included in the Oracle HTTP Server installation. You need not download and install it separately.

> **Note:** mod_wl_ohs provides features that are identical to those of the plug-in for Apache HTTP Server.

You can configure mod_wl_ohs either by using Fusion Middleware Control or by editing the `mod_wl_ohs.conf` configuration file manually.

This chapter contains the following topics:

2.1 Prerequisites for Configuring mod_wl_ohs

Before you begin configuring mod_wl_ohs, 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.
- Verify that Fusion Middleware Control has been installed and you can access the Enterprise Manager Console. This is required if you want configure mod_wl_ohs by using the graphical interface provided by Fusion Middleware Control.
- To be able to test the configuration, make sure that 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.

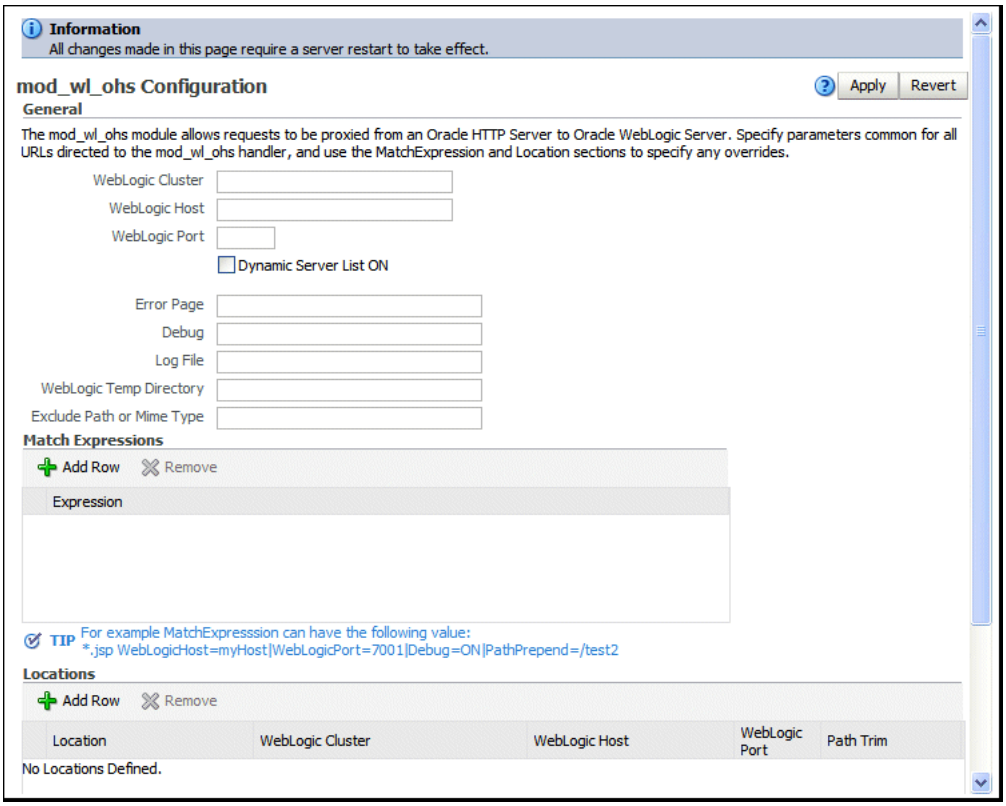
4. Scroll down to the Advanced section, expand it, and select the **WebLogic Plug-In Enabled** check box.
5. If you selected **Servers** in step 2, repeat steps 3 and 4 for the other servers to which you want to proxy requests from Oracle HTTP Servers.
6. Click **Save**.

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

2.2 Configuring mod_wl_ohs Using Fusion Middleware Control

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

1. Make sure that you have fulfilled the prerequisites listed in Section 2.1.
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 is displayed.



4. Specify the configuration settings as described in the online help, which you can invoke by clicking the help icon on the page.

5. Review the settings.

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

6. Restart Oracle HTTP Server 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 Configuring mod_wl_ohs Manually

You can configure mod_wl_ohs manually by specifying directives in the `mod_wl_ohs.conf` file.

1. Make sure that you have fulfilled the prerequisites listed in Section 2.1.
2. Open the `mod_wl_ohs.conf` file, which is located in the following directory, in a text editor:

   ORACLE_INSTANCE`/config/OHS/`*component_name*

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 mod_wl_ohs module 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>
     SetHandler weblogic-handler
     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>
SetHandler weblogic-handler
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>
  SetHandler weblogic-handler
  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>
  SetHandler weblogic-handler
  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 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>
  SetHandler weblogic-handler
  WebLogicHost host
  WeblogicPort port
  </Location>

  #For an application running on a cluster
  <Location path2>
  SetHandler weblogic-handler
  WebLogicCluster host:port,host:port,...
  </Location>

  </IfModule>
  ```

  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 the following command:

```
> $ORACLE_INSTANCE/bin/opmnctl startproc ias-component=component_name
```

2.4 Configuring SSL for mod_wl_ohs

For information about configuring mod_wl_ohs to support one-way and two-way SSL between Oracle HTTP Server and Oracle WebLogic Server, see "Enable SSL for Outbound Requests from Oracle HTTP Server" in the *Oracle Fusion Middleware Administrator's Guide*.

3

Installing and Configuring the Oracle iPlanet Web Server Plug-In

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

This chapter contains the following sections:

3.1 Overview of the Oracle iPlanet Web Server Plug-In

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

The Oracle iPlanet Web Server plug-in is designed for an environment where Oracle iPlanet Web Server serves static pages, and an Oracle WebLogic Server instance (operating in a different process, possibly on a different machine) is delegated to serve dynamic pages, such as JSPs or pages generated by HTTP Servlets. The connection between WebLogic Server and the Oracle iPlanet Web Server plug-in 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 iPlanet Web Server plug-in, providing access to all WebLogic Server services (not just dynamic pages).

The Oracle iPlanet Web Server plug-in operates as a module within a 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://download.oracle.com/docs/cd/E18958_01/doc.70/e18789/chapter.htm

3.2 Installing and Configuring the Oracle iPlanet Web Server Plug-In

The following sections provide information pertaining to the installation prerequisites and configuring the Oracle iPlanet Web Server Plug-in.

3.2.1 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-1.1/)
- Download the Oracle iPlanet Web Server plug-in, as described in Section 1.2, "Availability of Version 1.1 Plug-Ins."
- 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 3–1 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)	Helper libraries

- Installed JDK 7 if you want to use SSL. You must have a JDK 7 installation if you want 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.
- Created a supported Oracle 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 1.1 plug-in is supported on the Oracle iPlanet Web Server platforms described in:
  http://www.oracle.com/technology/software/products/ias/files/fusion_certification.html
- Created an iPlanet instance location (INSTANCE-DIR; for example, ${IPLANET_HOME}/https-foo.
- Created a supported version of WebLogic Server is configured and running on a target system. Note that this server does not need to run on the system to which you extracted the plug-in zip distribution. For the supported WebLogic Server versions, see:

  http://www.oracle.com/technology/software/products/ias/files/fusion_certification.html

3.2.2 Installing Oracle iPlanet Web Server Plug-In

The Oracle iPlanet Web Server plug-in is distributed as a shared object (`.so`) for Unix platforms and as a DLL (.dll) for Windows.

To instruct Oracle iPlanet Web Server to load the native library (on Unix or Windows) 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

3.2.3 Configuring the Oracle iPlanet Web Server Plug-In

This section provides information about configuring the Oracle iPlanet Web Server plug-in.

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

3.2.3.1 Proxying Requests by URL

If you want 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 iPlanet Web Server plug-in. For a complete list of plug-in parameters, see Section 8.2, Section 7.1, "General Parameters for Web Server Plug-Ins"

3.2.3.2 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://download.oracle.com/docs/cd/E19146-01/821-1828/gdabr/index.html

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

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

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

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

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

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

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

After adding entries for the Oracle iPlanet Web Server plug-in, 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 need to disable Java from the Oracle 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 wish 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 wish to handle HTTP tunneling requests.

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

3.2.3.3 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 iPlanet Web Server is already running, you must either restart or reconfigure the server.
3. You can test the Oracle 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*

3.2.4 Example: Configuring the iPlanet Plug-in

The following example demonstrates basic instructions for quickly setting up the iPlanet plug-in 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>
   ```

```
Debug="ALL" WLLogFile="C:\Temp\wl-proxy.log" DebugConfigInfo="ON"
PathTrim="/wls"
</Object>
...
```

For more information on configuring the contents of obj.conf, see Section 3.2.6, "Sample obj.conf File (Not Using a WebLogic Cluster)" and Section 3.2.7, "Sample obj.conf File (Using a WebLogic Cluster)".

