IBM WebSphere Application Server - Express for IBM i, Version 8.0

Setting up intermediary services

IBM

Note

Before using this information, be sure to read the general information under “Notices” on page 83.

Compilation date: July 12, 2011

Contents

How to send your comments

Your feedback is important in helping to provide the most accurate and highest quality information.

- To send comments on articles in the WebSphere Application Server Information Center
  1. Display the article in your Web browser and scroll to the end of the article.
  2. Click on the **Feedback** link at the bottom of the article, and a separate window containing an e-mail form appears.
  3. Fill out the e-mail form as instructed, and click on **Submit feedback** .
- To send comments on PDF books, you can e-mail your comments to: **wasdoc@us.ibm.com** or fax them to 919-254-5250.

  Be sure to include the document name and number, the WebSphere Application Server version you are using, and, if applicable, the specific page, table, or figure number on which you are commenting.

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Changes to serve you more quickly

Print sections directly from the information center navigation

PDF books are provided as a convenience format for easy printing, reading, and offline use. The information center is the official delivery format for IBM WebSphere Application Server documentation. If you use the PDF books primarily for convenient printing, it is now easier to print various parts of the information center as needed, quickly and directly from the information center navigation tree.

To print a section of the information center navigation:

1. Hover your cursor over an entry in the information center navigation until the **Open Quick Menu** icon is displayed beside the entry.
2. Right-click the icon to display a menu for printing or searching your selected section of the navigation tree.
3. If you select **Print this topic and subtopics** from the menu, the selected section is launched in a separate browser window as one HTML file. The HTML file includes each of the topics in the section, with a table of contents at the top.
4. Print the HTML file.

For performance reasons, the number of topics you can print at one time is limited. You are notified if your selection contains too many topics. If the current limit is too restrictive, use the feedback link to suggest a preferable limit. The feedback link is available at the end of most information center pages.

Under construction!

The Information Development Team for IBM WebSphere Application Server is changing its PDF book delivery strategy to respond better to user needs. The intention is to deliver the content to you in PDF format more frequently. During a temporary transition phase, you might experience broken links. During the transition phase, expect the following link behavior:

- Links to Web addresses beginning with http:// work
- Links that refer to specific page numbers within the same PDF book work
- The remaining links will *not* work. You receive an error message when you click them

Thanks for your patience, in the short term, to facilitate the transition to more frequent PDF book updates.

Chapter 1. How do I...Setting up intermediary services

Follow these shortcuts to get started quickly with popular tasks.

When you visit a task in the information center, look for the **IBM Suggests** feature at the bottom of the page. Use it to find available tutorials, demonstrations, presentations, developerWorks articles, IBM Redbooks, support documents, and more.

Use the administrative console to establish communication with local and remote web servers.

Use scripting to establish communication with local and remote web servers.

Configure transport chains.

Provide access to relational databases (JDBC resources) with scripting

Choosing a messaging provider

Provide access to messaging resources (default messaging provider) with scripting

Chapter 2. Implementing a web server plug-in

This topic describes how to implement a web server plug-in. The product works with a web server to route requests for dynamic content, such as servlets, from web applications. The web servers are necessary for directing traffic from browsers to the applications that run on an application server. The web server plug-in uses the XML configuration file to determine whether a request is for an application server.

Before you begin

- See the information about choosing a front end for your WebSphere® Application Server topology." This topic helps you determine whether to set up a web server plug-in, a proxy server, or a secure proxy server to provide session affinity, failover support, and workload balancing for your WebSphere Application Server topology. Install your web server if it is not already installed.

  gotcha: The web server that is provided with IBM® i, is already installed under product 5761-DG1 for IBM i V6R1 or 5770-DG1 for IBM i V7R1. TheIBM i web server is referred to as the IBM HTTP Server for IBM i. This web server is different from the IBM HTTP Server that is provided with WebSphere Application Server, which does not run on IBM i.

  If you want to use the IBM HTTP Server that is provided with the product, see the *Installing your application serving environment* PDF. Otherwise, see the installation information that is provided with your web server.
- Verify that your web server is configured to run the operations that are required by web applications, such as GET and POST. Typically, configuring your web server to run these operations involves setting a directive in the web server configuration file. Refer to the web server documentation for instructions.

If you are making a series of simultaneous changes, such as installing numerous applications, you might want the configuration service disabled until after you make the last change. The web server plug-in configuration service is enabled by default. To disable this service, in the administrative console click **Servers** > **Server Types** > **WebSphere application servers** > *server_name* > **administration services** > **web server plug-in configuration service**, and then clear the **Enable automated web server configuration processing** option.

gotcha: If your installation uses a firewall, make sure that you configure the web server plug-in to use a port that has been opened. See your security administrator for information about how to obtain an open port.

About this task

The appropriate plug-in file is installed. In addition, an `http` profile is created (`/QIBM/UserData/WebSphere/Plugins/V8/webserver/profiles/http`). The `http` profile can be used to facilitate the creation of web server definitions. See the topic about selecting a web server topology diagram and road map for instructions on how to configure IBM HTTP Server for IBM i to communicate with an application server.

The following procedure describes the steps for updating the plug-in configuration file, including configuring for SSL and web server tuning.

Procedure

1. Use the administrative console to change the settings in the plug-in configuration file.

   When setting up your web server plug-in, you must decide whether to have the configuration automatically generated in response to a configuration change. When the web server plug-in configuration service is enabled and any of the following conditions occur, the plug-in configuration file is automatically generated:
   - When the web server is created or saved
   - When an application is installed

- When an application is uninstalled
- When the virtual host definition is updated

You can either use the administrative console, or issue the GenPluginCfg command to regenerate your `plugin-cfg.xml` file.

Complete the following steps to regenerate your `plugin-cfg.xml` file by using the administrative console:

a. Select **Servers > Server Types > web Servers >** *web_server_name* **> plug-in properties**.

b. Select **Automatically generate plug-in configuration file,** or click one or more of the following topics to manually configure the `plugin-cfg.xml` file:

   Note: You must delete the `plugin-cfg.xml` file in the *profile_root*/`config/cells` directory before you complete this task. Otherwise, configuration changes do not persist to the `plugin-cfg.xml` file.

   - Caching
   - Request and response
   - Request routing
   - Custom Properties

   See the topic about web server plug-in configuration properties for information about how to map each property to one of these topics.

   gotcha: Do not manually update the `plugin-cfg.xml` file. Any manual updates you make for a web server are overridden whenever the `plugin-cfg.xml` file for that web server is regenerated.

c. Click **OK**.

d. Propagate the plug-in configuration. To propagate the plug-in configuration from the administrative console, click **Servers > Server Types > web Servers >** *web_server_name***Propagate plug-in**.

   Another method to propagate the plug-in configuration is to run the GenPluginCfg command. For more information, see the GenPluginCfg command documentation.

   You do not need to propagate the plug-in configuration if the web server is on the same machine as the associated stand-alone version of the product. If the propagation of the plug-in configuration fails due to an unknown cause, you must manually copy the `plugin-cfg.xml` file to the location for the remote web server installation.

   gotcha: If you use the FTP function to perform the copy, and the configuration reload fails, check the file authorities on the `plugin-cfg.xml` file and make sure that users QTMHHTTP, QNOTES and QEJBSVR have RWX authority. If the authorities are not correct, the web server cannot access the new version of the file, which causes the configuration reload to fail. To check the authorities, run the following IBM i command:

   ```
   wrklnk 'plug_in_folder_location/plugin-cfg.xml'
   ```

   Then select option 9 to view the authorities that are assigned to the users (QTMHHTTP, QNOTES, and QEJBSVR).

   If the authorities are incorrect, issue the following IBM i command to change the file authorities to the appropriate settings:

   ```
   CHGAUT USER(QEJBSVR QTMHHTTP QNOTES) OBJ('plug_in_folder_location/plugin-cfg.xml') DTAAUT(*RWX)
   ```

   The *plug_in_folder_location* is the location you specified when you transferred the `plugin-cfg.xml` file.

e. You might have to stop the application server and then start the application server for the web server to locate the `plugin-cfg.xml` file.

2. Tune your web server. See the *Tuning guide* PDF for more information.

Results

The configuration is complete. To activate the configuration, stop and restart the web server. If you encounter problems restarting your web server, check the `http_plugin.log` file for information about what portion of the `plugin-cfg.xml` file contains an error. The log file states the line number on which the error occurred, along with other details that might help you diagnose why the web server did not start. You can then use the administrative console to update the `plugin-cfg.xml` file.

If applications are infrequently installed or uninstalled, which is usually the situation in a production environment, or if you can tolerate the performance impact of generating and distributing the plug-in configuration file each time any of the previously listed actions occur, consider enabling the configuration service.

Setting up a remote web server

You can create a web server definition in the administrative console when the web server and the web server plug-in for WebSphere Application Server are on the same machine and the application server is on a different machine. This allows you to run an application server on one platform and a web server on another platform.

Before you begin

With a remote web server installation, WebSphere Application Server can facilitate plug-in administration functions and generation and propagation of the `plugin-cfg.xml` file for IBM HTTP Server for WebSphere Application Server, but not for other web servers.

About this task

You can choose a remote web server installation if you want the web server on the outside of a firewall and WebSphere Application Server on the inside of a firewall. You can create a remote web server on an unmanaged node. Unmanaged nodes are nodes without node agents. Because there is no WebSphere Application Server or node agent on the machine that the node represents, there is no way to administer a web server on that unmanaged node unless the web server is IBM HTTP Server for WebSphere Application Server. With IBM HTTP Server, there is an administration server that will facilitate administrative requests such as start and stop, view logs, and view and edit the `httpd.conf` file.

Important: The administration server is not provided with IBM HTTP Server for WebSphere Application Server which runs on z/OS® platforms. So, administration using the administrative console is not supported for IBM HTTP Server for z/OS on an unmanaged node.

The following steps will create a web server definition in the default profile. This procedure does not apply when setting up a remote web server for an i5/OS® web server. For information about setting up an i5/OS web server, see the topic entitled *Selecting a web server topology diagram and roadmap*.

Procedure

1. Install IBM Installation Manager.
2. Install your WebSphere Application Server product.
3. Install IBM HTTP Server or another supported web server.
4. Install the web server plug-ins.
5. Configure the web server plug-in.
6. Complete the setup by creating the web server definition.

   You can use the WebSphere Application Server administrative console or run the plug-in configuration script:

   - **Using the administrative console:**

a. Click **Servers > Server Types > Web servers > New** to launch the **Create new web server definition** tool. You will create the new web server definition using this tool. The values are as follows:

   1) Enter web server properties:
      - **Type**: The web server vendor type.
      - **Port**: The existing web server port. The default is 80.
      - **Installation Path**: The web server installation path. This field is required field for IBM HTTP Server only.
      - **WINDOWS Service Name**: The Windows operating system service name of the web server. The default is `IBMHTTPServer7.0`.
      - **Use secure protocol**: Use the HTTPS protocol to communicate with the web server. The default is `HTTP`.
      - **Plug-in installation location**: The directory path where the plug-in is installed.
      - **Application mapping to the web server**: Whether you want to create a mapping to existing applications that are currently deployed to the web server. Select `ALL` if you want the mapping created; select `None` if you do not want the mapping created.

        CAUTION:
        If you have enterprise applications in different security domains when you create a web server, the Key Database (KDB) files for your security configuration might not be created if you have Application mapping to the web server set to `All`. To resolve this problem, create the web server with Application mapping to the web server set to `None`. Then map the applications to the web server. All the KDB files for the web server are then created.

   2) Enter the remote web server properties. The properties for the IBM HTTP Server administration server follow:
      - **Port**: The administration server port. The default is `8008`.
      - **User ID**: The user ID that is created using the htpasswd script.
      - **Password**: The password that corresponds to the user ID created with the `htpasswd` script.
      - **Use secure protocol**: Use the HTTPS protocol to communicate with the administration server. The default is `HTTP`.

   3) Select a web server template. Select a system template or a user-defined template for the web server you want to create.

   4) Confirmation of web server creation.

- **Run the plug-in configuration script.**

7. **For AIX®, HP-UX, Linux or Solaris operating system**: On the remote web server, run the `setupadm` script. The administration server requires read and write access to configuration files and authentication files to perform web server configuration data administration. You can find the `setupadm` script in the `<`*`IHS_install_root`*`>/bin` directory. The administration server has to execute `adminctl restart` as root to perform successful restarts of IBM HTTP Server. In addition to the web server files, you must manually change the permissions to the targeted plug-in configuration files.

   The `setupadm` script prompts you for the following input:

   - User ID - The user ID that you use to log on to the administration server. The script creates this user ID.
   - Group name - The administration server accesses the configuration files and authentication files through group file permissions. The script creates the specified group through this script.
   - Directory - The directory where you can find configuration files and authentication files.
   - File name - The following file groups and file permissions change:
     - Single file name
     - File name with wildcard

  - All (default) - All of the files in the specific directory
- Processing - The `setupadm` script changes the group and file permissions of the configuration files and authentication files.

 In addition to the web server files, you must change the permissions to the targeted plug-in configuration files. See the topic on setting permissions manually for instructions.

8. **For AIX, HP-UX, Linux, Solaris, or Windows operating system**: On the remote web server, run the `htpasswd` script. The administration server is installed with authentication enabled and a blank `admin.passwd` password file . The administration server will not accept a connection without a valid user ID and password. This is done to protect the IBM HTTP Server configuration file from unauthorized access.

   Launch the **htpasswd** utility that is shipped with the administration server. This utility creates and updates the files used to store user names and password for basic authentication. Locate **htpasswd** in the `bin` directory.

   - On Windows operating systems: `htpasswd -cm <`*`install_dir`*`>\conf\admin.passwd [`*`login name`*`]`
   - On AIX, HP-UX, Linux, and Solaris platforms: `./htpasswd -cm <`*`install_dir`*`>/conf/admin.passwd [`*`login name`*`]`

   where <*install_dir*> is the IBM HTTP Server installation directory and *[login name]* is the user ID that you use to log into the administration server. The [login name] is the user ID that you entered in the user ID field for the remote web server properties in the administrative console.
9. Start IBM HTTP Server. Refer to the topic on starting and stopping the IBM HTTP Server Administration server for instructions.

What to do next

For a non-IBM HTTP Server web server on an unmanaged node, you can generate a plug-in configuration, based on WebSphere Application server repository changes. However, the following functions are not supported on an unmanaged node for a non-IBM HTTP Server web server:

- Starting and stopping the web server.
- Viewing and editing the web server configuration file.
- Viewing the web server logs.
- Propagation of the web server `plugin-cfg.xml` file.

Installing IBM HTTP Server

This topic describes how to install IBM HTTP Server.

Before you begin

The IBM HTTP Server product is included in the media package when you order any of the WebSphere Application Server for IBM i products. It is not a required product. To install the product on a non-IBM i server, use the following instructions.

To use a web server other than IBM HTTP Server, install and configure the web server before or after installing the WebSphere Application Server product but before installing the Web Server Plug-ins for IBM WebSphere Application Server.

About this task

Refer to the information center for IBM HTTP Server for detailed information on installation steps, configuring the web server for SSL, the Fast Response Cache Accelerator, or Apache directives.

What to do next

After installing the web server and the application server, install and configure the appropriate web server plug-in for a supported, installed web server. No further configuration is required for most web servers.

You can manually configure supported Web servers for WebSphere Application Server as described in "Editing web server configuration files."

The Global Security Kit (GSKit) installation image files for the WebSphere Web server plug-ins are packaged on the CD with the web server plug-in files. If you want to use GSKIT, you can download the appropriate GSKIT file to the workstation on which your web server is running.

Editing web server configuration files

Edit web server configuration files to configure a Web server.

Before you begin

Use the IBM Web Administration for IBM i to configure IBM i Web servers to communicate with a WebSphere Application Server. Doing so automatically updates the web server configuration files as needed. See the *Installing your application serving environment* PDF for more information.

If you must change a configuration for some reason, you can use the IBM Web Administration for IBM i to re-configure the web server, or you can edit the files. The recommended approach is to always use the IBM Web Administration for IBM i to configure the web server configuration file. However, sometimes you must edit the files.

You must determine whether your web server application uses a 32-bit or 64-bit architecture. If your machine supports a 64-bit architecture, you can use either a 32-bit or a 64-bit web server application. The following examples show how to determine which the architecture for your web server application:
- For Apache-based web servers, including IBM HTTP Server, you can determine the architecture using the following techniques:
- For other, non-Apache based web servers, use the following techniques. These techniques also work for Apache-based web servers.

If the web server application uses a 64-bit architecture, the plug-in library should be loaded from the `*plugin_root*/bin/64bit` directory. If the web server application uses a 32-bit architecture, the plug-in library should be loaded from the *plugin_root*`/bin/32bits` directory. If the *plugin_root*`/bin/64bits` directory does not exist, you must use the 32-bit architecture.

About this task

This task points you to information on editing web server configuration files. Select a link appropriate for your web server.

Procedure

- See the *Installing your application serving environment* PDF for information about configuring your web server.
- Configure IBM HTTP Server powered by Apache 2.x. See "Configuring IBM HTTP Server powered by Apache 2.x" on page 12.
- Configure IBM HTTP Server Version 6.x. See "Configuring IBM HTTP Server Version 6.x" on page 13.
- Configure IBM HTTP Server Version 7.0. See "Configuring IBM HTTP Server Version 7.0" on page 14.
- Configure IBM HTTP Server Version 78.0. See "Configuring IBM HTTP Server Version 8.0" on page 15.

Results

You can use the IBM Web Administration for IBM i to configure IBM i Web servers. You can also configure a web servers by editing its configuration.

Configuring Apache HTTP Server V2.2

This topic describes how to change configuration settings for Apache HTTP Server Version 2.2.

Before you begin

gotcha: If you are running IBM HTTP Server (powered by Apache) on IBM i, you can use the manual configuration steps outlined below. However, it is recommended that you use the IBM Web Administration for IBM i GUI.

Apache HTTP Server v2.2 is different from IBM HTTP Server (powered by Apache). Apache HTTP Server is not supported on IBM i.

This topic describes how to configure the Apache HTTP Server Version 2.2 Web server. Other procedures in “Editing web server configuration files” on page 8 describe configuring other supported web servers.

About this task

Perform the step that configures Apache 2.2 for your operating system.

Examples and messages are shown on more than one line for clarity. Place each directive in a web server configuration file on one line.

Local file path means a file path to the `plugin-cfg.xml` file on an application server that is on the same machine as the web server. Remote file path means the file path to the `plugin-cfg.xml` file when the application server is on a remote machine.

The name of the web server definition in the following steps is webserver1.