3. At the prompt, include the %PLUGIN_HOME%\lib in the PATH by entering:

```
set PATH=C:\myhome\weblogic-plugin-1.1\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.

3.2.5 Guidelines for Modifying the obj.conf File

To use the Oracle iPlanet Web Server plug-in, you must make several modifications to the `obj.conf` file. For more information, see

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

3.2.6 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---------------------
```

3.2.7 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

Installing and Configuring the Apache HTTP Server Plug-In

This chapter describes how to install and configure the Apache HTTP Server plug-in. It contains the following sections:

- Section 4.1, "Install the Apache HTTP Server Plug-In"
- Section 4.2, "Configure the Apache HTTP Server Plug-In"

> **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 mod_wl_ohs Plug-In for Oracle HTTP Server.".

4.1 Install the Apache HTTP Server Plug-In

After you download the Apache HTTP Server plug-in as described in Section 1.2, "Availability of Version 1.1 Plug-Ins,", you can install it as an Apache HTTP Server module in your Apache HTTP Server installation.

4.1.1 Installation Prerequisites

Before you install the Apache HTTP Server plug-in, do the following:

- Download the Apache HTTP Server plug-in, as described in Section 1.2, "Availability of Version 1.1 Plug-Ins."
- Plug-in zip extract location (PLUGIN_HOME; for example /home/myhome/weblogic-plugins-1.1/)
- Extract the plug-ins zip distribution to PLUGIN_HOME; for example, /home/myhome/weblogic-plugins-1.1/. This distribution contains these files:

Table 4–1 Files Included in the Apache Web Server Plug-in Zip

(path)/filename	Description
README.txt	This file
bin/orapki.bat	orapki tool for configuring Oracle wallets
jlib/*.jar	orapki helper Java libraries
lib/mod_wl.so	WebLogic proxy module

Table 4–1 (Cont.) Files Included in the Apache Web Server Plug-in Zip

(path)/filename	Description
lib/*.so(.dll)	Helper libraries

- Install JDK 7 if you want to use SSL. The JDK 7 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.

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

  For more information, see:

  http://www.oracle.com/technology/software/products/ias/files/fusion_certification.html.
- Ensure that a supported version of Oracle WebLogic Server is configured and running on a target system. Note that 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/technology/software/products/ias/files/fusion_certification.html.

4.1.2 Installing the Apache HTTP Server Plug-In

The Apache HTTP Server plug-in is distributed as a shared object (.so) file.

To install the Apache HTTP Server plug-in:

1. Make sure that the `weblogic-plugins-1.1/lib` folder is included in `LD_LIBRARY_PATH` on Unix systems (and PATH on Windows systems). If you do not do this, 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-1.1/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 on the UNIX/Linux platform:

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

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

   Note: The `apachectl` command is not available on the Windows platform.

   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. Install the Apache HTTP Server plug-in module for Apache 2.2.x by adding the following line. For Windows, specify the .dll file.

   ```
   LoadModule weblogic_module /home/myhome/weblogic-plugins-1.1/lib/mod_wl.so
   ```

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

     ```
     APACHE_HOME\bin> apachectl -t
     ```

   - 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
   ```

4.2 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.

4.2.1 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
    Debug ALL
    DebugConfigInfo ON
    WLLogFile /tmp/wl-proxy.log
   </IfModule>
   ```

3. To proxy requests by MIME type, add a MatchExpression line to the IfModule block. Note that if both MIME type and proxying by path are enabled, 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
 Debug ALL
 DebugConfigInfo ON
 WLLogFile /tmp/wl-proxy.log
</IfModule>
```

You can also use multiple `MatchExpressions`, for example:

```
<IfModule mod_weblogic.c>
 WebLogicHost my-weblogic.server.com
 WebLogicPort 7001
 MatchExpression *.jsp
 MatchExpression *.xyz
 Debug ALL
 DebugConfigInfo ON
 WLLogFile /tmp/wl-proxy.log
</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 `SetHandler` statement. `SetHandler` specifies the handler for the Apache HTTP Server plug-in module. For example the following `Location` block proxies all requests containing /weblogic in the URL:

   ```
   <Location /weblogic>
   SetHandler weblogic-handler
   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 Apache plug-in to use SSL

   - The following configuration is **incorrect**:

     ```
     <Location /weblogic>
      SetHandler weblogic-handler
     </Location>

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

The following configuration is the **correct** setup:

```
<Location /weblogic>
 SetHandler weblogic-handler
 PathTrim /weblogic
</Location>
```

- The current implementation of the WebLogic Server Apache plug-in does not support the use of multiple certificate files with Apache SSL.

6. Optionally, enable HTTP tunneling for t3 or IIOP protocol and `weblogic.jar`, add the following `Location` block to the `httpd.conf` file:.

```
<Location /bea_wls_internal/HTTPClnt>
 SetHandler weblogic-handler
</Location>
```

7. Define any additional parameters for the Apache HTTP Server plug-in.

   The Apache HTTP Server plug-in recognizes the parameters listed in Section 7.1, "General Parameters for Web Server Plug-Ins". To modify the behavior of your Apache HTTP Server plug-in, 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 on the UNIX/Linux platform:

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

   Note: The `apachectl` command is not available on the Windows platform.

   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.

   - Windows

```
APACHE_HOME\bin> httpd -k start
```

   - 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.

4.2.1.1 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>
SetHandler weblogic-handler
WebLogicHost myweblogic.server.com
WebLogicPort 7001
</Location>

<Location /weblogic>
SetHandler weblogic-handler
WebLogicCluster w1s1.com:7001,w1s2.com:7001,w1s3.com:7001
</Location>

<VirtualHost apachehost:80>
SetHandler weblogic-handler
WebLogicServer weblogic.server.com
WebLogicPort 7001
</VirtualHost>
```

4.2.1.2 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-1.1/lib/mod_wl.so

  <IfModule mod_weblogic.c>
    WebLogicHost wls-host
    WebLogicPort wls-port
    Debug ALL
    WLLogFile /tmp/wl-proxy.log
  </IfModule>

  <Location /mywebapp>
    SetHandler weblogic-handler
  </Location>
  ...
```

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

```
$ export LD_LIBRARY_PATH=/home/myhome/weblogic-plugin-1.1/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 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

4.2.2 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.

4.2.2.1 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/).
- If you want to have only one log file for all the virtual hosts configured in your environment, you can achieve it using global properties. Instead of specifying the same `Debug`, `WLLogFile` and `WLTempDir` properties in each virtual host you can specify them just once in the `<IfModule>` tag.
- Sample `httpd.conf` file:

  ```
  <IfModule mod_weblogic.c>
   WebLogicCluster johndoe02:8005,johndoe:8006
   Debug ON
   WLLogFile c:/tmp/global_proxy.log
   WLTempDir "c:/myTemp"
   DebugConfigInfo ON
   KeepAliveEnabled ON
   KeepAliveSecs 15
  </IfModule>

  <Location /jurl>
   SetHandler weblogic-handler
   WebLogicCluster agarwalp01:7001
  </Location>

  <Location /web>
   SetHandler weblogic-handler
   PathTrim /web
   Debug OFF
   WLLogFile c:/tmp/web_log.log
  ```

```
</Location>

<Location /foo>
 SetHandler weblogic-handler
 PathTrim /foo
 Debug ERR
 WLLogFile c:/tmp/foo_proxy.log
</Location>
```

- All the requests which match /jurl/* will have Debug Level set to ALL and log messages will be logged to `c:/tmp/global_proxy.log` file. All the requests which match /web/* will have Debug Level set to OFF and no log messages will be logged. All the requests which match /foo/* will have Debug Level set to ERR and log messages will be logged to `c:/tmp/foo_proxy.log` file.
- Oracle recommends that you use the `MatchExpression` statement instead of the `<Files>` block.

4.2.2.2 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 4–1 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 4–2, 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 WebLogicCluster hosts and their ports. The `paramName=value` combination following the pipe symbol specifies the error page for the cluster.

Example 4–2 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 4–3 shows an example without WebLogic clusters.

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

Example 4–4 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.

4.2.2.3 Template for the Apache HTTP Server httpd.conf File

This section contains a sample `httpd.conf` file for Apache 2.2. You can use this sample as a template and modify it to suit your environment and server. Lines beginning with # are comments.

Note that Apache HTTP Server is not case sensitive.

Example 4–5 Sample httpd.conf file for Apache 2.2

```
##################################################
APACHE-HOME/conf/httpd.conf file
##################################################
LoadModule weblogic_module lhome/myhome/weblogic-plugins-1.1/lib/mod_wl.so

<Location /weblogic>
 SetHandler weblogic-handler
 PathTrim /weblogic
 ErrorPage http://myerrorpage1.mydomain.com
</Location>

<Location /servletimages>
 SetHandler weblogic-handler
 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>
```

5

Installing and Configuring the Microsoft IIS Plug-In

The following sections describe how to install and configure the Microsoft Internet Information Server plug-in:

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

To install the Microsoft Internet Information Server plug-in:

1. Download the Microsoft Internet Information Server plug-in, as described in Section 1.2, "Availability of Version 1.1 Plug-Ins." The zip file contains these files:

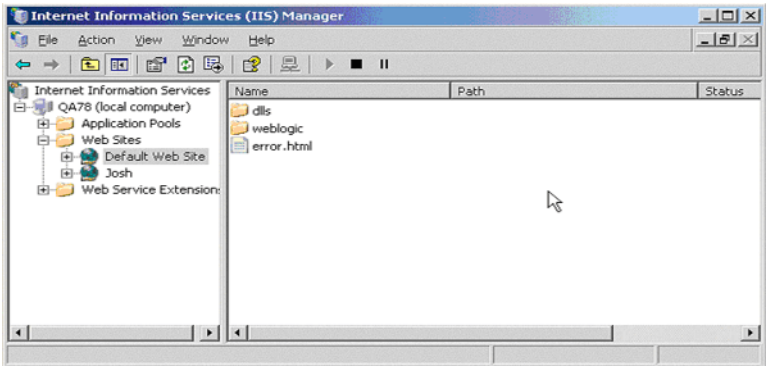
Table 5–1 Files Included in the Microsoft IIS Plug-In Zip

(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

2. Copy the iisproxy.dll file into a convenient directory that is accessible to IIS. This directory must also contain the `iisproxy.ini` file that you will create in step 6.

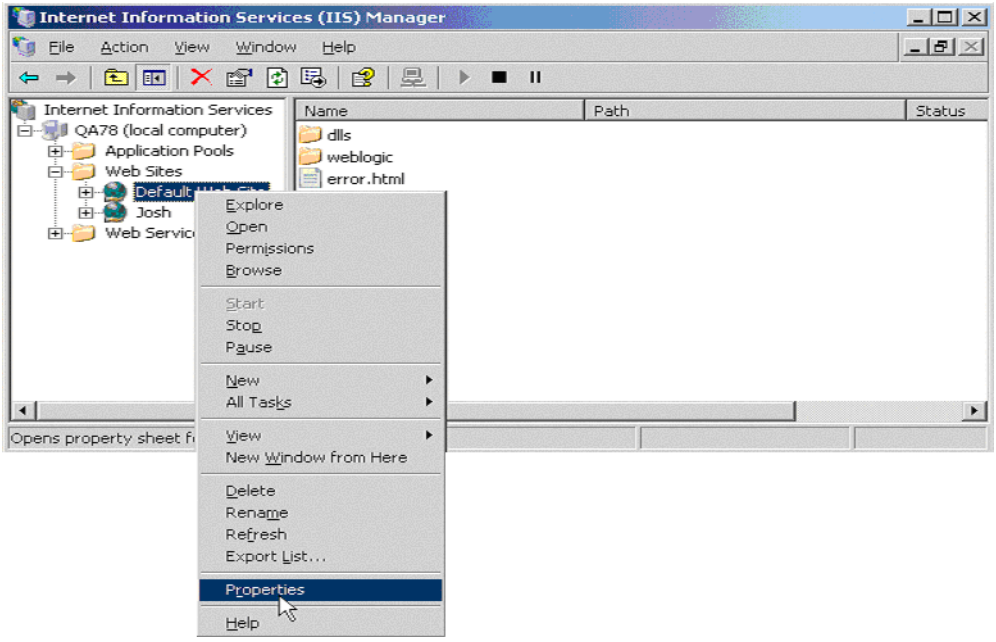
3. Set the user permissions for the `iisproxy.dll` file to include the name of the user who will be running IIS. One way to do this is by right clicking on the `iisproxy.dll` file and selecting Permissions, then adding the username of the person who will be running IIS.
4. If you want to configure proxying by file extension (MIME type) complete this step. (You can configure proxying by path in addition to or instead of configuring by MIME type. See step 5.)
   a. Start the Internet Information Service Manager by selecting it from the Start menu.
   b. In the left panel of the Service Manager, select your Web site (the default is "Default Web Site").

Figure 5–1 Selecting Web Site in Service Manager



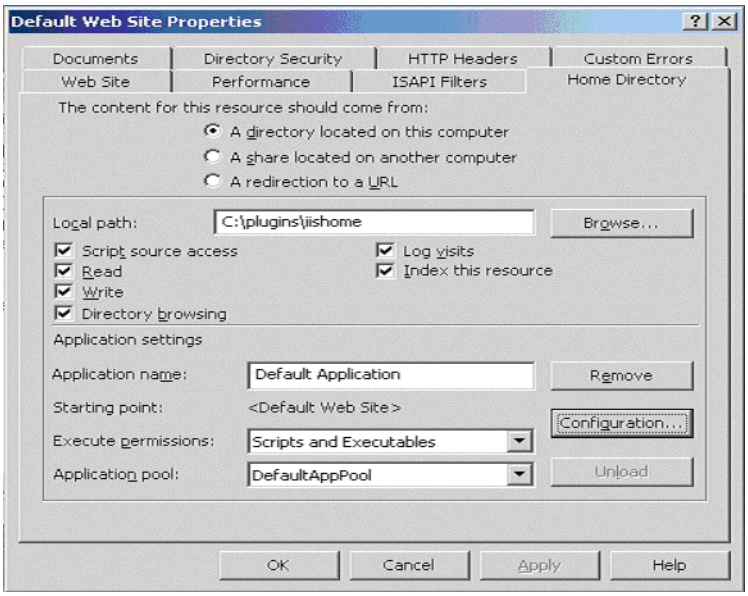
   c. Click the "Play" arrow in the toolbar to start.
   d. Open the properties for the selected Web site by right-clicking the Web site selection in the left panel and selecting Properties.