Procedure

Configure entries in the `httpd.conf` file. It is recommended that you use the IBM Web Administration for IBM i GUI to configure the httpd.conf file.
Use the following examples of the LoadModule and the WebSpherePluginConfig directives as models for configuring your file for IBM i:

```
LoadModule was_ap20_module
   /QSYS.LIB/product_library.LIB/QSVTAP22.SRVPGM
```

where *product_library* is the name of the WebSphere Application Server product library, such as QWAS8A or QWAS8B. See the "Directory conventions" article for information about how to determine the product library name.
Local standalone example:

```
WebSpherePluginConfig /QIBM/UserData/WebSphere/AppServer/V8/
   ND/profiles/profile1/config/cells/my_cell/nodes/
   webserver1_node/servers/webserver1/plugin-cfg.xml
```

Remote example:

```
WebSpherePluginConfig /QIBM/UserData/WebSphere/AppServer/V8/
   ND/profiles/httpprofile1/config/webserver1/plugin-cfg.xml
```

Results

The Apache 2.2 Web Server is re-configured.

What to do next

After configuring a web server, you can install applications on it. See the Applications section of the information center for more information.

Tip: To unconfigure a web server, reverse the manual steps and remove what was manually added in this procedure.

Configuring Lotus Domino

This task describes how to change configuration settings for Lotus® Domino®.

Before you begin

If you are running Domino Web Server on IBM i, use the IBM Web Administration for IBM i GUI.

Other procedures in "Editing web server configuration files" on page 8 describe configuring other supported web servers.

About this task

Use the following procedure to enable the web server plug-in to work with Lotus Domino. Ensure that you install the plug-in as root. Domino can only be installed as root, and the configuration files belong to root.

: If you install the plug-in as local and the WebSphere Application Server as nonroot, then web server management on WebSphere Application Server, such as generation, propagation, deletion of web server definitions and more, is unavailable because the `plugin-cfg.xml` file must be installed as root.

Procedure

1. Start the Domino server.
2. Access the `names.nsf` file using your web browser (for example, `http://tarheels2.raleigh.ibm.com/names.nsf`). The browser prompts you for a password.
3. Give the administrator name and password.
4. Click **Configuration** on the left side of the page.
5. Click **Servers** on the left side of the page.
6. Click **All Server Documents** on the left side of the page.
7. Click the server that you intend to have work with the product
8. Click **Edit Server** on the top-left of the center window.
9. Click **Internet Protocols** in the middle of the page.
10. Add the path to the Domino plug-in under the DSAPI Section in the middle-right of the page.

    The DSAPI filter location on IBM i is `/QSYS.LIB/QWAS8.LIB/LIBDOMINOH.SRVPGM`. The plug-in is installed in the `bin` directory of the Web Server Plug-ins.

    If specifications for filter files for the Domino Server Application Programming Interface (DSAPI) exist, use a space to delimit the web server plug-in for WebSphere Application Server.
11. Click **Save and Close** in the upper left of the center window.
12. Define the location of the `plugin-cfg.xml` configuration file.

    The location varies depending on how you have configured your system. If the web server and the application server are on separate machines, you have a remote installation.

If the two servers are on the same machine, you have a local installation.

In the following examples, webserver1 is the web server definition name.

Setting the path to the plug-in configuration file

Edit your notes.ini file.

a. From a CL command prompt, enter the Work with Domino Servers (WRKDOMSVR) command.
b. Type **13** next to the applicable Domino server, then press **Enter**.
c. Add or edit the WebSphereInit property. (See the following table.)
   To add a new line, type "**i**" next to the wanted insertion line, then press **Enter**.
d. Press **F3** to save and exit.

Table 1. Editing the notes.ini file. Use this table when editing your notes.ini file.

If the type of installation is:	Then use this command to set the environment variable:
Remote	`WebSpherePluginConfig` *profile_root*`/config/webserver1/plugin-cfg.xml`
Local standalone	`WebSpherePluginConfig` *profile_root*`/config/cells/my_cell/nodes/webserver1_node/servers/webserver1/plugin-cfg.xml`

The `setupPluginCfg.sh` file is created in two places:

- The *plugins_root*/`bin` directory
- The *lotus_root*/`notesdata` directory

You can run the script from either location to set the WAS_PLUGIN_CONFIG_FILE environment variable. However, if you are re-configuring the web server, you might want to set the path yourself by setting the value of the environment variable with a path from the preceding table.

13. If you are configuring a 64-bit version of the Domino Version 7 or higher web server, modify the notes.ini file to point to the *plugin_root*`\bin\64bit/domino5.dll` file.
14. Restart the Domino server. When the server starts, information like the following example is displayed:

```
01/21/2005 01:21:51 PM  JVM: Java virtual machine initialized
WebSphere Application Server DSAPI filter loaded
01/21/2005 01:21:52 PM  HTTP Web Server started
```

Results

This procedure results in reconfiguring Version 6.x of Lotus Domino.

What to do next

After configuring a web server, you can install applications on it. See the Applications section of the information center for more information.

For more information about configuring Lotus Domino to work with WebSphere Application Server, search the Lotus Support Services website at http://www-3.ibm.com/software/lotus/support/. Enter the search term `WebSphere` in the keyword search field.

Tip: To unconfigure a web server, reverse the manual steps and remove what was manually added in this procedure.

Lotus Domino file locations and troubleshooting tips

Lotus Domino Server is one of the Web servers that WebSphere Application Server supports. This topic describes configuration file locations and provides other tips related to using Lotus Domino.

Lotus Domino Server file locations

Lotus Domino server file locations are:

- `/QIBM/ProdData/LOTUS/NOTES/Notes.jar` *data-directory*`/names.nsf` where *data-directory* is the data directory of your Domino server.

  Perform the following steps to determine the directory name:

  1. From a control language (CL) command prompt, enter the Work with Domino Server (WRKDOMSVR) command.
  2. Type **13** next to the applicable server, then press **Enter**.
  3. The value of the property named Directory is the name of the data directory of your Domino Server.

The Domino Server plug-in might fail to load on an iSeries® system

Error messages in the Domino console can indicate more detailed error messages in the Notes® spool files. For example, if you see the following error message on your Domino console, perform the following steps:

Sample error message:

```
01/05/2006 13:28:56 HTTP Server:
Failed to load DSAPI module
/QSYS.LIB/QWAS8.LIB/LIBDOMINOH.SRVPGM
```

1. From a CL command prompt, enter the Work with Spool Files (WRKSPLF) command:

```
WRKSPLF QNOTES
```

2. Use option **5** to open the applicable spool file for your notes server.

   You might have to stop your Domino server to view the spool file.

3. Inspect the output to find applicable error messages. For example, suppose that the following entry is in the log:

```
ERROR: lib_sxp: sxpCreate: Can't open
  '/QIBM/UserData/webas5/base/rb5025test/config
  /cells/plugin-cfg.xml', OS Err: 3401
ERROR: ws_config_parser: configParserParse:
  Failed to create the sxpParser object for
  profile_root/config/plugin-cfg.xml
ERROR: ws_common: websphereUpdateConfig:
  Failed parsing plugin config file:
ERROR: profile_root/config/plugin-cfg.xml
ERROR: The above file may not have correct authorities.
ERROR: Do WRKLNK, Opt 9 on above file.  ERROR: Check that QTMHHTTP and QNOTES have correct authorities:
ERROR: QTMHHTTP *RX  ERROR: QNOTES *RX  ERROR: ws_common: websphereInit:
  Failed to load the config file
ERROR: domino5_plugin: FilterInit:
  Failed to initialize WebSphere
```

Configuring IBM HTTP Server powered by Apache 2.x

This topic describes how to change configuration settings for IBM HTTP Server powered by Apache 2.x.

Before you begin

If you are running IBM HTTP Server (powered by Apache) on IBM i, you can use the manual configuration steps outlined below with the IBM Web Administration for IBM i.

This topic describes how to configure the IBM HTTP Server. Other procedures in "Editing web server configuration files" on page 8 describe configuring other supported web servers.

About this task

Perform the step that configures IBM HTTP Server version 2.2 for your operating system. IBM HTTP Server powered by Apache 2.2 is supported on IBM i Versions 6.1 and 7.1.

Examples and messages are shown on more than one line for clarity. Place each directive in a web server configuration file on one line.

Local file path means a file path to the `plugin-cfg.xml` file on an spplication server that is on the same machine as the web server. Remote file path means the file path to the `plugin-cfg.xml` file when the spplication server is on a remote machine.

The web server definition name in the following examples is webserver1.

Procedure

Configure entries in the `httpd.conf` file.
Use the IBM Web Administration for IBM i GUI to configure the `httpd.conf` file. See the topic Configuring an HTTP server instance on IBM i for a description of how to use this GUI.
Use the following examples of the LoadModule and the WebSpherePluginConfig directives as models for configuring your file for IBM i:

```
LoadModule was_ap20_module
   /QSYS.LIB/product_library.LIB/QSVTAP22.SRVPGM
```

where *product_library* is the name of the WebSphere Application Server product library, such as QWAS8A, and QWAS8B. See the topic Directory conventions for information about how to determine the product library name.
Local standalone example:

```
WebSpherePluginConfig /QIBM/UserData/WebSphere/AppServer/V8/
   ND/profiles/profile1/config/cells/my_cell/nodes/
   webserver1_node/servers/webserver1/plugin-cfg.xml
```

Remote example:

```
WebSpherePluginConfig
   /QIBM/UserData/WebSphere/AppServer/V8/
   ND/profiles/httpprofile1/config/webserver1/plugin-cfg.xml
```

Results

This procedure results in editing and re-configuring IBM HTTP Server powered by Apache 2.x.

What to do next

If the IBM HTTP Server 1.3.2x directive, LoadModule ibm_app_server_http_module, is present in an IBM HTTP Server 2.x `httpd.conf` file, the IBM HTTP Server cannot start. You must comment or delete the directive to start the Version 2.x server.

After configuring a web server, you can install applications on it. See the Applications section of the information center for more information.

Tip: To unconfigure a web server, reverse the manual steps and remove what was manually added in this procedure.

Configuring IBM HTTP Server Version 6.x

This topic describes how to change configuration settings for IBM HTTP Server.

Before you begin

IBM HTTP Server Version 6.x is not supported on IBM i. See the *Installing your application serving environment* PDF for information on how to configure IBM HTTP Server powered by Apache 2.0.

This topic describes how to configure IBM HTTP Server, Version 6.x. Other procedures in "Editing web server configuration files" on page 8 describe configuring other supported web servers.

About this task

Perform the step that configures IBM HTTP Server for your operating system.

Examples and messages are shown on more than one line for clarity. Place each directive in a web server configuration file on one line.

Local file path means a file path to the `plugin-cfg.xml` file on an spplication server that is on the same machine as the web server. Remote file path means the file path to the `plugin-cfg.xml` file when the spplication server is on a remote machine.

The *node_name* in the following spplication server local file paths is *web_server_name*`_node` for a standalone spplication server

Procedure

Results

This procedure results in editing and re-configuring IBM HTTP Server.

What to do next

If the IBM HTTP Server V1.3.x directive, LoadModule ibm_app_server_http_module, is present in an IBM HTTP Server Version 6.x and later `httpd.conf` file, IBM HTTP Server cannot start. You must comment or delete the directive to start the Version 6.x server.

The mod_was_ap20_http plug-in module requires the GSKIT Secure Sockets Layer (SSL) encryption library if the plug-in is configured to support encrypted connections to back-end WebSphere Application Servers.

Installing the web server plug-ins installs the GSKIT SSL encryption library at the required level if it is not installed. If you manually copy the plug-in to a new machine, you might not have the required GSKIT libraries for encrypting back-end connections.

After configuring a web server, you can install applications on it. See the Applications section of the information center for more information.

Tip: To unconfigure a web server, reverse the manual steps and remove what was manually added in this procedure.

Configuring IBM HTTP Server Version 7.0

This topic describes how to change configuration settings for IBM HTTP Server.

Before you begin

IBM HTTP Server Version 7.0 is not supported on IBM i. See the *Installing your application serving environment* PDF for information on how to configure IBM HTTP Server powered by Apache 2.0, or IBM HTTP Server powered by Apache 2.0.

This topic describes how to configure IBM HTTP Server, Version 7.0. Other procedures in "Editing web server configuration files" on page 8 describe configuring other supported web servers.

About this task

Perform the step that configures IBM HTTP Server for your operating system.

Examples and messages are shown on more than one line for clarity. Place each directive in a web server configuration file on one line.

Local file path means a file path to the `plugin-cfg.xml` file on an application server that is on the same machine as the web server. Remote file path means the file path to the `plugin-cfg.xml` file when the application server is on a remote machine.

The *node_name* in the following Application Server local file paths is *web_server_name*`_node` for a standalone application server.

Procedure

Results

This procedure results in editing and re-configuring IBM HTTP Server.

What to do next

The mod_was_ap22_http plug-in module requires the GSKIT Secure Sockets Layer (SSL) encryption library if the plug-in is configured to support encrypted connections to back-end WebSphere Application Servers.

Installing the web server plug-ins installs the GSKIT SSL encryption library at the required level if it is not installed. If you manually copy the plug-in to a new machine, you might not have the required GSKIT libraries for encrypting backend connections.

After configuring a web server, you can install applications on it. See the Applications section of the information center for more information.

Tip: To unconfigure a web server, reverse the manual steps and remove what was manually added in this procedure.

Configuring IBM HTTP Server Version 8.0

This topic describes how to change configuration settings for IBM HTTP Server.

Before you begin

IBM HTTP Server Version 8.0 is not supported on IBM i. See the *Installing your application serving environment* PDF for information on how to configure IBM HTTP Server powered by Apache 2.0, or IBM HTTP Server powered by Apache 2.0.

This topic describes how to configure IBM HTTP Server, Version 8.0. Other procedures in “Editing web server configuration files” on page 8 describe configuring other supported web servers.

About this task

Perform the step that configures IBM HTTP Server for your operating system.

Examples and messages are shown on more than one line for clarity. Place each directive in a web server configuration file on one line.

Local file path means a file path to the `plugin-cfg.xml` file on an application server that is on the same machine as the web server. Remote file path means the file path to the `plugin-cfg.xml` file when the application server is on a remote machine.

The *node_name* in the following application server local file paths is *web_server_name*`_node` for a standalone application server.

Procedure

Results

This procedure results in editing and re-configuring IBM HTTP Server.

What to do next

The mod_was_ap22_http plug-in module requires the GSKIT Secure Sockets Layer (SSL) encryption library if the plug-in is configured to support encrypted connections to back-end WebSphere Application Servers.

Installing the web server plug-ins installs the GSKIT SSL encryption library at the required level if it is not installed. If you manually copy the plug-in to a new machine, you might not have the required GSKIT libraries for encrypting backend connections.

After configuring a web server, you can install applications on it. See the Applications section of the information center for more information.

Tip: To unconfigure a web server, reverse the manual steps and remove what was manually added in this procedure.

Creating web server templates

A web server template is used to define the configuration settings for a new web server. When you create a new web server, you can either create a new web server definition or use a previously created web server template that is based on another, already existing web server.

About this task

To create a web server template.

Procedure

1. In the administrative console, click **Servers > Server Types > Web servers >** , and then click **Templates**.

   You can also use the **createWebServerTemplate** command for the AdminTask object to create a web server template.
2. On the Server Templates page, click **New**.
3. Select the web server that you want to use to create the new template, and then click **OK**.
4. Enter the name of the new template and, optionally, a description of that template that distinguishes it from your other templates.
5. Click **OK**.

Results

Your new template displays in the list of server templates that you can use to create a new web server.

What to do next

You can perform one of the following actions to display a list of all of the server templates that are available on your system:

- In the administrative console, click **Servers > Server Types > Web servers**, and then click **Templates**.
- Issue a **listServerTemplates** wsadmin command that includes the `-serverType WEB_SERVER` parameter.

Allowing web servers to access the administrative console

This topic describes how to add the virtual host that servers the administrative console to the plug-in configuration file so that you can access the administrative console through a web server.

Before you begin

Install your WebSphere Application Server product, a web server, the Web Server Plug-ins, and the WebSphere Customization Toolbox.

When you configure a web server plug-in, a Web server definition is created on the application server system, either directly when they are on the same machine or by a script for remote scenarios.

After creating the web server definition, the plug-in configuration file exists within the web server definition.

About this task

This task gives you the option of configuring the admin_host so that web servers can access the administrative console. When the web server plug-in configuration file is generated, it does not include admin_host on the list of virtual hosts.

Procedure

1. Use the administrative console to change the admin_host virtual host group to include the web server port (80 by default).
   a. Click **Environment > Virtual Host > admin_host > Host Aliases > New**.

      The default port that displays is 80, unless you specify a different port during installation or profile creation.

   b. Specify the IP address, or the name of the machine that is hosting the HTTP server.

      For example, if you installed a WebSphere Application Server product on a machine that is named waslwaj.rtp.ibm.com, specify the name in this field.
2. Click **Apply > Save**.
3. Stop and restart the application server.

   For example, to access the administrative console of a stand-alone application server, stop and restart the server1 process.

   Start a Qshell session and run the following command:

   ```
   cd profile_root/bin
   stopServer server1
   ```

   Then issue the following command to stop the application server:

   ```
   stopServer -profileName myProfile server1
   ```

   After receiving the following message, you can restart the application server:

   ```
   Server server1 stop completed.
   ```

   To start the application server, issue the following command:

   ```
   startServer server1
   ```

When the application server is running, a message is displayed that indicates that the process is running. This message includes the iSeries job ID and the administrative console port.

4. Edit the `plugin-cfg.xml` file to include the following entries:

```
<VirtualHostGroup Name="admin_host">
        <VirtualHost Name="*:13060"/>
   </VirtualHostGroup>
    ...
    ...
    ...
    <ServerCluster Name="my60Profile.dmgr_muiSeries_Cluster">
        <Server LoadBalanceWeight="1" Name="myiSeries_my60Profile.dmgr">
            <Transport Hostname="myiSeries" Port="11060" Protocol="http"/>
        </Server>

        <PrimaryServers>
            <Server Name="myiSeries_my60Profile.dmgr"/>
        </PrimaryServers>
    </ServerCluster>
    ...
    ...
    ...
    <UriGroup Name="admin_host_my60Profile.dmgr_myiSeries_Cluster_URIs">
        <Uri AffinityCookie="JSESSIONID"
            AffinityURLIdentifier="jsessionid" Name="/ibm/console/*"/>
    </UriGroup>
    <Route ServerCluster="my60Profile.dmgr_myiSeries_Cluster"
        UriGroup="admin_host_my60Profile.dmgr_myiSeries_Cluster_URIs" VirtualHostGroup="admin_host"/>
```

   If your HTTP server has an HTTP port other than 80, add an entry to the VirtualHostGroup:

```
<VirtualHost Name="*:port"/>
```

   The *port* variable is your HTTP server port.

Results

You can configure your supported web servers to access the administrative console application of a stand-alone application server.

Administering web servers from the administrative console

Web server definition

To administer or manage a web server using the administrative console, you must create a web server definition or object in the WebSphere Application Server repository.

The creation of this object is exclusive of the actual installation of a web server. The web server object in the WebSphere Application Server repository represents the web server for administering and managing the web server from the administrative console.