Figure 5–2 Selecting Properties for Selected Web Site



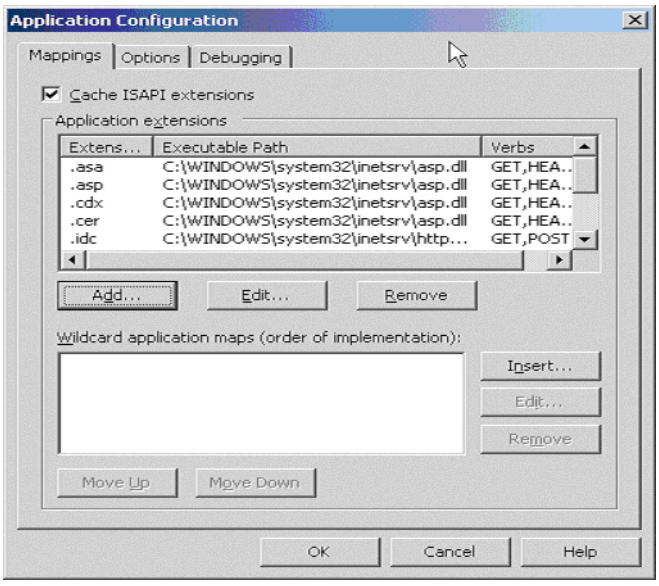
e. In the Properties panel, select the Home Directory tab, and click the Configuration button in the Applications Settings section.

Figure 5–3 Home Directory Tab of the Properties Panel



f. On the Mappings tab, click the Add button to add file types and configure them to be proxied to WebLogic Server.

Figure 5–4 Click the Add Button to Add File Types



g. In the Add dialog box, browse to find the `iisproxy.dll` file.

h. Set the Extension to the type of file that you want to proxy to WebLogic Server.

i. If you are configuring for IIS 6.0 or later, be sure to deselect the **Check that file exists** check box. The behavior of this check has changed from earlier versions of IIS: it used to check that the iisproxy.dll file exists; now it checks that files requested from the proxy exist in the root directory of the Web server. If the

check does not find the files there, the iisproxy.dll file will not be allowed to proxy requests to the WebLogic Server.

j. In the Directory Security tab, set the Method exclusions as needed to create a secure installation.

k. When you finish, click **OK** to save the configuration. Repeat this process for each file type you want to proxy to WebLogic.

l. When you finish configuring file types, click the OK button to close the Properties panel.

Note: In the URL, any path information you add after the server and port is passed directly to WebLogic Server. For example, if you request a file from IIS with the URL:

```
http://myiis.com/jspfiles/myfile.jsp
```

it is proxied to WebLogic Server with a URL such as
`http://mywebLogic:7001/jspfiles/myfile.jsp`

Note: To avoid out-of-process errors, ensure **Cache ISAPI Applications** is selected.

---

5. If you want to configure proxying by path, see Section 5.4, "Using Wildcard Application Mappings to Proxy by Path".

6. In WebLogic Server, create the `iisproxy.ini` file.

   The `iisproxy.ini` file contains `name=value` pairs that define configuration parameters for the plug-in. The parameters are listed in Section 7–1, " General Parameters for Web Server Plug-Ins".

   Use the example `iisproxy.ini` file in Section 5.5.1, "Sample iisproxy.ini File" as a template for your `iisproxy.ini` file.

   Note: Changes in the parameters will not go into effect until you restart the “IIS Admin Service” (under services, in the control panel).

   Oracle recommends that you locate the `iisproxy.ini` file in the same directory that contains the `iisproxy.dll` file. You can also use other locations. If you place the file elsewhere, note that WebLogic Server searches for `iisproxy.ini` in the following directories, in the following order:

   a. In the same directory where `iisproxy.dll` is located.

   b. In the home directory of the most recent version of WebLogic Server that is referenced in the Windows Registry. (If WebLogic Server does not find the `iisproxy.ini` file in the home directory, it continues looking in the Windows Registry for older versions of WebLogic Server and looks for the `iisproxy.ini` file in the home directories of those installations.)

   c. In the directory `c:\weblogic`, if it exists.

7. Define the Oracle WebLogic Server host and port number to which the Microsoft Internet Information Server plug-in proxies requests. Depending on your configuration, there are two ways to define the host and port:

- If you are proxying requests to a single WebLogic Server, define the WebLogicHost and WebLogicPort parameters in the `iisproxy.ini` file. For example:

  ```
  WebLogicHost=localhost
  WebLogicPort=7001
  ```

- If you are proxying requests to a cluster of WebLogic Servers, define the WebLogicCluster parameter in the `iisproxy.ini` file. For example:

  ```
  WebLogicCluster=myweblogic.com:7001,yourweblogic.com:7001
  ```

  Where myweblogic.com and yourweblogic.com are instances of Oracle WebLogic Server running in a cluster.

8. Optionally, enable HTTP tunneling by following the instructions for proxying by path (see Section 5.4, "Using Wildcard Application Mappings to Proxy by Path") substituting the WebLogic Server host name and the WebLogic Server port number, or the name of a WebLogic Cluster that you wish to handle HTTP tunneling requests.
9. Set any additional parameters in the `iisproxy.ini` file. A complete list of parameters is available in the appendix Section 7.1, "General Parameters for Web Server Plug-Ins".
10. If you are proxying servlets from IIS to WebLogic Server and you are not proxying by path, see Section 5.4, "Using Wildcard Application Mappings to Proxy by Path".
11. The installed version of IIS with its initial settings does not allow the `iisproxy.dll`. Use the IIS Manager console to enable the plug-in:
    a. Open the IIS Manager console.
    b. Select **Web Service Extensions**.
    c. Set **All Unknown ISAPI Extensions** to Allowed.

5.1.1 Example: Configuring the IIS Plug-In

The following example describes how to set up the IIS plug-in to proxy requests to a backend WebLogic Server (WLS).

1. Create iisproxy.ini file in%PLUGIN_HOME%\lib\. Include the following lines:

   ```
   WebLogicHost=wls-host
   WebLogicPort=wls-port
   Debug=ALL
   WLLogFile=C:\Temp\wl-proxy.log
   ```

2. Ensure that the %PLUGIN_HOME%\lib is included in the system PATH (**Control-Panel** > **System** > **System Properties** > **Environment Variables** > **System Properties** > **PATH**).
3. Open IIS Manager, use Default Web Site or create a Web Site. Click the site, open **Handler Mappings** and add a script map (set the **Extension**, for example '*.jsp' or '*', set **Executable** to %PLUGIN_HOME%\lib\iisproxy.dll, and assign a **Name**)
4. Start IIS.
5. Send a request to `http://iis-host:iis-port/mywebapp/my.jsp` from the browser. Validate the response.

5.2 Installing and Configuring the Microsoft Internet Information Server Plug-In for IIS 7.0

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:

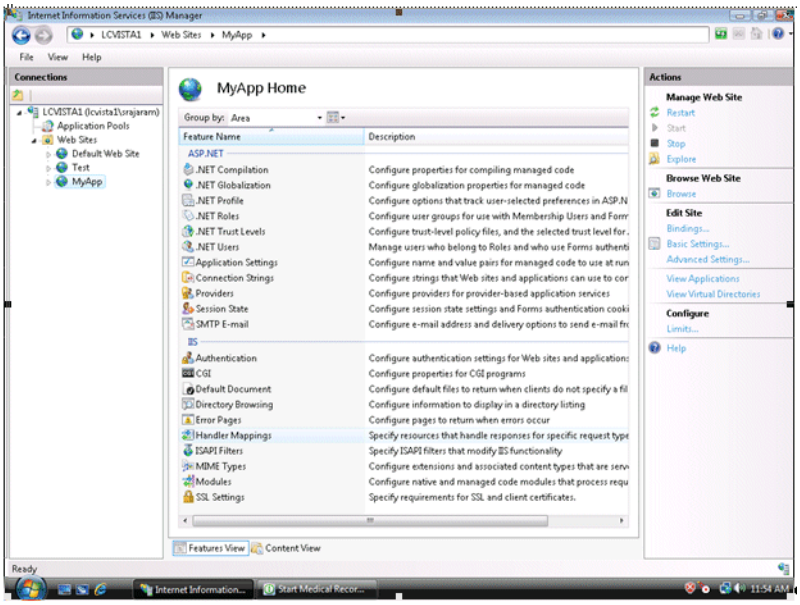
1. Create a web application in IIS Manager by right clicking on **Web Sites** > **Add Web Site**.

   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.

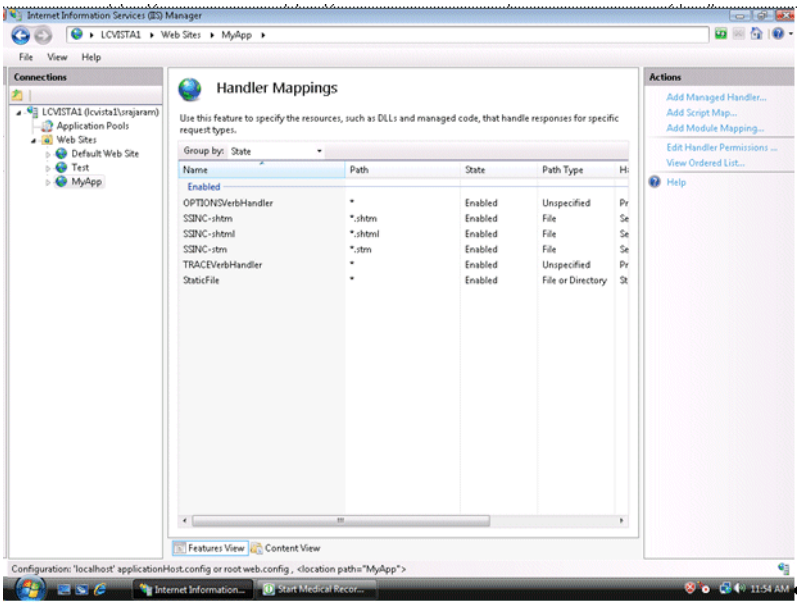
   If you can see the name of your application under Web Sites it means that your application has been created and started running. Click the MyApp node under Web Sites to see all of the settings related to the MyApp application, which you can change, as shown in Figure 5–5.

Figure 5–5 Application Home Page



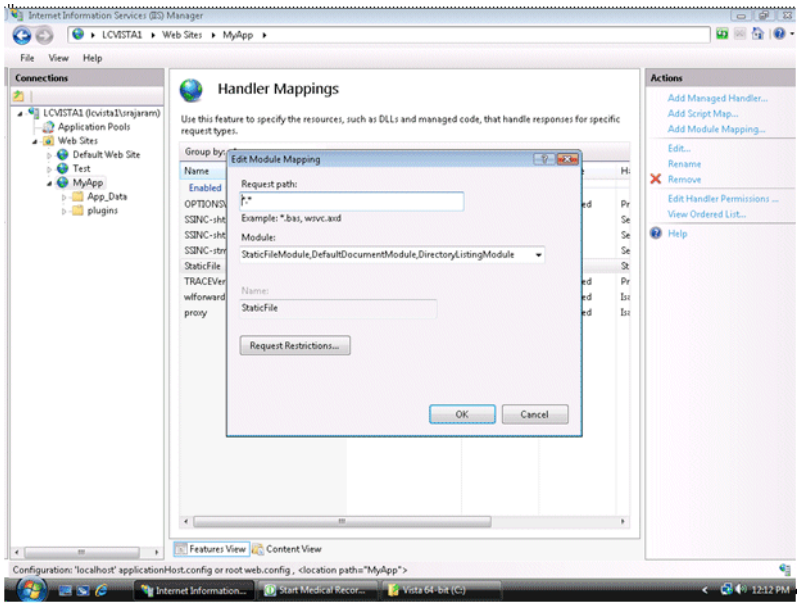
2. Click **Handler Mappings** to set the mappings to the handler for a particular MIME type.

Figure 5–6 Setting the Handler Mappings



3. Click the StaticFile and change the Request path from * to *.*. Click **OK**.

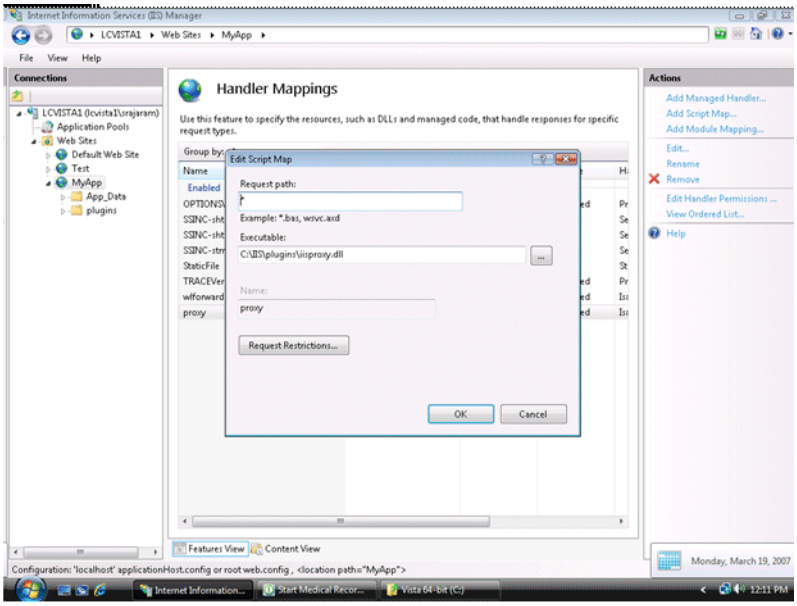
Figure 5–7 Editing the Request Path for Module



4. Click *MyApp* and then click **Add Script Map** on the right-hand side menu options. Enter * for the Request path.

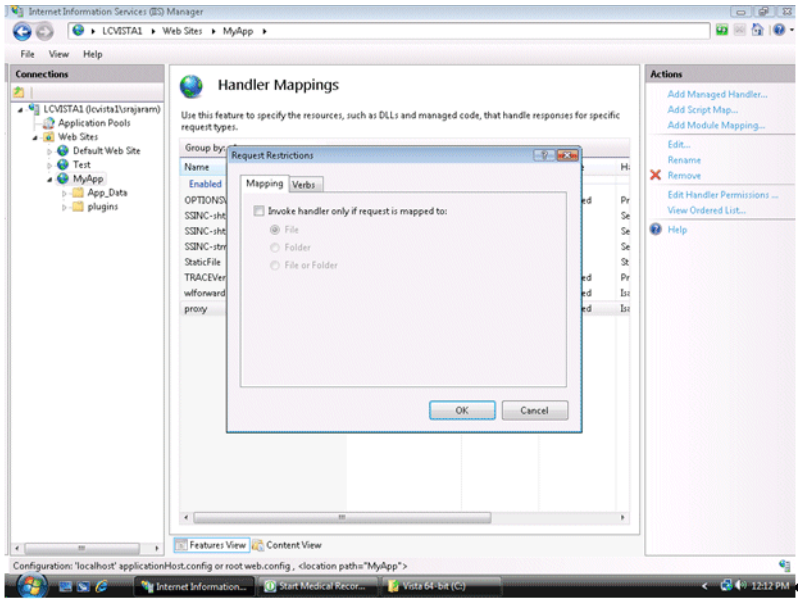
   Browse to the `iisproxy.dll` file and add it as the executable. Name it `proxy`.