The web server object contains the following web server properties:

- installation root
- port
- configuration file paths
- log file paths

In addition to web server properties, the web server contains a plug-in object. The plug-in object contains properties that define the `plugin-cfg.xml` file.

The definitions of the web server object are made using the **wsadmin** command or the administrative console. You can also define a web server object in the WebSphere Application Server repository using the profile create script during installation, a `.jacl` script, and by using the administrative console wizard.

There are three types of WebSphere Application Server nodes upon which you can create a web server. The type depends on the version of WebSphere Application Server, as follows:

- **Managed node**. A node that contains a node agent. This node can exist only in a deployment manager environment. The importance of defining a web server on a managed node is that the administration and configuration of the web server is handled through the node agent from the administrative console. Support for administration and configuration through the administrative console is limited to IBM HTTP Server only. Non-IBM HTTP Server web servers must be on a managed node to handle plug-in administrative functions and the generation and propagation of the `plugin-cfg.xml` file.
- **Stand-alone node**. A node that does not contain a node agent. This node usually exists in WebSphere Application Server (base) or WebSphere Application Server, Express environment. A stand-alone node can become a managed node in a deployment manager environment after the node is federated . A stand-alone node does not contain a node agent, so to administer and manage IBM HTTP Server, there must be an IBM HTTP Server administration server installed and running on the stand-alone machine that the node represents. IBM HTTP Server ships with the IBM HTTP Server administration server and is installed by default. Support for administration and configuration through the administrative console is limited to IBM HTTP Server only.
- **Unmanaged node**. A node that is not associated with a WebSphere Application Server node agent. This node cannot be federated. Typically, the unmanaged node represents a remote machine that does not have WebSphere Application Server installed. However, you can define an unmanaged node on a machine where WebSphere Application Server is installed. This node can exist in aWebSphere Application Server (base), WebSphere Application Server, Express, or deployment manager environment. An unmanaged node does not contain a node agent, so to administer and manage IBM HTTP Server, an IBM HTTP Server administration server must be installed and running on the stand-alone machine that the node represents. Support for administration and configuration through the administrative console is limited to IBM HTTP Server only.

Web servers, which are not IBM HTTP Servers for WebSphere Application Server, are not fully administered from the WebSphere Application Server administrative console. The administration functions for Web servers, which are not IBM HTTP Servers for WebSphere Application Server, are:

- On managed nodes:
  - Web server status in the web server collection panel or `serverStatus.sh`
  - Generation of the `plugin-cfg.xml`
  - Propagation of the `plugin-cfg.xml`
- On unmanaged nodes:
  - Web server status in the web server collection panel or `serverStatus.sh`
  - Generation of the `plugin-cfg.xml`

Web server configuration

Plug-in configuration involves configuring the web server to use the binary plug-in module that WebSphere Application Server provides. Plug-in configuration also includes updating the plug-in XML configuration file to reflect the current application server configuration. The binary module uses the XML file to help route web client requests.

The plug-ins configuration process uses the following files to configure a plug-in for the web server that you select:

- The **web server configuration file** on the web server machine, such as the `httpd.conf` file for IBM HTTP Server.
- The **binary web server plug-in file** on the web server machine.

- The **plug-in configuration file, `plugin-cfg.xml`**, on the application server machine that you propagate (copy) to a Web server machine.
- The **configuration script** for configuring the web server definition for your application server in a remote HTTP scenario.

See the following descriptions of each file.

Web server configuration file

The web server configuration file is installed as part of the web server.

Configuration consists of adding directives that identify file locations of two files:
- Binary web server plug-in file
- Plug-in configuration file, `plugin-cfg.xml`

Binary web server plug-in file

An example of a binary plug-in module is the `mod_was_ap22_http.dll` file for IBM HTTP Server on the Windows platform.

Another example of a binary plug-in module is the QSVTAP20 service program on the IBM i platform.

The binary plug-in file does not change. However, the configuration file for the binary plug-in is an XML file. The application server changes the configuration file when certain changes to your WebSphere Application Server configuration occur.

The binary module reads the XML file to adjust settings and to route requests to the application server.

Plug-in configuration file, `plugin-cfg.xml`

The plug-in configuration file is an XML file with settings that you can tune in the administrative console. The file lists all of the applications installed on the web server definition. The binary module reads the XML file to adjust settings and to route requests to the application server.

When you make application server configuration changes that affect deployed applications, regenerate the plug-in configuration XML file.

After regeneration, propagate (copy) the file to the web server machine. The binary plug-in then has access to the most current copy of its configuration file.

On IBM i systems, the plug-in is not automatically generated. You must regenerate and propagate the file manually.

Configuration script for the Web server definition

Configuring your web server with the `configureOs400WebserverDefinition` script or using the IBM i Administrative GUI creates the `configure`*`web_server_name`* script on the web server machine in the *plugins_root*`/bin` directory. The script is created for remote installation scenarios only.

Copy the script from the web server machine to the *app_server_root*`/bin` directory in the IBM i partition. Run the script to create a web server definition in the configuration of the application server.

The IBM i Administrative GUI has plug-ins that allow the administrative console to manage IBM HTTP Servers. Use the administrative console to update your web server definition with remote web server management options. Click **Servers > Server Types > Web servers > *web_server_name*** to see configuration options. For example, click **Remote Web server management** to change such properties as:

- Host name
- Administrative port
- User ID
- Password

If a web server definition already exists for a standalone application server, running the script does not add a new web server definition. Each standalone application server can have only one Web server definition.

You cannot use the administrative console of a standalone application server to add or delete a web server definition. However, you can do both tasks using the administrative scripting interface:

- Add a web server definition through the wsadmin facility using the `configure`*`web_server_name`* script. The script uses a Java Command Language (Jacl) script named `configureWebserverDefintion.jacl` to create and configure the web server definition.
- Delete a web server definition using wsadmin commands. The Web server is named webserver1 in the following example:

```
set webserverName webserver1
set webserverNodeSuffix _node
set webserverNodeName
$webserverName$webserverNodeSuffix
$AdminConfig remove
  [$AdminConfig getid
    /Node:$webserverNodeName/Server:$webserverName]
$AdminConfig remove
  [$AdminConfig getid /Node:$webserverNodeName]
$AdminConfig save
```

Alternatively, you can use the `configureOs400WebServerDefinition` and `removeOs400WebServerDefinition` scripts to perform these tasks.

Replacing the default plug-in configuration file with the file from the web server definition (propagation)

The default file uses fixed parameter values that might not match the parameter values in the actual file on the application server. The default file is a placeholder only.

The file cannot reflect changes that occur in the application server configuration. The file also cannot reflect nondefault values that might be in effect on the application server.

Web server collection

Use this page to configure, manage, and view information about your web servers.

Web servers

To view this administrative console page click **Servers > Server Types > Web Servers**.

To create a new web server, click **New** to launch the Create new web server entry wizard. To manage an installed web server, select the check box beside the application name in the list and click a button:

Button	Resulting Action
Generate Plug-in	When the plug-in configuration service is enabled, a plug-in configuration file is automatically generated for a web server whenever: • The WebSphere Application Server administrator defines new web server. • An application is deployed to an Application Server. • An application is uninstalled. • A virtual host definition is updated and saved.
Propagate Plug-in	Choosing this action will copy the `plugin-cfg.xml` file from the local directory where the Application Server is installed to the remote machine. If you are using IBM HTTP Server V6 or higher for your web server, WebSphere Application Server can automatically propagate the plug-in configuration file to remote machines provided there is a working HTTP transport mechanism to propagate the file.
New	Launches the wizard to create a new web server entry.
Delete	Deletes one or more of the selected web server entries.
Templates ...	Opens the web server templates list panel. From this panel you can create a new template or delete existing templates.
Start	Starts one or more of the selected web servers.
Stop	Stops one or more of the selected web servers.
Terminate	Terminates one or more of the selected web servers.

Name

Specifies a logical name for the web server. This can be the host name of the machine, or any name you choose.

Web server type

Indicates the type of web server you are using.

Node

Specifies a logical name for the web server. This can be the host name of the machine, or any name you choose.

Specifies the name of the node on which the web server is defined. This column only applies for the WebSphere Application Server, Network Deployment product.

Version

Specifies the version of the WebSphere Application Server on which the web server is defined.

Status

Indicates whether the web server is started, stopped, or unavailable.

If the status is unavailable, the IBM HTTP Server administration server is not running, and you must start the application server before you can start the web server.

Table 2. Status indicators. The following table describes the status indicators.

➡	**Started**	The web server is running.
✖	**Stopped**	The web server is not running.

Table 2. Status indicators (continued). The following table describes the status indicators.

?	**Unknown**	Status cannot be determined. A web server with an unknown status might, in fact, be running but has an unknown status because the application server that is running the administrative console cannot communicate with this web server.

Web server configuration

Use this page to configure web server properties.

Web servers:

To view this administrative console page click **Servers > Server Types > Web Servers >** *web_server_name*.

Web server name
Specifies a logical name for the web server.

Type
Specifies the vendor of the web server. The default value is IBM HTTP Server.

The options for the type of web servers are:
- IHS
- APACHE
- IIS
- SUNJAVASYSTEM
- DOMINO

Port
The port from which to ping the status of the web server. This field is required.

You can use the WebSphere Application Server administrative console to check if the web server is started by sending a ping to attempt to connect to the web server port that is defined. In most cases the port is 80. If you have a firewall between the web servers and application servers, you will not use port 80 on the firewall between the two systems. In most cases, your port will be different, such as 9080 or 9443. You should set the alternate ports for the web server using the WebSphere Application Server administrative console, then set that port to the web server to listen on, in addition to the typical port 80 and 443.

Installation path
Enter the fully qualified path where the web server is installed. This field is required if you are using IBM HTTP Server. For all other web servers, this field is not required. If you enable any administrative function for non-IBM HTTP Server Web servers, the installation path will be necessary.

Configuration file name
There are two ways to view or modify the contents of the configuration file:
1. Click **Edit** to view the configuration file. You will be able to make modifications from this view. This is valid for IBM HTTP Server only.
2. Click **Configuration file** under Additional properties. You will be able to make modifications from this view. This is valid for IBM HTTP Server only.

Service name - Microsoft Windows operating systems only	Specifies the Microsoft Windows operating system name for the Web server. The name is the service name and you can find it by opening the **General** properties tab of the web server service name.

Checking your IBM HTTP Server version

At times, you might need to determine the version of your IBM HTTP Server installation.

About this task

The following information describes how you can determine the IBM HTTP Server version and provides examples. The server versions that are provided in the output below are not necessarily the versions that are distributed with WebSphere Application Server.

Procedure

1. Change the directory to the installation root of the Web server.
2. Find the subdirectory that contains the executable. The executable for IBM HTTP Server is:
   - APACHEDFT
3. Issue the command with the -v option to display the version information.

   Run the STRTCPSVR command with the -v option to display the version information.

   ```
   STRTCPSVR SERVER(*HTTP) HTTPSVR(APACHEDFT '-V')
   ```

Results

This web server is not a part of WebSphere Application Server, but is distributed with IBM i. The version depends on the i/5OS release. The version is shown in the "Server version" field and will look something like the following:

```
i 6.1:
Server version: Apache/2.2.11 (i5)
Server built:   Jul 21 2009 19:24:31

i7.1:
Server version: Apache/2.2.11 (i5)
Server built:   Oct  7 2009 21:04:27
```

Web server log file

Use this page to view the log file for your web server.

To view this administrative console page, click **Servers > Server Types > Web Servers >** *web_server_name* **> Log file**.

Web server log file configuration:

Access log file name	Any request that is made to the web server displays in this file.
Error log file name	Any error that occurs in the web server displays in this file.

Web server log file runtime:

Access log file name	Click **View** to display the contents of this file.
Error log file name	Click **View** to display the contents of this file.

Web server custom properties

Use this page to view and manage arbitrary name-value pairs of data, where the name is a property key and the value is a string value that can be used to set internal system configuration properties.

The administrative console contains several Custom Properties pages that work similarly. To view one of these administrative pages, click a Custom Properties link.

Web servers:

To view this administrative console page click **Servers > Server Types > Web Servers >** *web_server_name* **> Custom properties**.

Name	Specifies the name (or key) for the property.
Value	Specifies the value paired with the specified name.
Description	Provides information about the name-value pair.

Compensation service custom properties

You can specify additional settings for the compensation service through setting a custom property.

Complete the following steps to set a custom property for the compensation service.

1. Start the administrative console.
2. In the navigation pane, click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name*** > **[Container Settings] Container Services** > **Compensation Service** > **[Additional Properties] Custom Properties**.
3. Click **New**.
4. On the settings page, enter the property that you want to configure in the **Name** field and the value that you want to set it to in the **Value** field.
5. Click **Apply** or **OK**.
6. Click **Save** to save your changes to the master configuration.
7. Restart the server.

You can use the custom properties page to define the following compensation service custom property:

- "Suppressing the compensation service"

Suppressing the compensation service: Not all web servers are configured to handle SOAP messages containing CoordinationContext elements. You can use WebSphere Application Server to configure a custom property for the compensation service which processes a predefined list of Enterprise Java Beans for which no CoordinationContext should be sent on web service requests.

When the compensation service is used, CoordinationContext elements are included in the outgoing SOAP header. For example:

```
<wscoor:CoordinationContext soapenv:mustUnderstand="1"
...
</wscoor:CoordinationContext>
```

If such a SOAP message is received by a web server which is not configured to process CoordinationContext elements, an exception message is produced. See the following example:

```
Header block local name 'CoordinationContext' is not defined.
```

You can construct a file containing a list of all Enterprise Java Beans which should not send the CoordinationContext element in web service requests. This file must be in plain text format and must contain one entry per line, in the following format:

```
application_name#module#bean
application_name#module#bean
application_name#module#bean
```

Here `application_name` is the name of the application as known on the server; `module` is the name of the Enterprise Java Bean jar; and `bean` is the name of the Enterprise Java Bean.

Note: This file must only contain entries for beans not configured to use the compensation service. This custom property will not be effective for any beans listed in the file which have compensation service metadata associated with them.

Name	**Value**
SUPPRESS_CSCOPE_ON_WS_CALLS	The fully qualified file name

Remote web server management

Use this page to configure a remote IBM HTTP Server web server.

To view the administrative console page for Remote web server management, click **Servers > Server Types > Web Servers >** *web_server_name* **> Remote web server management**.

Note: For a base WebSphere Application Server or WebSphere Application Server, Expressconfiguration, click **Servers > Server Types > Web Servers > New** to create a web server.

Web servers:

Port	Indicates the port to access the administration server (default is 8008). The HTTP administration server port is 2001 on IBM i.
Use SSL	Specifies if the port is secure.
User ID	Specifies a user ID in the <*install_dir*>`/conf/admin.passwd` file. Create this with the `htpasswd` script file, located in the <*install_dir*>`/bin` directory.
Password	Specifies a password in the <*install_dir*>`/conf/admin.passwd` file. Create this with the `htpasswd` script file, located in the <*install_dir*>`/bin` directory. On IBM i, there is no htpasswd script in the <*install_dir*>`/bin` directory. The HTTP administrator is typically created as an IBM i SECOFR profile.

Web server configuration file

Use this page to view or modify the contents of the Web server configuration file in your web browser. To view this administrative console page click **Servers > Server Types > Web Servers >** *web_server_name*. Under **Additional Properties**, click **Configuration File**.

Web servers:

If you have made changes to the configuration file you will need to restart your web server, in order for the changes to take effect.

Global directives

Use this page to configure the global directives for your web server.

To view this administrative console page, click **Servers > Server Types > Web Servers >** *web_server_name* **> Global directives**.

Security enabled:

Specifies if security is enabled in your web server.

Key store certificate alias:

If the **Security enabled** box is checked, specify the key store certificate alias.

Server name:

Specifies the hostname that the web server uses to identify itself.

Listen ports:

Specifies the port on which your web server will listen for requests.

Document root:

The directory where the web server will serve files.

Key store name:

Specifies the name you have assigned to your keystore. A button is also provided to **Manage keys and certificates**.

Target key store directory and file name:

Specifies the target directory and file name of your keystore on the machine where the web server is installed. A button is also provided to **Copy to web server key store directory**.

SSL Version 2 timeout:

Specifies the SSL Version 2 timeout.

SSL Version 3 timeout:

Specifies the SSL Version 3 timeout.

Web server virtual hosts collection

Use this page create or edit virtual hosts for your Web server.

To view this administrative console page, click **Servers > Server Types > Web Servers >** *web_server_name* **> Configuration settings > Virtual hosts**.

To create a new virtual host, click the **New** button to launch the virtual host configuration panel. To delete an existing virtual host, select the check box beside the virtual host in the list and click **Delete** .

IP address:Port:

The IP address and port number of your virtual host for the specified web server.

Server name:

Specifies the name of your virtual host for the specified web server.

Security enabled:

Specifies whether or not security is enabled for your virtual host for the specified web server. The values are **true** or **false**.

Web server virtual hosts detail

Use this page create or edit virtual hosts for your Web server.

To view this administrative console page, click **Servers** > **Server Types** > **Web Servers** > *web_server_name* > **Configuration settings** > **Virtual hosts** > **New**.

Security enabled:

Specifies whether or not security is enabled for your virtual host for the specified web server. Check the box to enable security

IP address:

The IP address of your virtual host for the specified Web server.

Port:

The port number of your virtual host for the specified web server.

Server name:

Specifies the name of your virtual host for the specified web server.

Document root:

Specifies the location of the `htdocs` directory for your web server.

Keystore filename:

Specifies the name you have assigned to your keystore.

Keystore directory:

Specifies the target directory for the key store file on the machine where your web server is installed.

Keystore certificate label:

Specifies the keystore certificate label. The certificate label specified here is the certificate that used in secure communication for this virtual host.

Editing the web server type

This topic provides information on how to change the type of Web server.

About this task

If you install a web server that is different from the one that is currently installed, you can modify the web server type from IBM HTTP Server to a non-IBM HTTP Server and vice versa, rather than delete the web server and create a new web server definition. If you change the web server type from IBM HTTP Server to non-IBM HTTP Server web server, the administration capabilities are lost accordingly.

Procedure

1. From the WebSphere Application Server administrative console, click **Servers** > **Server Types** > **Web servers**.
2. Select the server that you want to modify.

3. On the web server configuration panel, change your web server by selecting an option from the Type drop-down menu. If you are changing from a non-IBM HTTP Server to an IBM HTTP Server, you are also prompted for information such as IBM HTTP Server administration server port, user ID, and IBM HTTP Server administration server password.
4. Click **Apply**.

Results

You can verify your changes on the web servers collection panel. The web server type displays in the Web Server Type column.

Web server plug-ins

Web server plug-ins enable the web server to communicate requests for dynamic content, such as servlets, to the application server. A web server plug-in is associated with each web server definition. The configuration file (plugin-cfg.xml) that is generated for each plug-in is based on the applications that are routed through the associated web server.