Figure 5–8 Editing the Request Path for Script



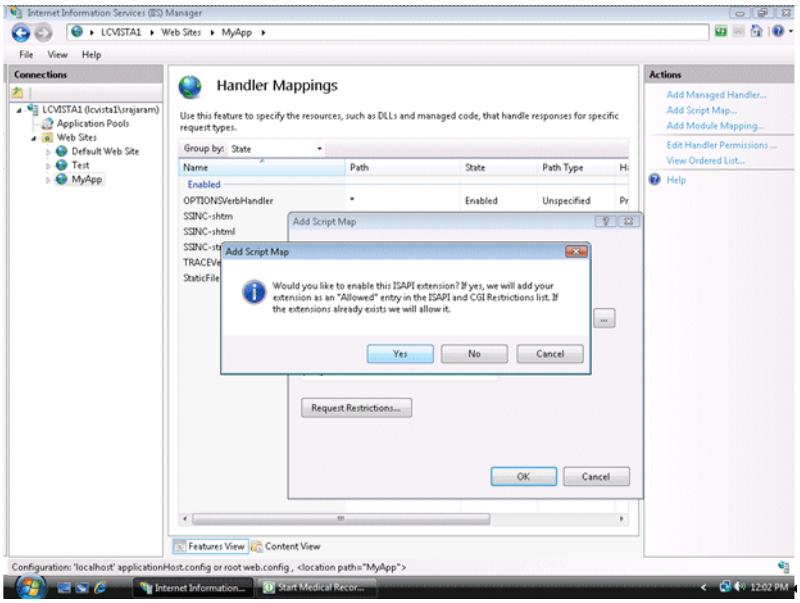
5. Click **Request Restrictions** and deselect **Invoke handler only if the request is mapped to**.

Figure 5–9 Editing the Request Restrictions



6. Click **OK** to add this Handler mapping. Click **Yes** on the Add Script Map dialog box.

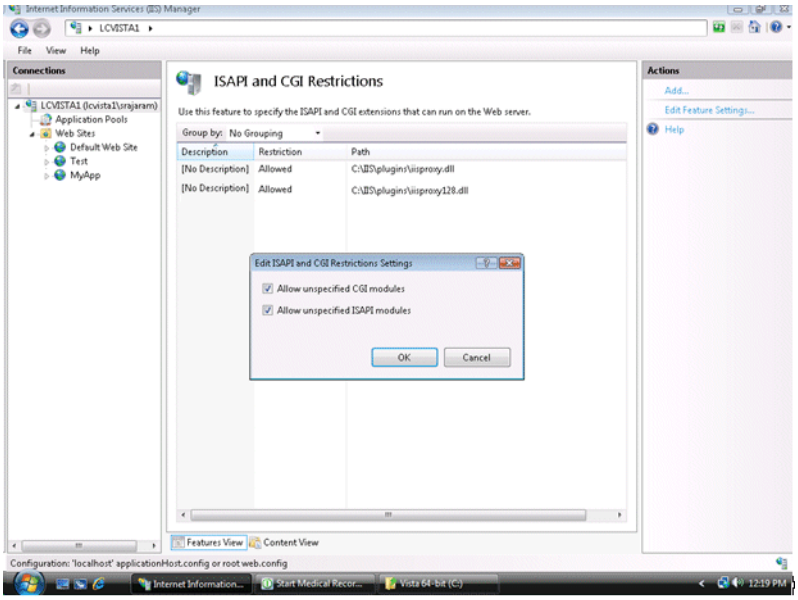
Figure 5–10 Adding the Script Map



7. If you want to configure proxying by path, see Section 5.4, "Using Wildcard Application Mappings to Proxy by Path".

8. 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 5–11 Editing ISAPI and CGI Restrictions



9. 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
```

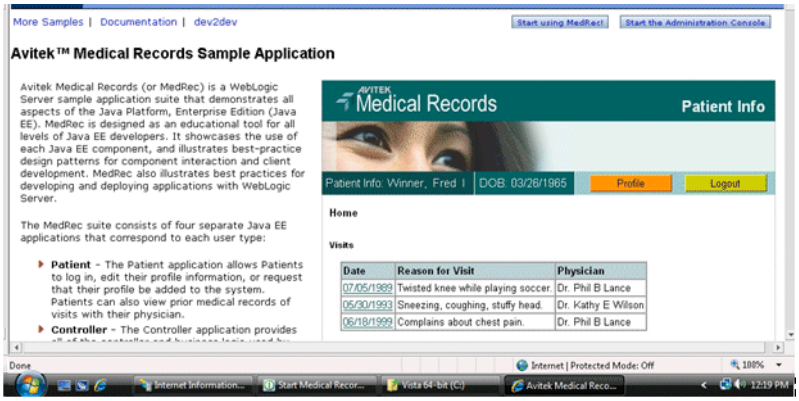
```
DebugConfigInfo=ON
KeepAliveEnabled=true

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

10. 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.

    If you want to run the plug-in in SSL mode, change the value of `WeblogicPort` to the SSL port of your application, and change the `SecureProxy` value to ON.

Figure 5–12 Medrec Sample Application



5.3 Serving Static Files from the Web Server

In order to have IIS 7.5 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 WebLogic Server plug-in 1.1 on IIS 7.5 Web Server as described in Section 5.1.

   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. To have IIS 7.5 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. Now, 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.

5.4 Using Wildcard Application Mappings to Proxy by Path

As described in "Installing Wildcard Application Mappings (IIS 6.0)" (http://www.microsoft.com/technet/prodtechnol/WindowsServer2003/Library/IIS/5c5ae5e0-f4f9-44b0-a743-f4c3a5ff68ec.mspx?mfr=true), and "Add a Wildcard Script Map" for IIS 7.5 (http://technet.microsoft.com/en-us/library/cc754606(WS.10).aspx), you can configure a Web site or virtual directory to run an Internet Server API (ISAPI) application at the beginning of every request to that Web site or virtual directory, regardless of the extension of the requested file. You can use this feature to insert a mapping to `iisproxy.dll` and thereby proxy requests by path to WebLogic Server.

5.4.1 Installing Wildcard Application Mappings (IIS 6.0)

The following steps summarize the instructions available at "Installing Wildcard Application Mappings (IIS 6.0)" (http://www.microsoft.com/technet/prodtechnol/WindowsServer2003/Library/IIS/5c5ae5e0-f4f9-44b0-a743-f4c3a5ff68ec.mspx?mfr=true) for adding a wildcard application mapping to a Web server or Web site in IIS 6.0:

1. In IIS Manager, expand the local computer, expand the Web Sites folder, right-click the Web site or virtual directory that you want, and then click Properties.
2. Click the appropriate tab: Home Directory, Virtual Directory, or Directory.
3. In the Application settings area, click Configuration, and then click the Mappings tab.
4. To install a wildcard application map, do the following:
   a. On the Mappings tab, click Insert.
   b. Type the path to the `iisproxy.dll` DLL in the Executable text box or click Browse to navigate to.
   c. Click OK.

5.4.2 Adding a Wildcard Script Map for IIS 7.5

The following steps summarize the instructions available at "Add a Wildcard Script Map" for IIS 7.5 (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 7.5:

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 it. 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.

5.5 Proxying Requests from Multiple Virtual Web Sites to WebLogic Server

To proxy requests from multiple Web sites (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 Web sites, 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 Web sites 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.

5.5.1 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.

5.6 Creating ACLs Through IIS

ACLs will not work through the Microsoft IIS plug-in 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 NT configurations appear to be correct.
- A user who has local log-on rights and who can obtain physical access to the host computer running IIS will be permitted to start an interactive session at the console.

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

5.7 Testing the Installation

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://myii.server.com/filename.jsp
   ```

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

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:

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. This is a new parameter for the 11.1.1.9 release and is not available for earlier releases of Web Server Plug-ins.
- `WLSSLWallet`—The version 1.1 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 Administrator's Guide for 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 11gR1 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 need to make sure 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 1.1 plug-ins are installed.

> **Note:** The examples in this section use the WebLogic Server demo CA. If you are using the plug-in in a production environment, make sure that trusted CAs are properly configured for the plug-in as well as 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 Apache plug-in 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 Apache plug-in or IIS plug-in is not the same user who creates the wallet (or for Windows, the account SYSTEM), you need to grant the user access to the wallet by running the command `cacls` (Windows) or `chmod` (UNIX) after you create the wallet. For example:

   IIS 6.0:

   ```
   cacls <wallet_path>\cwallet.sso /e /g IUSR_<Machine_Name>:R
   ```

   IIS 7.5:

   ```
   cacls <wallet_path>\cwallet.sso /e /g IUSR:R
   ```

   ---

3. Import the *`WL_HOME\server\lib\CertGenCA.der`* CA into the Oracle Wallet.

   ```
   orapki wallet add -wallet mywallet -trusted_cert -cert CertGenCA.der -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. Select the **WebLogic Plug-In Enabled** check box.
   f. Select the **Client Cert Proxy Enabled** check box.
   g. 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.
   h. 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 1.1 plug-ins are installed.

1. From the Oracle wallet, generate a certificate request.
2. Use this certificate request to create a certificate via 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 1.1 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` 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 via 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-->Configuration-->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 in order to use SSL between the plug-in and WebLogic Server. See Section 6.1, "Use SSL with Plug-Ins" for the steps you need to perform to configure SSL.

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

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 NT, set the `TcpTimedWaitDelay` on the proxy and WebLogic Server servers to a lower value. Set the TIME_WAIT interval in Windows NT by editing the registry key under HKEY_LOCAL_MACHINE:

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

    If this key does not exist you can create it as a DWORD value. The numeric value is the number of seconds to wait and may be set to any value between 30 and 240. If not set, Windows NT defaults to 240 seconds for TIME_WAIT.
  - On Windows 2000, lower the value of the `TcpTimedWaitDelay` by editing the registry key under HKEY_LOCAL_MACHINE:

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

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

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

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

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

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

  ```
  tcp_conn_req_max_q
  tcp_conn_req_max_q0
  ```

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 in your `httpd.conf` or `weblogic.conf` file 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

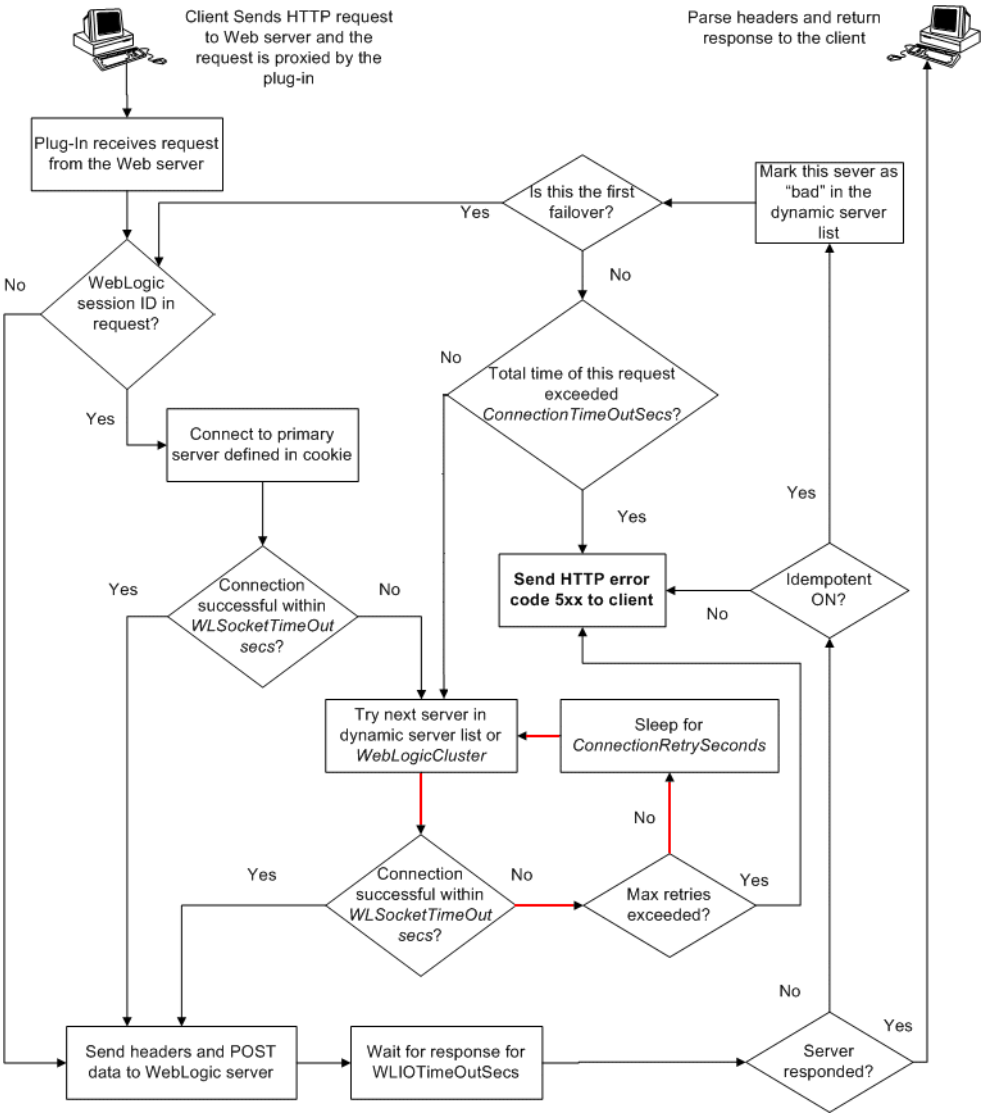
cluster. The updated list adds any new servers in the cluster and deletes any that are no longer part of the cluster or that have failed to respond to requests. This list is updated automatically with the HTTP response when a change in the cluster occurs.