A web server plug-in is used to forward HTTP requests from a supported web server to an application server. Using a web server plug-in to provide communication between a web server and an application server has the following advantages:
- XML-based configuration file
- Standard protocol recognized by firewall products
- Security using HTTPS, replacing proprietary Open Servlet Engine (OSE) over Secure Sockets Layer (SSL)

Each of the supported web server plug-ins runs on a number of operating systems. See the Supported Hardware and Software website for the product for the most current information about supported web servers. This site is located at http://www.ibm.com/support/docview.wss?rs=180&uid=swg27006921.

gotcha: The default behavior for the web server plug-in is to buffer requests up to 64 kilobytes, and retry the requests if there is no response from the application server. If you want to ensure high availability, and your HTTP requests tend to be large, set the **Maximum buffer size used when reading HTTP request content** property on the web server plug-in request routing property page in the administrative console to `-1`. Setting this property to -1 removes the maximum buffer size limit, and enables the web server plug-in to buffer all requests regardless of their size. Requests are retried if the request body fits within the buffer size. If you want to disable all request buffering, and thereby disable retries of requests with request bodies, you can set this property to `0`.

Affinity requests

Affinity requests are requests that contain a JSESSIONID. Session affinity means that all requests of the same JSESSIONID are sent to the same application server. For example, if the first request is sent to clone5, then the next affinity request from that same browser is also sent to clone5 regardless of the weight valued specified for the LoadBalanceWeight property in the plugin-cfg.xml file.

If you select `Round robin` for the **Load balancing option** Web server plug-in request routing property, and leave the IgnoreAffinityRequests property in the plugin-cfg.xml file set to its default value of `true`, the affinity requests do not lower the weight. This behavior might cause an uneven distribution of requests across the servers in environments that make use of session affinity. Setting the IgnoreAffinityRequests property to `false` causes the weight to be lowered every time an affinity request is received, which results in a more balanced round robin environment.

If you select `Random` for the **Load balancing option** property, affinity requests are still sent to the same cloneid, but new requests are routed randomly, and the value specified for the LoadBalanceWeight property is ignored.

Failover

If a request connection exceeds the time limit specified on the ConnectTimeout property in the plugin-cfg.xml file, or a 5xx error is returned from the application server, the web server plug-in marks the server as down, and attempts to connect to the next application server in the list of primary servers that is specified for the PrimaryServers property in the plugin-cfg.xml file. If the web server plug-in successfully connects to another application server, all requests that were pending for the down application server are sent to this other application server. All other new and affinity requests are sent to other servers, based on whether round robin or random is the setting for the **Load balancing option** Web server plug-in request routing property.

Failover typically does not occur the first time that the time limit specified on the ServerIOTimeout property in the plugin-cfg.xml file is exceeded for either a request or a response. Instead, the web server plug-in tries to resend the request to the same application server, using a new stream. If the time specified on the ServerIOTimeout property is exceeded a second time, the web server plug-in marks the server as down, and initiates the failover process.

gotcha: Sending a large number of pending requests to the same application server might impact the performance of that application server if a failover situation occurs. You can use the MaxConnections property to limit the number of requests that might be pending for an application server.

Running multiple web server child processes

You can configure most web servers to start multiple child processes. In this situation, each child process loads its own instance of the web server. When running multiple web server child processes, remember that:

- Multiple running instances of the web server plug-in cannot share information. Therefore the dynamically changing load balance weight of each application server is not shared between the web server plug-in instances. For example, one instance of the web server plug-in might consider an application server to be running with a weight of 5, while another instance of the web server plug-in might be consider the same application server to be down and unusable. This difference in perspective might cause an incoming request to be handled differently, depending on which web server plug-in instance handles the request.
- The web server plug-in settings are handled on a per instance basis. For example, the MaxConnections property specifies the number of pending requests that are allowed on that web server, for each web server plug-in instance. If the MaxConnections property is set to `20`, and you start three web server child processes, each of the three web server plug-in instances allow 20 pending connections to the same application server, which means that there could be up to 60 pending connections.

Web server plug-in connections

The web server plug-ins are used to establish and maintain persistent HTTP and HTTPS connections to application servers.

When the plug-in is ready to send a request to the application server, it first checks its connection pool for existing connections. If an existing connection is available the plug-in checks its connection status. If the status is still good, the plug-in uses that connection to send the request. If a connection does not exist, the plug-in creates one. If a connection exists but has been closed by the application server, the plug-in closes that connection and opens a new one.

After a connection is established between a plug-in and an application server, it will not be closed unless the application server closes it for one of the following reasons:

- If the **Use Keep-Alive** property is selected and the time limit specified on the **Read timeout** or **Write timeout** property for the HTTP inbound channel has expired.
- The maximum number of persistent requests which can be processed on an HTTP inbound channel has been exceeded. This number is set using the **Maximum persistent requests** property that is specified for the HTTP inbound channel.
- The Application Server is shutting down.

Even if the application server closes a connection, the plug-in will not know that it has been closed until it tries to use it again. The connection will be closed if one of the following events occur:

- The plug-in receives a new HTTP request and tries to reuse the existing connection.
- The number of httpd processes drop because the web server is not receiving any new HTTP requests. For the IBM HTTP Server, the number of httpd processes that are kept alive depends on the value specified on the web server's MinSpareServers directive.
- The web server is stopped and all httpd processes are terminated, and their corresponding sockets are closed.

gotcha: Sometimes, if a heavy request load is stopped or decreased abruptly on a particular application server, a lot of the plug-in's connections to that application server will be in CLOSE_WAIT state. Because these connections will be closed the first time the plug-in tries to reuse them, having a large number of connections in CLOSE-WAIT state should not affect performance

Web server plug-in remote user information processing

You can configure your web server with a vendor-acquired authentication module and then configure the web server plug-in to route requests to an application server.

If an application calls the getRemoteUser method, it relies on a private HTTP header that contains the remote user information and is parsed by the plug-in. The plug-in sets the private HTTP header value whenever a web server authentication module populates the remote user in the web server data structure. If the private HTTP header value is not set, the call to getRemoteUser method by the application returns a null value.

- In the case of an Apache Web Server or the IBM HTTP Server, the plug-in builds the private header from the information contained in the associated request record.
- In the case of a Sun One Web Server, the plug-in builds the private header from the information contained in the **auth_user** property associated with the request. The private header is usually set to the name of the local HTTP user of the web browser, if HTTP access authorization is activated for the URL.
- In the case of a Domino Web Server, the plug-in builds the private header from the information contained in the **REMOTE_USER** environment variable. The plug-in sets this variable to **anonymous** for users who have not logged in and to the *username* for users who are logged into the application.
- In the case of an Internet Information Services (IIS) Web Server, the plug-in builds the private header from the information contained in the **REMOTE_USER** environment variable. The plug-in sets this variable to the name of the user as it is derived from the authorization header sent by the client.

Note: If the private header is not being set in the Sun One, IIS, or Domino Web Server plug-in, make sure the request record includes information about the user requesting the data.

Note: If an call to getRemoteUser method by the application returns a null value, or if the correct remote user information is not being added to the data structure for the web server plug-in, make sure the remote user parameter within the vendor-acquired authentication module is still set to **YES**. (Sometimes this parameter gets set to **NO** when service is applied.)

Private headers

A web server plug-in can use private headers to forward requests for dynamic content, such as servlets, to the application server.

After you configure a web server plug-in, in addition to regular plug-in functions, you can use private headers as a mechanism for forwarding proxy information from the plug-in to an application server. This information is not normally included in HTTP requests.

Private headers are implemented as a set of HTTP request header name and value pairs that the plug-in adds to the HTTP request header before the request is forwarded to an application server. The application server's web container removes this information from the header and then processes this information.

Private headers can include such information as the remote (client) user, the remote (client) host name, or an SSL client certificate. They conform to a naming standard so that there is no namespace collision with the architected HTTP header fields.

For example, authentication information, such as a client certificate, is normally requested by the web server once during the establishment of an HTTP session. It is not required again for individual requests within that session. However, a client certificate must accompany each request forwarded to the application server. The application server can then use the certificate as needed.

Similarly, the web server examines TCP/IP socket connections for information about the host address of the client. The application server cannot perform this examination because its socket connection is with the plug-in and not with the actual client. Therefore, one of the private headers is the host address of the actual client.

Gskit install images files

The Global Security Kit (GSKit) installation image files for the WebSphere Web server plug-ins are packaged on the CD with the web server plug-in files.

You can download the appropriate GSKIT file to the workstation on which your web server is running.

The Global Security Kit (GSKit) installation images files for the WebSphere Web Server Plug-ins are provided as a "local install" and are not installed globally nor are registered with the native package management utilities.

Plug-ins: Resources for learning

Use the following links to find relevant supplemental information about web server plug-ins. The information resides on IBM and non-IBM Internet sites, whose sponsors control the technical accuracy of the information.

These links are provided for convenience. Often, the information is not specific to the IBM WebSphere Application Server product, but is useful all or in part for understanding the product. When possible, links are provided to technical papers and Redbooks® that supplement the broad coverage of the release documentation with in-depth examinations of particular product areas.

View links to additional information about:
- Programming model and decisions
- Programming instructions and examples

Programming model and decisions

- Best Practice: WebSphere Plug-in Configuration Regeneration at http://www-128.ibm.com/developerworks/websphere/library/bestpractices/plug_in_configuration_regeneration.html

Programming instructions and examples

- WebSphere Application Server education at http://www-128.ibm.com/developerworks/search/searchResults.jsp?searchType=1&searchSite=dW&searchScope=dW&query=WebSphere+education
- Listing of all IBM WebSphere Application Server Redbooks at http://publib-b.boulder.ibm.com/Redbooks.nsf/Portals/WebSphere

Installing and configuring web server plug-ins

WebSphere Application Server supplies a unique binary plug-in module for each supported web server. The plug-in configuration file, which the WebSphere Application Server products create and maintain, interacts with the binary module to provide information about the application server configuration to the web server. The web server uses the information to determine how to communicate with the application server.

Before you begin

trns: In WebSphere Application Server Version 7 and earlier, you use the Plug-ins installation wizard to install the plug-in module, configure the web server for communicating with the application server, and create a web server configuration definition in the application server if possible. In WebSphere Application Server Version 8 and later, you must first install the Web Server Plug-ins, the web server, and the WebSphere Customization Toolbox; then, you run the Web Server Plug-ins Configuration Tool that is contained in the toolbox to configure the web server to communicate with the application server and, if possible, create a web server configuration definition in the application server. If you know how to set up your initial web server and plug-in configuration manually, you can do so in Version 7 and Version 8.

Go to http://www-1.ibm.com/support/docview.wss?rs=180&context=SSEQTP&uid=swg21160581 for information about how to verify what Version 4.0 through Version 8.0 plug-in versions are installed on local or remote web servers and how to determine if the installation complies with supported configurations.

You must install a supported web server before you can install and configure a plug-in for the web server.

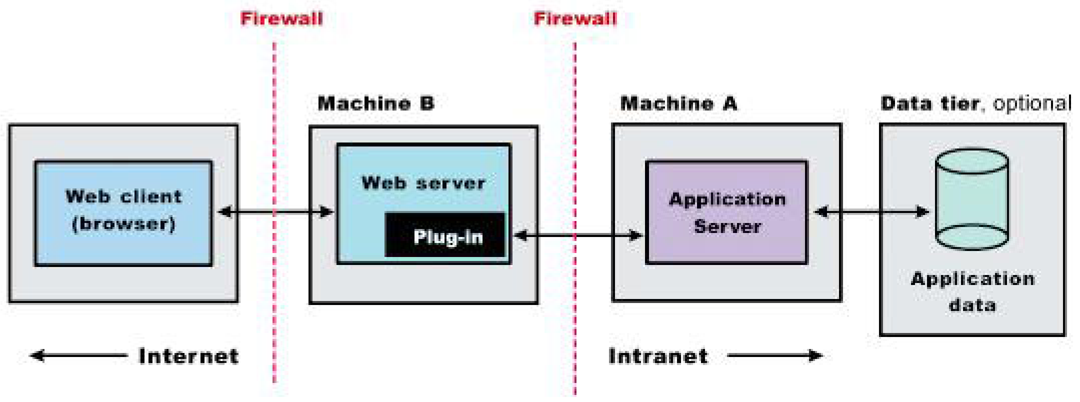
About this task

This article describes installing and configuring web server plug-ins for WebSphere Application Server. WebSphere Application Server products supplies a unique binary plug-in module for each supported web server. The plug-in configuration file, which the WebSphere Application Server products create and maintain, interacts with the binary module to provide information about the application server configuration to the web server. The web server uses the information to determine how to communicate with the application server.

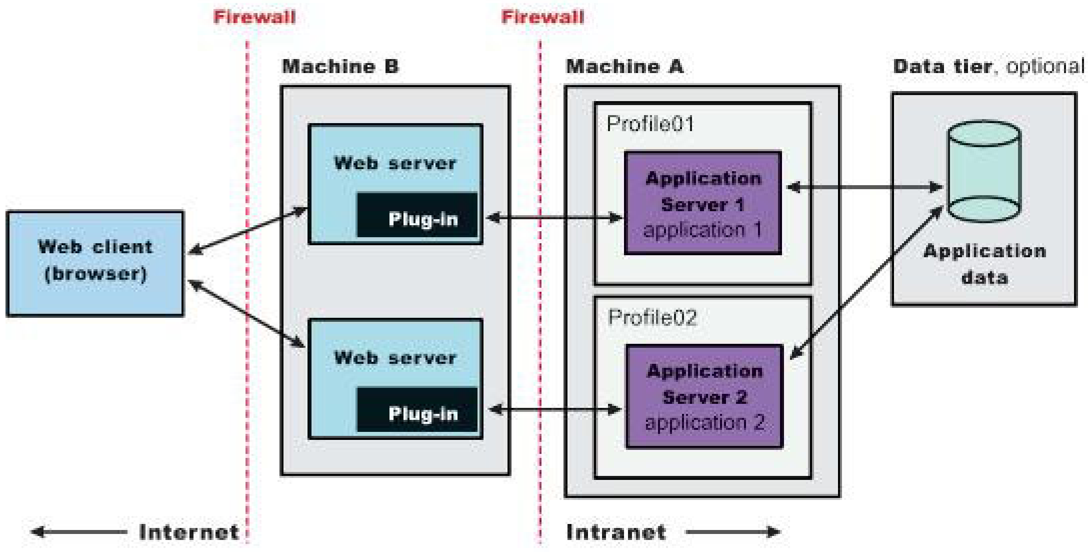
Select one of the following topology scenarios and follow the steps below the diagram to install the plug-in and configure both the web server and the application server.

Procedure

- **Scenario 1: Local application server profile** The application server and the web server are on a single system or logical partition.
- **Scenario 2: Remote** The application server and the web server are on separate machines or logical partitions.



- **Scenario 3: Remote** Multiple standalone application servers are on one system, and each application server has a dedicated web server on a separate system or logical partition.



Results

You can install a web server and the web server plug-ins for various standalone application server topologies by following the procedures described in this article.

What to do next

See “Selecting a web server topology diagram and roadmap” for an overview of the installation procedure.

Selecting a web server topology diagram and roadmap

Install and configure the Web Server Plug-ins to allow the application server to communicate with the web server.

Before you begin

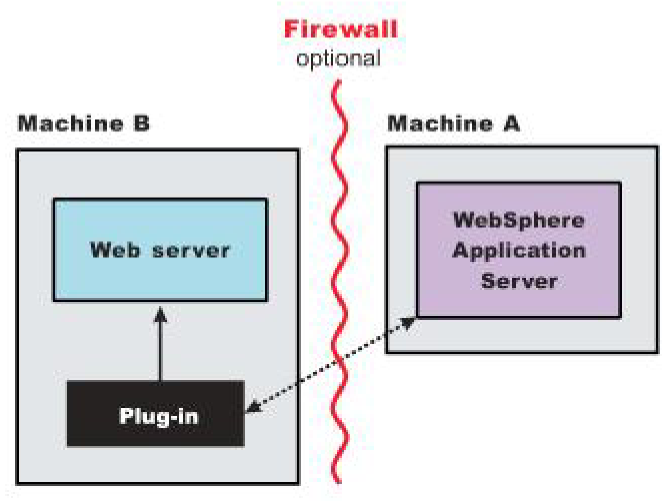
The primary production configuration for a web server is an application server on one machine and a web server on a separate machine. This configuration is referred to as a *remote* configuration. Contrast the remote configuration to the local configuration, where the application server and the web server are on the same machine.

Procedure

- **Set up a remote web server installation for a standalone node.**

  The remote web server configuration is recommended for production environments.

  The remote installation installs the web server plug-in on the web server machine when the application server is on a separate machine, such as shown in the following graphic:



Remote installation scenario

Table 3. Installation and configuration. Remote installation scenario

Step	Machine	Task
1	A	Install your WebSphere Application Server product.
2	A	Create an application server profile.
3	B	If you plan to run IBM HTTP Server on IBM i, it is already installed as product 5761-DG1 or 5770-DG1. You can also use a Domino Web Server on IBM i. Refer to the Domino documentation for installation instructions. For either scenario, you must install the Web Server Plug-ins component of the WebSphere Application Server product.
4	B	Run the **manageprofiles** Qshell command to create an http profile. For example, run this command from Qshell: `app_server_root/bin/manageprofiles -create -profileName myHttpProfile -templatePath http` The *myHttpProfile* variable is the name of the profile.

Table 3. Installation and configuration (continued). Remote installation scenario

Step	Machine	Task

5	B	Configure the IBM HTTP Server with your http profile myHttpProfile.
		
		As a result of the above, an IBM i qshell script called *configureIHS_MyWebServer* will be created in the `myHttpProfile_profile_root/config/IHS_myWebServer` directory on Machine B. For the default WebSphere Application Server install, the*myHttpProfile_profile_root* of the profile myHttpProfile is `/QIBM/UserData/WebSphere/AppServer/V61/Base/profiles/myHttpProfile`.
		
		Note: In the remainder of this example, webServerName refers to IHS_myWebServer. If you choose to configure a DOMINO web server as listed below, then webServerName refers to DOMSRV01.
		
		The following steps apply to **DOMINO web servers only**:
		1. Run the *configureOs400WebServerDefinition* script on the http profile myHttpProfile. For example:
		`configureOs400WebServerDefinition -profileName myHttpProfile`
		`   -webserver.name DOMSRV01 -webserver.type DOMINO -webserver.port 80`
		2. Using the WRKDOMSVR command to update the notes.ini file of your Domino server, insert the following directive: `WebSphereInit=myHttpProfile_profile_root/config/DOMSRV01/plugin-cfg.xml`
		3. From the Lotus Notes® client connected to the Domino server, click the internet protocols tab and then click the HTTP tab. Under DSAPI filter names add the following: `/QSYS.LIB/` *product_lib*`.LIB/LIBDOMINO.SRVPGM`
		4. Save your changes

6	A	Copy the *configurewebServerName* script from Machine B to Machine A. The script is found in the `myHttpProfile_profile_root/config/webServerName` directory described above
7	A	Place the file you copied from the previous step into the `profile_root/bin` directory on Machine A, where *profile_root* is the directory where your application server profile is located.