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 Oracle iPlanet Web Server Plug-in

You can use the Secure Sockets Layer (SSL) protocol to protect the connection between the Oracle 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 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 or not the SSL protocol will be used to protect the connection between the Oracle iPlanet Web Server plug-in 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 iPlanet Web Server Plug-In 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 Configuring SSL with WebLogic Proxy Plug-In and Oracle WebLogic Server

WebLogic Proxy Plug-In 11.1.1.9 supports additional SSL protocols such as TLS 1.1 and TLS 1.2 and will default to these secure protocols during the SSL handshake with WebLogic Server. When WebLogic Server Proxy Plug-In 11.1.1.9 front-ends HTTPS traffic from WebLogic Server, then administrators must do one of the following:

- Ensure that WebLogic Server supports TLS 1.2 protocol.

  WebLogic Server 10.3.x must be explicitly configured to use TLS 1.2 protocol. For more information see:

  - "Transport Layer Security (TLS) 1.2 Support" in *What's New in Oracle WebLogic Server* and Oracle Support docs for more information.
  - "Enabling and Disabling the JSSE-Based SSL Implementation" in *Securing Oracle WebLogic Server*.
  - "How to Change SSL Protocols (to Disable SSL 3.0) in Oracle Fusion Middleware Products (Doc ID 1936300.1)" at `https://support.oracle.com`

  WebLogic Server 12.1 and later releases support the TLS 1.2 protocol out of the box. No additional configuration is needed.

- If WebLogic Server 10.3.6 cannot be configured to support the TLS 1.2 protocol, then you must explicitly configure WebLogic Server Proxy Plug-In 11.1.1.9 to communicate by using the TLS 1.0 protocol during the SSL handshake with WebLogic Server 10.3.6, using the WebLogicSSLVersion directive, for example:

```
<IfModule mod_weblogic.c>
```

```
  WebLogicSSLVersion TLSv1
</IfModule>
```

Note: The WebLogicSSLVersion directive is available only for the Oracle HTTP Server and Apache HTTP Server.

For more information on this directive, see "WebLogicSSLVersion" in Oracle Fusion Middleware Administrator's Guide for Oracle HTTP Server.

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 4, "Installing and Configuring the Apache HTTP Server Plug-In"
- Chapter 2, "Configuring the mod_wl_ohs Plug-In for Oracle HTTP Server"
- Chapter 5, "Installing and Configuring the Microsoft IIS Plug-In"
- Chapter 3, "Installing and Configuring the Oracle iPlanet Web Server Plug-In"

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
`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

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

Parameter Name	Default	Description	Applicable to
`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
`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 4, "Installing and Configuring the Apache HTTP Server Plug-In" ■ Chapter 2, "Configuring the mod_wl_ohs Plug-In for Oracle HTTP Server" ■ Chapter 5, "Installing and Configuring the Microsoft IIS Plug-In" ■ Chapter 3, "Installing and Configuring the Oracle iPlanet Web Server Plug-In" 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 originally 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
PathTrim	null	As per the RFC specification, generic syntax for URL is: [PROTOCOL]://[HOSTNAME]:{PORT}/{PATH}/{FILENAME};{PATH_PARAMS}/{QUERY_STRING}... PathTrim specifies the string trimmed by the plug-in from the {PATH}/{FILENAME} portion of the original URL, before the request is forwarded to WebLogic Server. For example, if 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 Note that 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
PathPrepend	null	As per the RFC specification, generic syntax for URL is: [PROTOCOL]://[HOSTNAME]:{PORT}/{PATH}/{FILENAME};{PATH_PARAMS}/{QUERY_STRING}... PathPrepend specifies the path that the plug-in prepends to the {PATH} portion of the original URL, after PathTrim is trimmed and before the request is forwarded to WebLogic Server. Note that if you need to append File Name, use DefaultFileName parameter instead of PathPrepend.	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
`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
`Debug`	OFF	Sets the type of logging performed for debugging operations. The debugging information is written to the `/tmp/wlproxy.log` file on UNIX systems and `c:\TEMP\wlproxy.log` on Windows 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 tmp or 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.	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
WLLogFile	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.	Oracle HTTP Server Oracle iPlanet Web Server Apache HTTP Server Microsoft IIS
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 in the event that 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
WLTempDir	See the Debug parameter	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 in Windows and /tmp in all Unix platforms. Also specifies the location of the _wl_proxy directory for POST data files. When both WLTempDir and WLLogFile are set, WLLogFile will override as to the location of wlproxy.log. WLTempDir will still determine the location of _wl_proxy directory.	Oracle HTTP Server Oracle iPlanet Web Server Apache HTTP Server Microsoft IIS
DebugConfigInfo	OFF	Enables the special query parameter "__ WebLogicBridgeConfig". Use it to get details about configuration parameters from the plug-in. For example, if you enable "__ WebLogicBridgeConfig" by setting DebugConfigInfo and then send a request that includes the query string ?__ WebLogicBridgeConfig, then the plug-in gathers the configuration information and run-time statistics and returns the information to the browser. The plug-in does not connect to WebLogic Server in this case. This parameter is strictly for debugging and the format of the output message can change with releases. For security purposes, keep this parameter turned OFF in production systems.	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 is unable to 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
`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
`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 dead, and fails over to the next server. The value should be set to a very large value. If the value is less than the time the servlets take to process, then you may see unexpected results. Minimum value: 10 Maximum value: 2147483647	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.	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 need to 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.	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
DefaultFileName	none	If the URI is "/" then the plug-in performs the following steps: Trims the path specified with the PathTrim parameter. Appends the value of DefaultFileName. 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.	Oracle HTTP Server Oracle iPlanet Web Server Apache HTTP Server Microsoft IIS
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

MatchExpression	none	When proxying by MIME type, set the filename pattern inside of an IfModule block using the MatchExpression parameter.
		Example when proxying by MIME type:
		`<IfModule weblogic_module>`
		`MatchExpression *.jsp`
		`WebLogicHost=myHost|paramName=value`
		`</IfModule>`
		Example when proxying by path:
		`<IfModule weblogic_module>`
		`MatchExpression /weblogic`
		`WebLogicHost=myHost|paramName=value`
		`</IfModule>`
		It is possible to define a new parameter for MatchExpression using the following syntax:
		`MatchExpression *.jsp PathPrepend=/test PathTrim=/foo`
			Oracle HTTP Server
			Apache HTTP Server

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. Note that 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. Note that 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
`WLExcludePathOrMimeType`	none	This parameter allows you make exclude certain requests from proxying. This parameter can be defined locally at the Location tag level as well as globally. When the property is defined locally, it does not override the global property but defines a union of the two parameters.	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
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 crash. 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
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 Oracle iPlanet Web Server Apache HTTP Server
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

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 dead 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
`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 WebLogic Server proxy plug-ins) ■ 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
`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
`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. If `WLLocalIP` is not set, If WLLocalIP is not set, the TCP/IP stack will choose the source IP address.	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
WLSendHdrSeparately	ON	When this parameter is set to ON, the header and body of the response are sent in separate packets. **Note:** If you need to 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
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
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 or not 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
WLServerInitiatedFailover	On	This controls whether or not 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
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

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

Parameter Name	Default	Description	Applicable to

WLSRequest	OFF	This is an alternative to the SetHandler weblogic-handler 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

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 WebLogic Server Proxy Plug-Ins. 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 WebLogic Server Proxy Plug-Ins

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
`WebLogicSSLVersion`	The best protocol supported by both the plug-in and WebLogic Server.	Specifies the SSL protocol version to use for communication between the plug-in and the WebLogic Server. This setting does not need to match that of the web server's ssl.conf file. Plug-in can have its own SSL version to communicate with WebLogic Server. **Note:** This is a new parameter for the 11.1.1.9 release and is not available for earlier releases of Web Server Plug-ins. 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/COMPONENT_TYPE/COMPONENT_NAME/default"`	Oracle HTTP Server Oracle iPlanet Web Server Apache HTTP Server Microsoft IIS