8	A	Start the application server and then run the script that you copied in the previous step.
		
		For example, run these commands from Qshell:
		`app_server_root/bin/startServer`
		`  -profileName myProfile`
		
		`cd profile_root/bin`
		
		`./configurewebServerName [wasAdminUserId] [wasAdminPassword]`
		
		Note: *wasAdminUserId* and *wasAdminPassword* are optional and only needed when the application server of *myProfile* is running in secure mode.

9	A	If you use IBM HTTP Server on IBM i, verify that the application server is running. Open the administrative console (ISC) and do the following: 1. Expand **Servers > Server Types > Web servers** 2. Select your web server, in this case it is *IHS_MyWebServer*, then click Remote web server management 3. Enter the user ID and password used to authenticate to Machine B. The authorities required by this profile are the same as that required to access the HTTP administration GUI. For details, see User profiles and required authorities for HTTP Server in the IBM i Information Center. 4. Save your configuration.
10	A	Configure a virtual host alias for the web server machine (B) and web server port of MyWebServer.
11	A	Stop and restart your application server.
12	A	In the administrative console (ISC) do the following: 1. Select *webServerName* and click Generate Plug-in to generate the plugin-cfg.xml file. 2. Select *webServerName* and click Propagate Plug-in to propagate the plugin-cfg.xml file to Machine B.
13	B	If you use IBM HTTP Server on IBM i, start the web server. Open the administrative console (ISC) and do the following: 1. Expand **Servers > Server Types > Web servers**. 2. Select your web server, in this case it is *IHS_MyWebServer*, then click Start. If you use Domino HTTP Server on IBM i, start the web server from a CL command line: 1. Run the Work with Domino Servers (WRKDOMSVR) command. 2. Specify option 1 next to your Domino server. 3. Press Enter.
14	B	Run the Snoop servlet. Access the following URL in your browser: `http://host_name_of_machine_B:http_transport_port/Snoop` If you get an error, retrace your steps. Add a virtual host to Machine A before restarting the application server on Machine A.

Regeneration of the `plugin-cfg.xml` file

The web server plug-in configuration service regenerates the `plugin-cfg.xml` file automatically.

However, if the service is disabled or if you want to force regeneration, use the administrative console or the `GenPluginCfg` script. In the administrative console, perform these steps:

1. Expand **Servers** > **Server Types** > **Web servers** .
2. Select the web server for which you want to regenerate the `plugin-cfg.xml` file.
3. Click **Generate Plugin**.

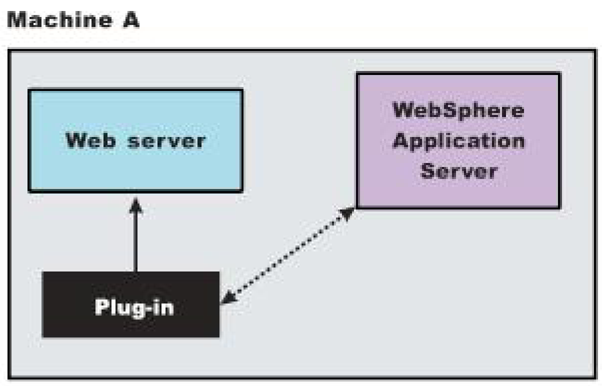
Propagation of the `plugin-cfg.xml` file

The `plugin-cfg.xml` file can be propagated manually. Manual propagation of the `plugin-cfg.xml` file is required for DOMINO web servers. Copy the `plugin-cfg.xml` file from the application server machine to the `myHttpProfile_profile_root/config/IHS_MyWebServer` directory on the web server Machine B. The `plugin-cfg.xml` file is generated in the directory named `profile_root/config/cells/cell_name/nodes/host_name_of_machine_B-node/servers/IHS_myWebServer` on the application server Machine A.

- **Set up a local web server configuration for a standalone node.**

  The local web server configuration is recommended for a development or test environment.

  A local installation includes the web server plug-in, the web server, and the Application Server on the same machine:



Local installation scenario

Table 4. Installation and configuration. Local installation scenario

Step	Machine	Task
1	A	Install your WebSphere Application Server product.
2	A	Create an application server profile.
3	A	IBM HTTP Server on IBM i is already installed as product 5761-DG1 or 5770-DG1. Alternatively, you can also run Domino Web Server on IBM i. Refer to the Domino documentation for installation instructions.

4	A	Configure the IBM HTTP Server with your WebSphere Application Server profile.
		
		The following steps apply to DOMINO web servers only. For these steps, assume your web server's name is *MyWebServer*.
		1. Run the *configureOs400WebServerDefinition* script on the application server profile. For example:
		`configureOs400WebServerDefinition -profileName myAppServerProfile -webserver.name DOMSRV01 -webserver.type DOMINO -webserver.port 80`
		2. Configure a virtual host alias for the web server machine and web server port of DOMSRV01.
		3. Using the WRKDOMSVR command to update the notes.ini file of your Domino server, insert the following directive:
		`WebSphereInit=profile_root/config/cells/cell_name/nodes/node_name/servers/DOMSRV01/plugin-cfg.xml`
		4. From the Lotus Notes client connected to the Domino server, click the Internet protocols tab, then click the HTTP tab. Under DSAPI filter names add the following: `/QSYS.LIB/` *product_lib*`.LIB/LIBDOMINO.SRVPGM`
		5. Save your changes

Table 4. Installation and configuration (continued). Local installation scenario

Step	Machine	Task
5	A	Stop and restart the application server.
6	A	If you use IBM HTTP Server on IBM i, open the administrative console (ISC) do the following: 1. Expand **Servers > Server Types > Web servers**. 2. Select your web server, in this case it is *IHS_MyWebServer*, then click Remote web server management. 3. Enter the user ID and password used to authenticate to Machine A. The authorities required by this profile are the same as that required to access the HTTP administration GUI. For details, see User profiles and required authorities for HTTP Server in the IBM i Information Center. 4. Save your changes.
7	A	In the administrative console (ISC) do the following: 1. Expand **Servers > Server Types > Web servers**. 2. If you use IBM HTTP Server on IBM i, Select *IHS_MyWebServer* and click Generate Plug-in to generate the plugin-cfg.xml file. 3. If you use Domino HTTP Server on IBM i, Select DOMSRV01 and click Generate Plug-in to generate the plugin-cfg.xml file.
8	A	If you use IBM HTTP Server on IBM i, start the web server. Open the administrative console (ISC) and do the following: 1. Expand **Servers > Server Types > Web servers**. 2. Select your web server, in this case it is *IHS_MyWebServer*, then click Start. If you use Domino HTTP Server on IBM i, start the web server from a CL command line: 1. Run the Work with Domino Servers (WRKDOMSVR) command. 2. Specify option 1 next to your Domino server. 3. Press Enter.
9	A	Run the Snoop servlet. Access the following URL in your browser: `http://`*`host_name_of_machine_A`*`:`*`http_transport_port`*`/Snoop` If you get an error, retrace your steps. Add a *virtual host* to Machine A before restarting the application server on Machine A.

Regeneration of the `plugin-cfg.xml` file

The web server plug-in configuration service regenerates the `plugin-cfg.xml` file automatically.

However, if the service is disabled or you want to force regeneration, use the administrative console or the GenPluginCfg script. In the administrative console, perform these steps:

1. Expand **Servers > Server Types > Web servers**.
2. Select the web server for which you want to regenerate the `plugin-cfg.xml` file.
3. Click **Generate Plug-in**.

Propagation of the `plugin-cfg.xml` file

The local file does not require propagation.

Results

You can set up a remote or local web server by installing Application Server, the web server, and then the Web Server Plug-ins.

What to do next

See "Installing and configuring web server plug-ins" on page 33 for information about other installation scenarios for installing Web Server Plug-ins.

Installing and uninstalling the Web Server Plug-ins on IBM i operating systems

IBM Installation Manager is a common installer for many IBM software products that you use to install or uninstall the Web Server Plug-ins.

About this task

For more information on using Installation Manager, read the IBM Installation Manager Information Center.

Perform one of these procedures to install or uninstall the product using Installation Manager.

Procedure

- "Installing the Web Server Plug-ins on IBM i operating systems using response files"
- "Uninstalling the Web Server Plug-ins from IBM i operating systems using response files" on page 42

Results

- The following locations are the defaults for Installation Manager files on IBM i systems:
  - **Installation location:** `/QIBM/ProdData/InstallationManager`
  - **Agent data location:** `/QIBM/UserData/InstallationManager`
  - **Registry:** `/QIBM/InstallationManager/.ibm/registry/InstallationManager.dat`
- Logs are located in the `logs` directory of Installation Manager's agent data location. For example:

```
/QIBM/UserData/InstallationManager/logs
```

  The main log files are time-stamped XML files in the `logs` directory, and they can be viewed using any standard web browser.

Installing the Web Server Plug-ins on IBM i operating systems using response files

You can use Installation Manager to install the Web Server Plug-ins using response files.

Before you begin

Before you install the Web Server Plug-ins, ensure that your user profile has *ALLOBJ and *SECADM special authorities.

Install Installation Manager on the system onto which you want to install the product.

- If you want to use the Installation Manager that comes with this product, perform the following actions:
  1. Obtain the necessary files from the physical media or the web.

     There are three basic options for obtaining and installing the product.
     - **Access the physical media, and use local installation**

       You can access the product repositories on the product media.
       a. Install Installation Manager on your system.

          You can install Installation Manager using the product media, using a file obtained from the Passport Advantage® site, or using a file containing the most current version of Installation Manager from the IBM Installation Manager website.
       b. Use Installation Manager to install the product from the product repositories on the media.
     - **Download the files from the Passport Advantage site, and use local installation**

       Licensed customers with a Passport Advantage ID and password can download the necessary product repositories from the Passport Advantage site.
       a. Download the files from the Passport Advantage site.
       b. Install Installation Manager on your system.

          You can install Installation Manager using the product media, using a file obtained from the Passport Advantage site, or using a file containing the most current version of Installation Manager from the IBM Installation Manager website.
       c. Use Installation Manager to install the product from the downloaded repositories.
     - **Access the live repositories, and use web-based installation**

If you have a Passport Advantage ID and password, you can install the product from the web-based repositories.

a. Install Installation Manager on your system.

   You can install Installation Manager using the product media, using a file obtained from the Passport Advantage site, or using a file containing the most current version of Installation Manager from the IBM Installation Manager website.

b. Use Installation Manager to install the product from the web-based repository located at

```
http://www.ibm.com/software/repositorymanager/com.ibm.websphere.PLG.v80
```

Whenever possible, you should use the remote web-based repositories so that you are accessing the most up-to-date installation files.

Note: If you do not have a Passport Advantage ID and password, you must install the product from the product repositories on the media or local repositories.

2. Install Installation Manager silently.

   a. Sign on to the IBM i system with a user profile that has *ALLOBJ and *SECADM special authorities.

   b. On a CL command line, run the STRQSH command to start the Qshell command shell.

   c. Make sure that the umask is set to 022.

      To verify the umask setting, issue the following command:

```
umask
```

      To set the umask setting to 022, issue the following command:

```
umask 022
```

   d. Change to the temporary directory where you unpacked the Installation Manager files.

   e. Run the following command in the temporary folder:

```
installc -acceptLicense -log log_file_path_and_name
```

      Notes:

      – For more information on installing Installation Manager silently, see the IBM Installation Manager Information Center.

      – Use only the installc command to install Installation Manager.

- If you already have a version of Installation Manager installed on your system and you want to use it to install and maintain the product, obtain the necessary product files from the physical media or the web.

  There are three basic options for installing the product.

  – **Access the physical media, and use local installation**

    You can access the product repositories on the product media. Use Installation Manager to install the product from the product repositories on the media.

  – **Download the files from the Passport Advantage site, and use local installation**

    Licensed customers with a Passport Advantage ID and password can download the necessary product repositories from the Passport Advantage site.

    1. Download the product repositories from the Passport Advantage site.
    2. Use Installation Manager to install the product from the downloaded repositories.

  – **Access the live repositories, and use web-based installation**

    If you have a Passport Advantage ID and password, you can use Installation Manager to install the product from the web-based repositories. Use Installation Manager to install the product from the web-based repository located at

```
http://www.ibm.com/software/repositorymanager/com.ibm.websphere.PLG.v80
```

    Whenever possible, you should use the remote web-based repositories so that you are accessing the most up-to-date installation files.

Note: If you do not have a Passport Advantage ID and password, you must install the product from the product repositories on the media or local repositories.

About this task

Using Installation Manager, you can work with response files to install the Web Server Plug-ins.

Procedure

1. Optional: If the repository requires a username and password, create a keyring file to access this repository.

   For more information on creating a keyring file for Installation Manager, read the IBM Installation Manager Information Center.
2. Sign on to the IBM i system with a user profile that has *ALLOBJ and *SECADM special authorities.
3. On a CL command line, run the STRQSH command to start the Qshell command shell.
4. Make sure that the umask is set to 022.

   To verify the umask setting, issue the following command:

```
umask
```

   To set the umask setting to 022, issue the following command:

```
umask 022
```

5. Use a response file to install the Web Server Plug-ins silently.

   Change to the `eclipse/tools` subdirectory in the directory where you installed Installation Manager, and install the Web Server Plug-ins silently. For example:

```
./imcl -acceptLicense
  input $HOME/WASFiles/temp/install_response_file.xml
  -log $HOME/WASFiles/temp/install_log.xml
  -keyring $HOME/WASFiles/temp/im.keyring
```

   Notes:

   - The relevant terms and conditions, notices, and other information are provided in the license-agreement files in the `lafiles` or *`product_name`*`/lafiles` subdirectory of the installation image or repository for this product.
   - `/QIBM/ProdData/InstallationManager` is the default installation location for Installation Manager files on IBM i systems.
   - The program might write important post-installation instructions to standard output.

   Read the IBM Installation Manager Information Center for more information.

Example

The following is an example of a response file for silently installing the Web Server Plug-ins.

```
<?xml version="1.0" encoding="UTF-8"?>
<agent-input>
<server>
  <repository location='http://www.ibm.com/software/repositorymanager/com.ibm.websphere.PLG.v80'/>
</server>
<profile id='Web Server Plug-ins for IBM WebSphere Application Server V8.0' installLocation='/QIBM/ProdData/WebSphere/Plugins/V8/webserver'>
  <data key='eclipseLocation' value='/QIBM/ProdData/WebSphere/Plugins/V8/webserver'/>
  <data key='was.install.os400.profile.location' value='/QIBM/UserData/WebSphere/Plugins/V8/webserver'/>
  <data key='user.import.profile' value='false'/>
  <data key='cic.selector.nl' value='en'/>
</profile>
<install modify='false'>
  <offering profile='Web Server Plug-ins for IBM WebSphere Application Server V8.0' features='core.feature' id='com.ibm.websphere.PLG.v80'/>
</install>
<preference name='com.ibm.cic.common.core.preferences.eclipseCache' value='/QIBM/UserData/InstallationManager/IMShared'/>
<preference name='com.ibm.cic.common.core.preferences.connectTimeout' value='30'/>
<preference name='com.ibm.cic.common.core.preferences.readTimeout' value='30'/>
<preference name='com.ibm.cic.common.core.preferences.downloadAutoRetryCount' value='0'/>
<preference name='offering.service.repositories.areUsed' value='true'/>
<preference name='com.ibm.cic.common.core.preferences.ssl.nonsecureMode' value='false'/>
<preference name='com.ibm.cic.common.core.preferences.http.disablePreemptiveAuthentication' value='false'/>
<preference name='http.ntlm.auth.kind' value='NTLM'/>
```

```
<preference name='http.ntlm.auth.enableIntegrated.win32' value='true'/>
<preference name='com.ibm.cic.common.core.preferences.preserveDownloadedArtifacts' value='true'/>
<preference name='com.ibm.cic.common.core.preferences.keepFetchedFiles' value='false'/>
<preference name='PassportAdvantageIsEnabled' value='false'/>
<preference name='com.ibm.cic.common.core.preferences.searchForUpdates' value='false'/>
</agent-input>
```

What to do next

You can create a plug-ins profile using the **manageprofiles** command.

The following is an example of using the **manageprofiles** command to create a default plug-ins profile.

```
manageprofiles -create
  -templatePath /QIBM/ProdData/WebSphere/Plugins/V8/webserver/profileTemplates/http
  -profileName PluginsSrv01
```

Note: After you create a profile successfully, the console prints a message that indicates success and advises you to check the `AboutThisProfile.txt` file. However, a `AboutThisProfile.txt` file is not generated when you create a plug-ins profile on IBM i.

The following is an example of using the **manageprofiles** command to make profile PluginsSrv01 the default profile.

```
manageprofiles -setDefaultName -profileName PluginsSrv01
```

Uninstalling the Web Server Plug-ins from IBM i operating systems using response files

You can use Installation Manager to uninstall the on IBM i operating systems using response files.

Before you begin

Removing the plug-ins will break communication between any web servers and their associated application servers. Be sure to unconfigure any web and applications servers that are using these plug-ins before uninstalling them.

About this task

Using Installation Manager, you can work with response files to uninstall the Web Server Plug-ins.

Procedure

1. Stop all servers and applications on the WebSphere Application Server installations that contain the Web Server Plug-ins.
2. Sign on to the IBM i system with a user profile that has *ALLOBJ and *SECADM special authorities.
3. On a CL command line, run the STRQSH command to start the Qshell command shell.
4. Use a response file to uninstall the Web Server Plug-ins silently.

   Complete one of the following actions:

   - From a command line on each of the systems from which you want to uninstall the Web Server Plug-ins, run the **uninstall** script (which uses the `uninstall.xml` response file in the same directory) to silently uninstall the Web Server Plug-ins. For example:

```
app_server_root/uninstall/uninstall
```

   - From a command line on each of the systems from which you want to uninstall the Web Server Plug-ins, change to the `eclipse/tools` subdirectory in the directory where you installed Installation Manager and use the `uninstall.xml` response file in the same directory to silently uninstall the Web Server Plug-ins. For example:

```
./imcl
  input app_server_root/uninstall/uninstall/uninstall.xml
  -log $HOME/WASFiles/temp/uninstall_log.xml
```

- From a command line on each of the systems from which you want to uninstall the Web Server Plug-ins, change to the `eclipse/tools` subdirectory in the directory where you installed Installation Manager and use a response file that you created to silently uninstall the on IBM i operating systems. For example:

```
./imcl
  input $HOME/WASFiles/temp/uninstall_response_file.xml
  -log $HOME/WASFiles/temp/uninstall_log.xml
```

  Go to the Installation Manager information center for more information.

5. Optional: Uninstall IBM Installation Manager.

   Important: Before you can uninstall IBM Installation Manager, you must uninstall all of the packages that were installed by Installation Manager.

   Read Uninstalling Installation Manager silently in the Installation Manager information center for information about using the uninstall script to perform this procedure.

Example

The *app_server_root*`/uninstall/uninstall/uninstall.xml` file is an example of a response file for silently uninstalling the Web Server Plug-ins.

Web server configuration

Plug-in configuration involves configuring the web server to use the binary plug-in module that WebSphere Application Server provides. Plug-in configuration also includes updating the plug-in XML configuration file to reflect the current application server configuration. The binary module uses the XML file to help route web client requests.

The plug-ins configuration process uses the following files to configure a plug-in for the web server that you select:

- The **web server configuration file** on the web server machine, such as the `httpd.conf` file for IBM HTTP Server.
- The **binary web server plug-in file** on the web server machine.
- The **plug-in configuration file, `plugin-cfg.xml`**, on the application server machine that you propagate (copy) to a Web server machine.
- The **configuration script** for configuring the web server definition for your application server in a remote HTTP scenario.

See the following descriptions of each file.

Web server configuration file

The web server configuration file is installed as part of the web server.

Configuration consists of adding directives that identify file locations of two files:

- Binary web server plug-in file
- Plug-in configuration file, `plugin-cfg.xml`

Binary web server plug-in file

An example of a binary plug-in module is the `mod_was_ap22_http.dll` file for IBM HTTP Server on the Windows platform.

Another example of a binary plug-in module is the QSVTAP20 service program on the IBM i platform.

The binary plug-in file does not change. However, the configuration file for the binary plug-in is an XML file. The application server changes the configuration file when certain changes to your WebSphere Application Server configuration occur.

The binary module reads the XML file to adjust settings and to route requests to the application server.

Plug-in configuration file, `plugin-cfg.xml`

The plug-in configuration file is an XML file with settings that you can tune in the administrative console. The file lists all of the applications installed on the web server definition. The binary module reads the XML file to adjust settings and to route requests to the application server.

When you make application server configuration changes that affect deployed applications, regenerate the plug-in configuration XML file.

After regeneration, propagate (copy) the file to the web server machine. The binary plug-in then has access to the most current copy of its configuration file.

On IBM i systems, the plug-in is not automatically generated. You must regenerate and propagate the file manually.

Configuration script for the Web server definition

Configuring your web server with the `configureOs400WebserverDefinition` script or using the IBM i Administrative GUI creates the `configure`*`web_server_name`* script on the web server machine in the *plugins_root*/`bin` directory. The script is created for remote installation scenarios only.

Copy the script from the web server machine to the *app_server_root*/`bin` directory in the IBM i partition. Run the script to create a web server definition in the configuration of the application server.

The IBM i Administrative GUI has plug-ins that allow the administrative console to manage IBM HTTP Servers. Use the administrative console to update your web server definition with remote web server management options. Click **Servers > Server Types > Web servers > *web_server_name*** to see configuration options. For example, click **Remote Web server management** to change such properties as:

- Host name
- Administrative port
- User ID
- Password

If a web server definition already exists for a standalone application server, running the script does not add a new web server definition. Each standalone application server can have only one Web server definition.

You cannot use the administrative console of a standalone application server to add or delete a web server definition. However, you can do both tasks using the administrative scripting interface:

- Add a web server definition through the wsadmin facility using the `configure`*`web_server_name`* script. The script uses a Java Command Language (Jacl) script named `configureWebserverDefintion.jacl` to create and configure the web server definition.
- Delete a web server definition using wsadmin commands. The Web server is named webserver1 in the following example:

```
set webserverName webserver1
set webserverNodeSuffix _node
set webserverNodeName
$webserverName$webserverNodeSuffix
$AdminConfig remove
  [$AdminConfig getid
```

```
    /Node:$webserverNodeName/Server:$webserverName]
$AdminConfig remove
  [$AdminConfig getid /Node:$webserverNodeName]
$AdminConfig save
```

Alternatively, you can use the `configureOs400WebServerDefinition` and `removeOs400WebServerDefinition` scripts to perform these tasks.

Replacing the default plug-in configuration file with the file from the web server definition (propagation)

The default file uses fixed parameter values that might not match the parameter values in the actual file on the application server. The default file is a placeholder only.

The file cannot reflect changes that occur in the application server configuration. The file also cannot reflect nondefault values that might be in effect on the application server.

Creating or updating a global web server plug-in configuration file

If all of the application servers in a cell use the same web server to route requests for dynamic content, such as servlets, from web applications to application servers, you can create a global web server plug-in configuration file for that cell. The resulting `plugin-cfg.xml` file is located in the `profile_root/config/cells` directory.

Before you begin

Note: You can update the *global* `plugin-cfg.xml` file using the administrative console or running the GenPluginCfg command for all of the clusters in a cell. However, you must delete the `config/cells/plugin-cfg.xml` file before you update the *global* `plugin-cfg.xml` file. If you do not delete the `config/cells/plugin-cfg.xml` file, only the new properties and their values are added to the *global* `plugin-cfg.xml` file. Any updates to existing plug-in property values are not added to the *global* `plugin-cfg.xml` file.

About this task

You must update the global web server plug-in configuration file whenever you perform one of the following actions:

- Change the configuration settings for an application server, cluster, virtual host, or web container transport that is part of that cell.
- Add a new application server, cluster, virtual host, or web container transport to that cell.

To update the configuration settings for a global web server plug-in, you can either use the Update global web server plug-in configuration page in the administrative console, or issue the following command:

```
%was_profile_home%/config/cells/GenPluginCfg.sh|bat
```

Both methods for regenerating the global web server plug-in configuration create a `plugin-cfg.xml` file in ASCII format.

To use the Update global web server plug-in configuration page in the administrative console:

Procedure

1. Click **Environment > Update global web server plug-in configuration**.
2. Click **OK** to update the `plugin-cfg.xml` file.
3. Optional: Click **View or download the current web server plug-in configuration file** if you want to view or download the current version of this file.

   You can select this option if you want to:

- View the current version of the file before you update it.
- View the file after it is updated.
- Download a copy of this file to a remote machine.

Results

Regenerating the configuration might take a while to complete. After it finishes, all objects in the administrative cell use their newest settings, which the web server can access. Whether triggered manually or occurring automatically, plug-in regeneration requires about 30 to 60 seconds to complete when the application server is on the same physical machine (node) as the web server. In other cases, it takes more time.

The delay is important because it determines how soon the new plug-in configuration takes effect. Suppose you add a new served path for a servlet, then regenerate the plug-in configurations. The regeneration requires 40 seconds, after which a user should be able to access the servlet by the new served path.

For an HTTP plug-in, the length of the delay is determined by the Refresh Interval attribute of the Config element in the `plugin-cfg.xml` file. The plug-in polls the disk, or file system, at this interval to see whether the configuration has changed. The default interval is 60 seconds. To regenerate the plug-in configuration requires twice the refresh interval.

In a development environment in which you are frequently changing settings in the administrative console, it is recommended that you set the refresh interval to 3 to 5 seconds.

In a production environment, set a longer refresh interval, perhaps as long as 30 minutes, depending on the frequency of changes.

What to do next

You might need to stop the application servers in the cell and then start the application servers again before the changes to the plug-in configuration go into effect.

If the web server is running on a remote machine, click **View or download the current web server plug-in configuration file** to download a copy of the `plugin-cfg.xml` file to a that machine.

Creating or updating a global web server plug-in configuration file

If all of the application servers in a cell use the same web server to route requests for dynamic content, such as servlets, from web applications to application servers, you can create a global web server plug-in configuration file for that cell. The resulting `plugin-cfg.xml` file is located in the `profile_root/config/cells` directory.

Before you begin

Note: You can update the *global* `plugin-cfg.xml` file using the administrative console or running the GenPluginCfg command for all of the clusters in a cell. However, you must delete the `config/cells/plugin-cfg.xml` file before you update the *global* `plugin-cfg.xml` file. If you do not delete the `config/cells/plugin-cfg.xml` file, only the new properties and their values are added to the *global* `plugin-cfg.xml` file. Any updates to existing plug-in property values are not added to the *global* `plugin-cfg.xml` file.

About this task

You must update the global web server plug-in configuration file whenever you perform one of the following actions:

- Change the configuration settings for an application server, cluster, virtual host, or web container transport that is part of that cell.
- Add a new application server, cluster, virtual host, or web container transport to that cell.

To update the configuration settings for a global web server plug-in, you can either use the Update global web server plug-in configuration page in the administrative console, or issue the following command:

```
%was_profile_home%/config/cells/GenPluginCfg.sh|bat
```

Both methods for regenerating the global web server plug-in configuration create a `plugin-cfg.xml` file in ASCII format.

To use the Update global web server plug-in configuration page in the administrative console:

Procedure

1. Click **Environment > Update global web server plug-in configuration**.
2. Click **OK** to update the `plugin-cfg.xml` file.
3. Optional: Click **View or download the current web server plug-in configuration file** if you want to view or download the current version of this file.

   You can select this option if you want to:

   - View the current version of the file before you update it.
   - View the file after it is updated.
   - Download a copy of this file to a remote machine.

Results

Regenerating the configuration might take a while to complete. After it finishes, all objects in the administrative cell use their newest settings, which the web server can access. Whether triggered manually or occurring automatically, plug-in regeneration requires about 30 to 60 seconds to complete when the application server is on the same physical machine (node) as the web server. In other cases, it takes more time.

The delay is important because it determines how soon the new plug-in configuration takes effect. Suppose you add a new served path for a servlet, then regenerate the plug-in configurations. The regeneration requires 40 seconds, after which a user should be able to access the servlet by the new served path.

For an HTTP plug-in, the length of the delay is determined by the Refresh Interval attribute of the Config element in the `plugin-cfg.xml` file. The plug-in polls the disk, or file system, at this interval to see whether the configuration has changed. The default interval is 60 seconds. To regenerate the plug-in configuration requires twice the refresh interval.

In a development environment in which you are frequently changing settings in the administrative console, it is recommended that you set the refresh interval to 3 to 5 seconds.

In a production environment, set a longer refresh interval, perhaps as long as 30 minutes, depending on the frequency of changes.

What to do next

You might need to stop the application servers in the cell and then start the application servers again before the changes to the plug-in configuration go into effect.

If the web server is running on a remote machine, click **View or download the current web server plug-in configuration file** to download a copy of the `plugin-cfg.xml` file to a that machine.

Update the global web server plug-in configuration setting

Use this page to create or update a global plug-in configuration file. The configuration settings this file contains are based on the topology of the cell that contains the applications servers that use this web server plug-in. The web server plug-in configuration file settings determine whether an application server or the web server handles user requests.

A global web server plug-in configuration file must be regenerated whenever:
- You change the configuration settings for an application server, cluster, web container transport, or virtual host alias that is contained in the cell.
- You add a new application server, cluster, web container transport, or virtual host alias to the cell.

The generated `plugin-cfg.xml` file is placed in the `%was_profile_home%/config/cells` directory. If your web server is located on a remote machine, you must manually move this file to that machine.

Note: You can update the *global* `plugin-cfg.xml` file using the administrative console or running the GenPluginCfg command for all of the clusters in a cell. However, you must delete the `config/cells/plugin-cfg.xml` file before you update the *global* `plugin-cfg.xml` file. If you do not delete the `config/cells/plugin-cfg.xml` file, only the new properties and their values are added to the *global* `plugin-cfg.xml` file. Any updates to existing plug-in property values are not added to the *global* `plugin-cfg.xml` file.

To view this administrative console page, click **Environment > Update global web server plug-in configuration**.

Click **OK** to update the global `plugin-cfg.xml` file.

Click **View or download the current web server plug-in configuration file** if you want to:
- View the current version of the file before you update it.
- View the file after it is updated.
- Download a copy of this file to a remote machine.

Directory conventions

References in product information to *app_server_root*, *profile_root*, and other directories imply specific default directory locations. This topic describes the conventions in use for WebSphere Application Server.

Default product locations - IBM i

These file paths are default locations. You can install the product and other components in any directory where you have write access. You can create profiles in any valid directory where you have write access. Multiple installations of WebSphere Application Server products or components require multiple locations.

app_client_root
: The default installation root directory for the Application Client for IBM WebSphere Application Server is the `/QIBM/ProdData/WebSphere/AppClient/V8/client` directory.

app_client_user_data_root
: The default Application Client for IBM WebSphere Application Server user data root is the `/QIBM/UserData/WebSphere/AppClient/V8/client` directory.

app_client_profile_root
: The default Application Client for IBM WebSphere Application Server profile root is the `/QIBM/UserData/WebSphere/AppClient/V8/client/profiles/`*`profile_name`* directory.

app_server_root
: The default installation root directory for WebSphere Application Server - Express is the `/QIBM/ProdData/WebSphere/AppServer/V8/Express` directory.

java_home

Table 5. Root directories for supported Java Virtual Machines.

This table shows the root directories for all supported Java Virtual Machines (JVMs).

JVM	Directory
32–bit IBM Technology for Java	`/QOpenSys/QIBM/ProdData/JavaVM/jdk60/32bit`
64–bit IBM Technology for Java	`/QOpenSys/QIBM/ProdData/JavaVM/jdk60/64bit`

plugins_profile_root
: The default Web Server Plug-ins profile root is the `/QIBM/UserData/WebSphere/Plugins/V8/webserver/profiles/`*profile_name* directory.

plugins_root
: The default installation root directory for Web Server Plug-ins is the `/QIBM/ProdData/WebSphere/Plugins/V8/webserver` directory.

plugins_user_data_root
: The default Web Server Plug-ins user data root is the `/QIBM/UserData/WebSphere/Plugins/V8/webserver` directory.

product_library
product_lib
: This is the product library for the installed product. The product library for each Version 8.0 installation on the system contains the program and service program objects (similar to `.exe`, `.dll`, `.so` objects) for the installed product. The product library name is `QWAS8`*`x`* (where *x* is A, B, C, and so on). The product library for the first WebSphere Application Server Version 8.0 product installed on the system is `QWAS8A`. The *`app_server_root`*`/properties/product.properties` file contains the value for the product library of the installation, `was.install.library`, and is located under the *`app_server_root`* directory.

profile_root
: The default directory for a profile named *profile_name* for WebSphere Application Server - Express is the `/QIBM/UserData/WebSphere/AppServer/V8/Express/profiles/`*`profile_name`* directory.

shared_product_library
: The shared product library, which contains all of the objects shared by all installations on the system, is `QWAS8`. This library contains objects such as the product definition, the subsystem description, the job description, and the job queue.

user_data_root
: The default user data directory for WebSphere Application Server - Express is the `/QIBM/UserData/WebSphere/AppServer/V8/Express` directory.

: The `profiles` and `profileRegistry` subdirectories are created under this directory when you install the product.

web_server_root
: The default web server path is /www/*web_server_name*.

Administering web server plug-ins

Web server plug-in properties

Use this page to view or change the settings of a web server plug-in configuration file. The plug-in configuration file, `plugin_cfg.xml`, provides properties for establishing communication between the Web server and the Application Server.

To view this administrative console page, click **Servers > Server Types > Web servers >** *web_server_name* > **Plug-in Properties**.

On the **Configuration** tab, you can edit fields. On the **Runtime** tab, you can look at read-only information.

The **Runtime** tab is available only when this web server has accessed applications running on application servers and there is an `http_plugin.log` file.

Note: This topic references one or more of the application server log files. Beginning in WebSphere Application Server Version 8.0 you can configure the server to use the High Performance Extensible Logging (HPEL) log and trace infrastructure instead of using `SystemOut.log` , `SystemErr.log`, `trace.log`, and `activity.log` files or native z/OS logging facilities. If you are using HPEL, you can access all of your log and trace information using the LogViewer command-line tool from your server profile bin directory. See the information about using HPEL to troubleshoot applications for more information on using HPEL.

Ignore DNS failures during web server startup

Specifies whether the plug-in ignores DNS failures within a configuration when starting.

This field corresponds to the IgnoreDNSFailures element in the plugin-cfg.xml file.

When you set the value to **true**, the plug-in ignores DNS failures within a configuration and starts successfully if at least one server in each ServerCluster is able to resolve the host name. If a server host name cannot be resolved, it is marked **unavailable** for the continued existence of the configuration. Further attempts to resolve the host name are not made during the routing of requests. If a DNS failure occurs, a message is written to the plug-in log file and the plug-in initialization process continues.

By default, when the value is **false**, DNS failures cause the plug-in to not initialize and requests fail. However, the web server starts.

Data type	String
Default	false

Refresh configuration interval

Specifies the time interval, in seconds, at which the plug-in should check the configuration file to see if updates or changes have occurred. The plug-in checks the file for any modifications that have occurred since the last time the plug-in configuration was loaded.

In a development environment in which changes are frequent, a lower setting than the default setting of 60 seconds is preferable. In production, a higher value than the default is preferable because updates to the configuration will not occur so often. If the plug-in reload fails for some reason, a message is written to the plug-in log file and the previous configuration is used until the plug-in configuration file successfully reloads. If you are not seeing the changes you made to your plug-in configuration, check the plug-in log file for indications of the problem.

Data type	Integer
Default	60 seconds.

Plug-in configuration file name

Specifies the file name of the configuration file for the plug-in. The Application Server generates the `plugin-cfg.xml` file by default. The configuration file identifies applications, Application Servers, clusters, and HTTP ports for the web server. The web server uses the file to access deployed applications on various Application Servers.

Note: This field is disabled and cannot be changed, but the value is displayed for information purposes only.

If a different location is desired, you need to rerun the Plugin installer to define the new location, and then run the new **configureWebserver** script that is produced from the install process on your WebSphere Application Server machine.

If you select a web server plug-in during installation, the installer program configures the web server to identify the location of the `plugin-cfg.xml` file, if possible. The plug-in configuration file, by default, is installed in the *plugins_root*/`config`/*`web_server_name`* directory.

The installer program adds a directive to the web server configuration that specifies the location of the `plugin-cfg.xml` file.

For remote web servers, you must copy the file from the local directory where the Application Server is installed to the remote machine. This is known as propagating the plug-in configuration file. If you are using IBM HTTP Server V6.1 or higher for your web server, WebSphere Application Server can automatically propagate the plug-in configuration file for you to remote machines provided there is a working HTTP transport mechanism to propagate the file.

You can click **View** to display a copy of the current plug-in configuration file.

Data type	String
Default	`plugin-cfg.xml`

Automatically generate plug-in configuration file

To automatically generate a plug-in configuration file to a remote web server:

- This field must be checked.
- The plug-in configuration service must be enabled

When the plug-in configuration service is enabled, a plug-in configuration file is automatically generated for a web server whenever:

- The WebSphere Application Server administrator defines new web server.
- An application is deployed to an Application Server.
- An application is uninstalled.
- A virtual host definition is updated and saved.

By default, this field is checked. Clear the check box if you want to manually generate a plug-in configuration file for this web server.

Automatically propagate plug-in configuration file

Specifies whether or not you want the application server to automatically propagate a copy of a changed plug-in configuration file to a Web server:

- This field must be checked.
- The plug-in configuration service must be enabled

By default, this field is checked.

Note: The plug-in configuration file can only be automatically propagated to a remote web server if that web server is an IBM HTTP Server V6.1 or higher web server and its administration server is running.

Because the plug-in configuration service runs in the background and is not tied to the administrative console, the administrative console cannot show the results of the automatic propagation.

Plug-in key store file name

Specifies the fully qualified directory path and file name of the database file containing your security key rings that the Web server plug-in uses for HTTPS requests. This file resides on the Web server that is associated with this web server plug-in. After you specify the fully qualified directory path and file name of the database file, you can:

- Cick **Manage keys and certificates** to update this file.
- Click **Copy to web server key store directory** to add a copy of this file to the key store directory for the web server.

Data type	String
Default	None

Plug-in configuration directory and file name

Specifies the fully qualified path of the web server copy of the web server plug-in configuration file. This path is the name of the file and its location on the machine where the web server is running.

Note: This field is disabled and cannot be changed, but the value is displayed for information purposes only.

Plug-in key store directory and file name

Specifies the fully qualified path of the web server copy of the database file that contains your security key rings. This path is the name of the file and its location on the machine where the web server is running.

Note: This field is disabled and cannot be changed, but the value is displayed for information purposes only.

Plug-in logging

Specifies the location and name of the `http_plugin.log` file. Also specifies the scope of messages in the log.

The log describes the location and level of log messages that are written by the plug-in. If a log is not specified within the configuration file, then, in some cases, log messages are written to the web server error log.

On a distributed platform, if the log file does not exist then it will be created. If the log file already exists, it will be opened in append mode and the previous plug-in log messages will remain.

Log file name - The fully qualified path to the log file to which the plug-in will write error messages.

Data type	String
Default	*plugins_root*/`logs`/*web_server_name*/`http_plugin.log` Specify the file path of the `http_plugin.log` file.

Log level- The level of detail of the log messages that the plug-in should write to the log. You can specify one of the following values for this attribute:

- Trace. All of the steps in the request process are logged in detail.

- Stats. The server selected for each request and other load balancing information relating to request handling is logged.
- Warn. All warning and error messages resulting from abnormal request processing are logged.
- Error. Only error messages resulting from abnormal request processing are logged.
- Debug. All of the critical steps performed in processing requests are logged.
- Detail. All of the information about requests and responses are logged.

If a Log level is not specified, the default value **Error** is used.

Be careful when setting the level to **Trace**. A lot of messages are logged at this level which can cause the disk space/file system to fill up very quickly. A **Trace** setting should never be used in a normally functioning environment as it adversely affects performance.

Important: If the web server and web server plug-in are running on an AIX, HP-UX, Linux, or Solaris system, and you change the log level, in the plugin-cfg.xml file, this change is not picked up dynamically. You must restart the web server to pick up the change. For example on Solaris, if you do not restart the web server, the following error message appears in the *Plugin_Home*/logs/http_plugin.log file:

```
ERROR: ws_config_parser:handleLogEnd: Failed to open log file
'/opt/IBM/WebSphere/Plugin/logs/sunwebserver/http_plugin.log', OS
```

Data type	String
Default	Error

Web server plug-in request and response optimization properties

Use this page to view or change the request and response optimization properties for a web server plug-in.

To view this administrative console page, click **Servers > Server Types > Web servers >** *web_server_name* **> Plug-in properties > Request and response**.

Maximum chunk size used when reading the HTTP response body:

Specifies the maximum chunk size the plug-in can use when reading the response body.

This field corresponds to the ResponseChunkSize element in the plugin-cfg.xml file.

The plug-in reads the response body in 64K chunks until all of the response data is read. This approach causes a performance problem for requests whose response body contains large amounts of data.

If the content length of the response body is unknown, the values specified for this property is used as the size of the buffer that is allocated. The response body is then read in this size chunks, until the entire body is read. If the content length is known, then a buffer size of either the content length or the specified size (whichever is less) is used to read the response body.

Data type	Integer
Default	64 kilobytes Specify the size in kilobytes (1024 byte blocks).

Enable Nagle algorithm for connections to the Application Server:

When checked, the Nagle algorithm is enabled for connections between the plug-in and the Application Server.

This field corresponds to the ASDisableNagle element in the plugin-cfg.xml file.

The Nagle algorithm is named after engineer John Nagle, who invented this standard part of the transmission control protocol/internet protocol (TCP/IP). The algorithm reduces network overhead by adding a transmission delay (usually 200 milliseconds) to a small packet, which lets other small packets arrive and be included in the transmission. Because communications has an associated cost that is not as dependent on packet size as it is on frequency of transmission, this algorithm potentially reduces overhead with a more efficient number of transmissions.

By default, this field is not checked, and the Nagle algorithm is disabled. Select this field to enable the Nagle algorithm.

Enable Nagle Algorithm for the IIS Web Server:

When checked, the Nagle algorithm is used for connections from the Microsoft Internet Informations Services (IIS) Web Server to the application server.

This field corresponds to the IISDisableNagle element in the plugin-cfg.xml file. It only applies if you are using the Microsoft Internet Informations Services (IIS) Web Server.

By default, this field is not checked, and the Nagle algorithm is disabled. Select this field to enable the Nagle algorithm for this connection.

Chunk HTTP response to the client:

When checked, responses to the client are broken into chunks if a `Transfer-Encoding : Chunked` response header is present in the response.

This field corresponds to the ChunkedResponse element in the plugin-cfg.xml file. It only applies if you are using a Microsoft Internet Informations Services (IIS) Web Server, a Java System web server, or a Domino Web Server. The IBM HTTP Server automatically handles breaking the response into chunks to send to the client.

By default, this field is not checked, and responses are not broken into chunks. Select this field to enable responses to the client to be broken into chunks if a `Transfer-Encoding : Chunked` response header is present in the response.

Accept content for all requests: This field corresponds to the AcceptAllContent element in the plugin-cfg.xml file.

When selected, you can include content in GET, and HEAD requests when a Content-Length or Transfer-encoding header is contained in the request header.

By default, this field is checked.

Select this field to enable users to include content in GET, and HEAD requests when a Content-Length or Transfer-encoding header is contained in the request header.

Note: The **Accept content for all requests** setting on the administrative console corresponds to the AcceptAllContent attribute in the plugin-cfg.xml file. For Version 8, the default for the setting is checked and for the attribute is true. Before Version 8, the default for the setting is not checked and for the attribute is false.

Virtual host matching:

When selected, virtual host mapping is performed by physically using the port number for which the request was received.

This field corresponds to the VHostMatchingCompat element in the plugin-cfg.xml file.

By default, this field is not checked, and matching is done logically using the port number contained in the host header. Select this field if you want virtual host mapping performed by physically using the port number for which the request was received.

Use the radio buttons to make your physical or logical port selection.

Application server port preference:

Specifies which port number the Application Server should use to build URIs for a sendRedirect. This field is only applicable for a sendRedirect if you use relative URIs and does not affect absolute redirects. This field also specifies where to retrieve the value for HttpServletRequest.getServerPort().

This field corresponds to the AppServerPortPreference element in the plugin-cfg.xml file.

Specify:
- `hostHeader` if the port number from the host header of the HTTP request coming in is to be used.
- `webserverPort` if the port number on which the web server received the request is to be used.

The default is `hostHeader`.

Web server plug-in caching properties

Use this page to view or change the caching properties for a web server plug-in.

To view this administrative console page, click **Servers > Server Types > Web servers >** *web_server_name* > **Plug-in properties > Caching**.

Enable Edge Side Include (ESI) processing to cache the responses:

Specifies whether to enable Edge Side Include processing to cache the responses.

This field corresponds to the esiEnable element in the plugin-cfg.xml file.

By default, this field is checked. Select this field if you want Edge Side Include (ESI) processing used to cache responses. If ESI processing is disabled for the plug-in, the other ESI plug-in properties are ignored. Clear the checkbox to disable Edge Side Include processing.

Enable invalidation monitor to receive notifications:

When checked, the ESI processor receives invalidations from the application server.

This field corresponds to the ESIInvalidationMonitor element in the plugin-cfg.xml file. It is ignored if Edge Side Include (ESI) processing is not enabled for the plug-in.

By default, this field is not selected. Clear the check box if you do not want the application server to send invalidations to the ESI processor.

Maximum cache size:

Specifies, in 1K byte units, the maximum size of the cache. The default maximum size of the cache is 1024K bytes (1 megabyte). If the cache is full, the first entry to be evicted from the cache is the entry that is closest its expiration time.

This field corresponds to the esiMaxCacheSize element in the plugin-cfg.xml file.

Data type	Integer
Default	1024 kilobytes Specify the size in kilobytes (1024 byte blocks).

Web server plug-in request routing properties

Use this page to view or change the request routing properties for a web server plug-in.

To view this administrative console page, click **Servers > Server Types > Web servers >** *web_server_name* **> Plug-in properties > Request routing**.

Load balancing option:

Specifies the load balancing option that the plug-in uses in sending requests to the various application servers associated with that web server.

This field corresponds to the LoadBalance element in the plugin-cfg.xml file.

Select the appropriate load balancing option:

- Round robin
- Random

The Round Robin implementation has a random starting point. The first application server is picked randomly. Round Robin is then used to pick application servers from that point forward. This implementation ensures that in multiple process based web servers, all of the processes don't start up by sending the first request to the same Application Server.

The default load balancing type is Round Robin.

Retry interval:

Specifies the length of time, in seconds, that should elapse from the time an application server is marked down to the time that the plug-in retries a connection.

This field corresponds to the RetryInterval element in the plugin-cfg.xml file.

Data type	Integer
Default	60 seconds

Maximum size of request content:

Select whether there is a limit on the size of request content. If limited, this field also specifies the maximum number of kilobytes of data a request can contain. When a limit is set, the plug-in fails any request that is received that contains more data than the specified limit.

This field corresponds to the PostSizeLimit element in the plugin-cfg.xml file.

Select whether to limit the size of request content:

- `No limit`
- `Set limit`

If you select `Set limit`, specify a limit size.

Data type	Integer Specify the size in kilobytes (1024 byte blocks).
Default	`-1`, which indicates there is no limit for the post size.

Maximum buffer size used when reading HTTP request content:

Specifies, in kilobytes, the maximum buffer size that is used when the content of an HTTP request is read. If the application server that initially receives a request cannot process that request, the data contained in this buffer is sent to another application server in an attempt to have that application server process the request.

This field corresponds to the PostBufferSize element in the plugin-cfg.xml file.

If **Set limit** is selected, specify a limit size.

Data type	Integer Specify the size in kilobytes (1024 byte blocks).
Default	`64`

Remove special headers:

When checked, the plug-in will remove any headers from incoming requests before adding the headers the plug-in is supposed to add before forwarding the request to an application server.

This field corresponds to the RemoveSpecialHeaders element in the plugin-cfg.xml file.

The plug-in adds special headers to the request before it is forwarded to the application server. These headers store information about the request that will need to be used by the application. Not removing the headers from incoming requests introduces a potential security exposure.

By default, the special headers are not retained. Clear the check box to retain special headers.

Clone separator change:

When this option is selected, the plug-in expects the plus character (+) as the clone separator.

This field corresponds to the ServerCloneID element in the plugin-cfg.xml file.

Some pervasive devices cannot handle the colon character (:) used to separate clone IDs in conjunction with session affinity. If this field is checked, you must also change the configurations of the associated application servers such that the application servers separates clone IDs with the plus character as well.

By default, this option is selected. Clear the field if you want to use the colon character to separate clone IDs.

Web server plug-in configuration service property

Use this page to view or change the configuration settings for the web server plug-in configuration service.

Note: This topic references one or more of the application server log files. Beginning in WebSphere Application Server Version 8.0 you can configure the server to use the High Performance Extensible Logging (HPEL) log and trace infrastructure instead of using `SystemOut.log` , `SystemErr.log`, `trace.log`, and `activity.log` files or native z/OS logging facilities. If you are using

HPEL, you can access all of your log and trace information using the LogViewer command-line tool from your server profile bin directory. See the information about using HPEL to troubleshoot applications for more information on using HPEL.

If you are using a stand-alone application server, click **Servers > Server Types > WebSphere application servers >** *server_name* **> Administration Services > Web server plug-in configuration service** to view this administrative console page.

Enable automated web server configuration processing

The web server plug-in configuration service is selected by default. The service automatically generates the plug-in configuration file whenever the web server environment changes, with a few exceptions. For example, the plug-in configuration file is regenerated whenever one of the following activities occurs:

- A new application is deployed on an associated application server
- The web server definition is saved
- An application is removed from an associated application server
- A new virtual host is defined

In a base WebSphere Application Server or WebSphere Application Server, Express configuration, the plug-in configuration file does not regenerate when TCP channel settings are updated for an application server.

In a base WebSphere Application Server or WebSphere Application Server, Express configuration, whenever a virtual host definition is updated, the plug-in configuration file is automatically regenerated for all of the web servers.

By default, this option is selected. Clear the field to disable automated web server configuration processing.

Application Server property settings for a web server plug-in

Use this page to view or change application server settings for a web server plug-in.

To view this administrative console page, click **Servers > Server Types > WebSphere application servers >** *server_name*, and then, in the Additional Properties section, click **Web server plug-in properties**.

Server Role

Specifies the role this application server is assigned.

Select `Primary` to add this application server to the list of primary application servers. The plug-in initially attempts to route requests to the application servers on this list.

Select `Backup` to add this application server to the list of backup application servers. The plug-in does not load balance across the backup application servers. A backup server is only used if a primary server is not available. When the plug-in determines that a backup application server is required, it goes through the list of backup servers, in order, until no servers are left in the list or until a request is successfully sent and a response received from one of the servers on this list.

Default setting	Primary

Connection timeout

Specifies whether there is a limited amount of time during which the application server maintains a connection with the web server.

This field corresponds to the ConnectTimeout element in the plugin-cfg.xml file.

The setting for this field determines whether the plug-in performs non-blocking connections with the application server. Non-blocking connections are beneficial when the plug-in is unable to contact the destination to determine whether or not the port is available.

If the **Use connection timeout** setting is not selected, the plug-in performs nonblocking connections with the application server. If the **Use connection timeout** setting is selected, you must specify a value in the **seconds** field:

- If you specify a value greater than 0 in the **seconds** field, the plug-in waits the specified number for seconds to perform a successful connection. If a connection does not occur during that time interval, the plug-in marks the server unavailable, and sends the request to another application server in the cluster.
- If you specify a value of 0 in the **seconds** field, the plug-in performs a blocking connection.
- If you do not specify a value in the **seconds** field, the plug-in performs a blocking connection where the plug-in sits until the operating system times out, whhich might be as long as two minutes, depending on the platform, before it marks the server `unavailable`.

Data type	Integer
Default	5

Use read/write timeout

Specifies whether there is a time limit for how long the plug-in waits to send a request to or receive a response from the application server.

This field corresponds to the ServerIOTimeout element in the plugin-cfg.xml file.

Select the **Use read/write timeout** property to set a read/write timeout. When you select this setting, you must specify the length of time, in seconds that the plug-in waits to send a request or to receive a response. When selecting a value to specify for this field, remember that it might take a couple of minutes for an application server to process a request. Setting the value too low might cause the plug-in to send a false server error response to the client. If the checkbox is not checked, the plug-in uses blocked I/O to write requests to and read responses from the application server until the TCP connection times out.

Note: The **Use read/write timeout** setting on the administrative console corresponds to the ServerIOTimeout attribute in the plugin-cfg.xml file. The default value for this setting is different from the default value in previous versions of the product.

Data type	Integer
Default	900 seconds

Use maximum number of connections

Specifies the maximum number of pending connections to an Application Server that can be flowing through a web server process at any point in time.

This field corresponds to the ServerMaxConnections element in the plugin-cfg.xml file.

Select the **Use maximum number of connections** property to set a maximum number of connections. When you select this setting, you, must specify the maximum number of connections that can exist between the web server and the Application Server at any given point in time.

If this property is set to either zero or -1, there is no limit to the number of pending connections to the Application Servers.

Data type	Integer

Default	0

Use extended handshake to check whether application server is running

When selected, the web server plug-in uses an extended handshake to check whether the application server is running.

This field corresponds to the ExtendedHandshake element in the plugin-cfg.xml file.

Select this property if a proxy firewall is between the plug-in and the application server.

The plug-in marks a server as down when the connect() fails. However, when a proxy firewall is in between the plug-in and the application server, the connect() will succeed, even though the back end application server is down. This causes the plug-in to not failover correctly to another application server.

If the plug-in performs some handshaking with the application server to ensure that the application server is started before the plug-in sends a request, the plug-in can failover to another application server if it detects that the application server with which it is attempting to perform a handshake is not available.

By default, this field is not selected. Select this field if you want to use extended handshake to check whether an application server is running.

Send the header "100 Continue" before sending the request content

Specifies whether the web server plug-in sends the header "100 Continue" to the application server before it sends the request content.

This field corresponds to the WaitForContinue element in the plugin-cfg.xml file.

When selected, the web server plug-in sends the header "100 Continue" to the application server before it sends the request content.

By default, this field is not selected. Select this field to enable this function.

plugin-cfg.xml file

There are two types of `plugin-cfg.xml` files: application-centric and topology-centric.

An application-centric file has an application that is mapped to both web server and application server definitions. Changes that you make to the plug-in by using the administrative console persist to the `plugin-cfg.xml` file upon generation.

A topology-centric file represents everything that is defined in the environment. Typically, this `plugin-cfg.xml` file is used when you do not have web servers defined. Consider the following rules when you want to update a topology-centric `plugin-cfg.xml` file:

- If the `plugin-cfg.xml` file *exists* within the *app_server_root*`\dmgr\cells` directory, the plug-in generation process ignores the updated values from the **application server** > **Web Server Plugin Properties** panel of the administrative console and uses the existing values within the XML file. In this case, you must manually update the XML file for those values to persist.
- If the `plugin-cfg.xml` file *does not exist* within the *app_server_root*`\dmgr\cells` directory, the plug-in generation process creates a new `plugin-cfg.xml` file. The process persists the latest values that are set on the **Application Server** > **Web Server Plugin Properties** panel within the administrative console.

The design of this file and its related function enable the product to support different types of configurations. For example, web server definitions might not exist. In addition, many web server plug-in properties, such as RefreshInterval, LogLevel, and the Edge Side Include (ESI) processor properties, can be updated manually only. Those values must be maintained through each iteration.

The `plugin-cfg.xml` file includes the following elements and attributes. Unless indicated otherwise, you can specify each element and attribute only once within the `plugin-cfg.xml` file.

gotcha: Use the administrative console to set these properties for each web server definition. Any manual changes you make to the plug-in configuration file for each web server are overridden whenever the file is regenerated.

Note: You can update the *global* `plugin-cfg.xml` file using the administrative console or running the GenPluginCfg command for all of the clusters in a cell. However, you must delete the `config/cells/plugin-cfg.xml` file before you update the *global* `plugin-cfg.xml` file. If you do not delete the `config/cells/plugin-cfg.xml` file, only the new properties and their values are added to the *global* `plugin-cfg.xml` file. Any updates to existing plug-in property values are not added to the *global* `plugin-cfg.xml` file.

The file can include one or more of the following elements and attributes. The `Config` element is required.

- "Config"
- "IgnoreDNSFailures" on page 62
- "RefreshInterval" on page 62
- "ASDisableNagle" on page 62
- "IISDisableNagle" on page 62
- "VHostMatchingCompat" on page 62
- "AppServerPortPreference" on page 62
- "ResponseChunkSize" on page 63
- "AcceptAllContent" on page 63
- "ChunkedResponse" on page 63
- "IISPluginPriority" on page 63
- "TrustedProxyEnable" on page 64
- "TrustedProxyList" on page 64
- "SSLConsolidate" on page 64
- "plugin-cfg.xml file" on page 60
- "plugin-cfg.xml file" on page 60
- "Log" on page 64
- "Property Name="esiEnable" Value="true/false"" on page 65
- "Property Name="esiMaxCacheSize" Value="integer"" on page 65
- "Property Name="ESIInvalidationMonitor" Value="true/false" " on page 65
- "Property Name="FIPSEnable" Value="true/false" " on page 65
- "Property Name="PluginInstallRoot" Value="C:\IBM\WebSphere\Plugins" " on page 65
- "ServerCluster" on page 66
- "VirtualHostGroup" on page 71
- "UriGroup" on page 72
- "Route" on page 72
- "RequestMetrics" on page 73

Config

This element, which is required, starts the HTTP plug-in configuration file.

IgnoreDNSFailures

Specifies whether the plug-in ignores DNS failures within a configuration when starting. When set to `true`, the plug-in ignores DNS failures within a configuration and starts successfully if at least one server in each server cluster is able to resolve the host name. Any server for which the host name cannot be resolved is marked *unavailable* for the life of the configuration. No attempts to resolve the host name are made during the routing of requests. If a DNS failure occurs, a log message is written to the plug-in log file, and the plug-in initialization continues rather than causing the web server not to start. The default value is `false`, meaning DNS failures cause the web server not to start.

RefreshInterval

Specifies the time interval, in seconds, at which the plug-in checks the configuration file to see if updates or changes have occurred. The plug-in checks the file for any modifications that have occurred since the last time the plug-in configuration was loaded.

In a development environment in which changes are frequent, a lower setting than the default setting of 60 is preferable. In production, a higher value than the default is preferable because updates to the configuration do not occur so often. If the plug-in reload fails for some reason, a message is written to the plug-in log file and the previous configuration is used until the plug-in configuration file successfully reloads. If you are not seeing the changes you made to your plug-in configuration, check the plug-in log file for indications of the problem.

ASDisableNagle

Specifies whether the user wants to disable the nagle algorithm for the connection between the plug-in and the application server. By default, the nagle algorithm is enabled.

The value can be `true` or `false`.

IISDisableNagle

Specifies whether the user wants to disable the nagle algorithm on Microsoft Internet Information Services (IIS). By default, the nagle algorithm is enabled.

The value can be `true` or `false`.

VHostMatchingCompat

Specifies that the port number is to be used for virtual host matching. Specify one of the following values:
- `true` if matching is to be done physically by using the port number for which the request was received.
- `false` if matching is to be done logically by using the port number contained in the host header.

The default value is `false`.

AppServerPortPreference

Specifies which port number the application server uses to build URIs for a sendRedirect() method. The following values can be specified:
- `hostHeader` if the port number from the host header of the HTTP request coming in is to be used.
- `webserverPort` if the port number on which the web server received the request is to be used.

The default value is `hostHeader`.

ResponseChunkSize

Specifies the maximum chunk size to use when reading the response body. For example, specify `Config ResponseChunkSize="N">`, where *N* equals the chunk size in kilobytes.

The plug-in reads the response body in 64 K chunks until all of the response data is read. This approach causes a performance problem for requests whose response body contains large amounts of data.

If the content length of the response body is unknown, a buffer size of *N* KBs is allocated and the body is read in *N* KB size chunks, until the entire body is read. If the content length is known, then a buffer size of either content length or *N* (whichever is less) is used to read the response body.

The default chunk size is 64 K.

AcceptAllContent

Specifies whether users can include content in POST, PUT, GET, and HEAD requests when a Content-Length or Transfer-encoding header is contained in the request header. You can specify one of the following values for this attribute:
- `True` if content is expected and read for all requests
- `False` if content is only expected and read for POST and PUT requests.

The default value is `True`.

ChunkedResponse

Specifies whether the plug-in must use chunks the response to the client when a Transfer-Encoding: Chunked response header is present in the response.

This attribute applies to the IIS, Oracle iPlanet, and Lotus Domino web servers only. The IBM HTTP Server automatically handles the chunking of the response to the client.

You can specify one of the following values for this attribute:
- `true` if the plug-in is to chunk the response to the client when a Transfer-Encoding: Chunked response header is present in the response.
- `false` if the response is not to be chunked.

The default value is `false`.

IISPluginPriority

Specifies the priority in which the IIS web server loads the WebSphere web server plug-in. Specify one of the following values for this attribute:
- `High`
- `Medium`
- `Low`

The default value is `High`.

Note: The IIS web server uses this value during startup. Therefore, you must restart the web server before this change takes effect.

Note: The default value of High ensures that all requests are handled by the WebSphere web server plug-in before they are handled by any other filter or extensions. If problems occur when using a priority of Medium or Low, you must rearrange the order or change the priority of the interfering filter or extension.

TrustedProxyEnable

Permits the web server plug-in to interface with the proxy servers and load balancers that are listed for the TrustedProxyList custom property. When this property is set to `true`, the proxy servers and load balancers in this trusted proxy list can set values for the $WSRA and $WSRH internal headers. The $WSRA internal header is the IP address of the remote host, which is typically the browser, or an internal address that is obtained by Network Address Translation (N.A.T.). The $WSRH internal header is the host name of the remote host. This header information enables the web server plug-in to interface with that specific proxy server or load balancer.

When you use this custom property you must also use the TrustedProxyList custom property to specify a list of trusted proxy servers and load balancers. Also, you must clear the Remove special headers check box on the Request Routing panel within the administrative console. For more information, see the documentation on web server plug-in request routing properties.

TrustedProxyList

Specifies a comma delimited list of all proxy servers or load balancers that have permission to interface with this web server plug-in. You must use this property with the `TrustedProxyEnable=true` custom property setting. If the TrustedProxyEnable custom property is set to `false`, this list is ignored.

SSLConsolidate

Specifies whether the web server plug-in is to compare the setup of each new SSL transport with the setup of other SSL transports that are already defined in the configuration file. When you set this property to `true`, and the plug-in determines that the keyring and CertLabel values specified for the new SSL transport match the values specified for an already defined SSL transport, the plug-in uses the existing SSL environment instead of creating a new SSL environment. Creating fewer SSL environments means that the plug-in requires less memory, and the plug-in initialization time decreases, thereby optimizing your overall GSkit environment.

Log

Describes the location and level of log messages that are written by the plug-in. If a log element is not specified within the configuration file, then, in some cases, log messages are written to the web server error log.

For example, you might specify the following line of code:

```
<Log LogLevel="Error" Name="/opt/WebSphere/AppServer60/logs/http_plugin.log"/>
```

- **Name**

  Specifies the fully qualified path to the log file to which the plug-in writes error messages. Specify exactly one attribute for each log.

  If the file does not exist, then one is created. If the file exists, then it is opened in append mode, and the previous plug-in log messages remain.

- **LogLevel**

  Specifies the level of detail of the log messages that the plug-in writes to the log. Specify zero or one of the following values for each log.

Log Level Value	Log Level Description
Trace	All of the steps in the request process are logged in detail.
Stats	The server selected for each request and other load balancing information relating to request handling is logged.

Log Level Value	Log Level Description
Warn	All warning and error messages resulting from abnormal request processing are logged
Error	Only error messages resulting from abnormal request processing are logged
Debug	All of the critical steps performed in processing requests are logged.
Detail	All of the information about requests and responses are logged.

If a LogLevel value is not specified for the Log element, the default value, Error, is used.

Note: Be careful when setting the level to Trace. Multiple messages are logged at this level, which can consume disk space quickly. Do not use a Trace setting in a normally functioning environment because it adversely affects performance.

Property Name="esiEnable" Value="*true/false*"

Enables or disables the Edge Side Include (ESI) processor. If the ESI processor is disabled, the other ESI elements in this file are ignored.

You can set `Value` to `true` or `false`. By default, the ESI processor is enabled with its value set to `true`.

Property Name="esiMaxCacheSize" Value="*integer*"

Specifies, in 1 KB units, the maximum size of the cache. The default maximum size of the cache is 1024 KB (1 MB). If the cache is full, the first entry deleted from the cache is the entry that is closest its expiration time.

Property Name="ESIInvalidationMonitor" Value="*true/false*"

Specifies whether the ESI processor receives invalidations from the application server.

You can set `Value` to `true` or `false`. By default, this property is set to `false`.

Property Name="FIPSEnable" Value="*true/false*"

Specifies whether the Federal Information Processing Standard (FIPS) is enabled for making Secure Sockets Layer (SSL) connections to the application server. Set this property to `true`, if FIPS is enabled on the application server.

You can set `Value` value to `true` or `false`. By default, this property is set to `false`.

Property Name="PluginInstallRoot" Value="*C:\IBM\WebSphere\Plugins*"

Specifies the installation path for the plug-in. This property is mandatory if using the Global Security Kit (GSKit) because WebSphere Application Server supports the local installation of the GSKit instead of a global installation. The attribute value is set to a fully qualified path to the plug-in installation root.

Supported names recognized by the transport are keyring, stashfile, and password. By default, this property is set to `none`.

ServerCluster

Specifies a group of servers that are generally configured to service the same types of requests. Specify one or more clusters for each configuration.

In the simplest case, the cluster contains only one server definition. In the case in which more than one server is defined, the plug-in load balances across the defined servers by using either a Round Robin or a Random algorithm. The default algorithm is Round Robin.

The following code is an example of a ServerCluster element.

```
<ServerCluster Name="Servers">
<ClusterAddress Name="ClusterAddr">
<Transport Hostname="192.168.1.2" Port="9080" Protocol="HTTP"/>
<Transport Hostname="192.168.1.2" Port="9443" Protocol="HTTPS">
<Property Name="Keyring" value="c:/WebSphere/AppServer/keys/keyring.kdb"/>
<Property Name="Stashfile" value="c:/WebSphere/AppServer/keys/keyring.sth"/>
</Transport>
</ClusterAddress>
<Server Name="Server1">
<Transport Hostname="192.168.1.3" Port="9080" Protocol="HTTP"/>
<Transport Hostname="192.168.1.3" Port="9443" Protocol="HTTPS">
<Property Name="Keyring" value="c:/WebSphere/AppServer/keys/keyring.kdb"/>
<Property Name="Stashfile" value="c:/WebSphere/AppServer/keys/keyring.sth"/>
</Transport>
</Server>
<Server Name="Server2">
<Transport Hostname="192.168.1.4" Port="9080" Protocol="HTTP"/>
<Transport Hostname="192.168.1.4" Port="9443" Protocol="HTTPS">
<Property Name="Keyring" value="c:/WebSphere/AppServer/keys/keyring.kdb"/>
<Property Name="Stashfile" value="c:/WebSphere/AppServer/keys/keyring.sth"/>
</Transport>
</Server>
<Server Name="Server3">
<Transport Hostname="192.168.1.5" Port="9080" Protocol="HTTP"/>
<Transport Hostname="192.168.1.5" Port="9443" Protocol="HTTPS">
<Property Name="Keyring" value="c:/WebSphere/AppServer/keys/keyring.kdb"/>
<Property Name="Stashfile" value="c:/WebSphere/AppServer/keys/keyring.sth"/>
</Transport>
</Server>
<PrimaryServers>
<Server Name="Server1"/>
<Server Name="Server2"/>
</PrimaryServers>
<BackupServers>
<Server Name="Server3"/>
</BackupServers>
</ServerCluster>
```

The z/OS PTF UK35083 package includes the SSL interface change for the z/OS HTTP Server, Version 5.3, that corresponds to this web server plug-in change. Therefore, you must apply this PTF to your system before the new web server plug-in SSL interface can function properly.

| You must also include the `SSLMODE=MULTI` option in the `httpd.conf` file for the IBM HTTP Server for z/OS,
| Version 5.3. The SSLMODE=ON option is not supported in Version 7.0 or higher.

If the `SSLMode multi` option is not specified in the `httpd.conf` file, or if you do not have the z/OS PTF UK35083 package applied to your system, you might receive error message IMW0584W. This message indicates that the SSL mode, which is specified for the HTTP Server, is not compatible with the SSL mode for the web server plug-in that is used with the IBM HTTP Server for z/OS, Version 5.3. In either of these situations, unpredictable results might occur.

For the web server plug-ins for both the IBM HTTP Server for z/OS, Version 5.3 and the IBM HTTP Server on z/OS powered by Apache:

- If you use a `kdb` file with a stashfile in the hierarchical file system (HFS), specify both the `Property Name=keyring` and the `Property Name=stashfile` elements, as shown in the preceding example.

  | **gotcha:** The format of the values you specify for these elements is different from what you specified in
  | earlier versions of the product.

- If you use a System Authorization Facility (SAF) keyring, instead of a `kdb` file, you must create the following two custom plug-in properties from the administrative console:

  KeyringLocation
  Specify the directory location of the SAF keyring as the value for this property. When you save this configuration change, this directory location becomes the value of the keyring property in the `plugin-cfg.xml` file.

  StashfileLocation
  Specify "" (null) as the value for this property. When you save this configuration change, "" (null) becomes the value of the stashfile property in the `plugin-cfg.xml` file

  See "Web server plug-in configuration properties" on page 76 for instructions on how to create `KeyringLocation` and `StashfileLocation` from the administrative console.

  Use the following example for the SAF keyring:

```
<Transport Hostname="appserver.example.com" Port="9443" Protocol="https">
<Property name="keyring" value="safkeyring:///SAF_keyring_name"/>
<Property Name="stashfile" value=""/>
</Transport>
```

- **Name**

  Specifies the logical or administrative name to be used for this group of servers. Specify one attribute for each ServerCluster.

- **LoadBalance**

  The following values can be specified for this attribute:

  Round Robin

  Random

  The Round Robin implementation has a random starting point. the first application server is picked randomly. Round Robin is then used to pick application servers from that point forward. This implementation ensures that in multiple process based wed servers, all of the processes don't start up by sending the first request to the same Application Server.

  The Random implementation also has a random starting point. However with this implementation all subsequent servers are also randomly selected. Therefore, the same server might get selected repeatedly while other servers remain idle.

  The default load balancing type is Round Robin.

- **IgnoreAffinityRequests**

  Specifies whether the plug-in ignores the number of affinity requests made to a server when selecting servers based on the Round Robin algorithm. Specify zero or one attribute for each ServerCluster. The value is `true` or `false`. If the value is set to `false`, the number of affinity requests made is also taken into account in the server selection process.

  The default value is `true`, which means the number of affinity requests made are not used in the Round Robin algorithm.

- **RetryInterval**

  Specifies an integer value for the length of time that elapses from the time that a server is marked down to the time that the plug-in tries a connection again. The default is 60 seconds. Specify zero or one attribute for each ServerCluster.

- **RemoveSpecialHeaders**

The plug-in adds special headers to the request before it is forwarded to the application server. These headers store information about the request that is used by the application. By default, the plug-in removes these headers from incoming requests before adding the headers it is supposed to add. Specify zero or one attribute for each ServerCluster.

The value can be `true` or `false`. Setting the attribute to false introduces a potential security exposure by not removing headers from incoming requests.

- **CloneSeparatorChange**

  Tells the plug-in to expect the plus character (+) as the clone separator. Some pervasive devices cannot handle the colon character (:) that is used to separate clone IDs in conjunction with session affinity. You must change application server configurations so that an application server separates clone IDs with the plus character as well. Specify zero or one attribute for each ServerCluster.

  The value can be `true` or `false`.

- **PostSizeLimit**

  The maximum number of KBs (1024 byte) blocks of request content allowed for the plug-in to attempt to send the request to an application server. If a request is received that is greater than this size, the plug-in fails the request. The default value is -1 byte, which indicates that there is no limit for the post size. Specify zero or one attribute for each ServerCluster.

- **PostBufferSize**

  Specifies, in KBs, the maximum buffer size that is used when the content of an HTTP request is read. If the application server that initially receives a request cannot process that request, the data contained in this buffer is sent to another application server. It then attempts to have that application server process the request.

  This option improves the availability of the plug-in. Pending requests with content are tried again if the selected application server is not responding. If the value is set to zero, the requests with content are not buffered and are not tried again. The default value is 64. Specify zero or one attribute for each ServerCluster.

- **Server**

  Specifies a WebSphere Application Server instance that is configured to handle requests routed to it, based on the routing rules of the plug-in configuration. The server corresponds to an application server running on either the local machine or a remote machine. Specify zero or one attribute for each ServerCluster.

  – **Name**

    Specifies the administrative or logical name for the server. Specify exactly one attribute for each Server.

  – **WaitForContinue**

    Specifies whether to use the HTTP 1.1 `100 Continue` support before sending the request content to the application server. Possible attribute values are `true` or `false`. The default value is false; the plug-in does not wait for the `100 Continue` response from the application server before sending the request content because it is a performance hit. Specify zero or one attribute for each Server.

    This property is ignored for POST requests to prevent a failure from occurring if the application server closes a connection because of a keep-alive timeout.

    Enable this function true when configuring the plug-in to work with certain types of proxy firewalls.

  – **LoadBalanceWeight**

    Specifies the weight associated with this server when the plug-in performs weighted Round Robin load balancing. Specify zero or one attribute for each Server. The starting value for a server can be any integer between `0` and `20`. However, specify zero only for a server that is not running.

    The LoadBalanceWeight value for each server is decremented for each request that is processed by that server. After the weight for a particular server in a server cluster reaches zero, only requests with session affinity are routed to that server. When all servers in the cluster reach a weight of zero, the weights for all servers in the cluster are reset, and the algorithm restarts.

Note: When a server is not running, set the weight for that server to zero. The plug-in can then reset the weights of the servers that are still running, and maintain proper load balancing.

– **ConnectTimeout**

Enables the plug-in to perform non-blocking connections with the application server. Non-blocking connections are beneficial when the plug-in is unable to contact the destination to determine if the port is available or unavailable. Specify zero or one attribute for each Server.

If a ConnectTimeout value is not specified, the plug-in performs a blocking connect in which the plug-in sits until an operating system times out (as long as 2 minutes depending on the platform) and allows the plug-in to mark the server *unavailable*. A value of `0` causes the plug-in to perform a blocking connect. A value greater than 0 specifies the number of seconds you want the plug-in to wait for a successful connection. If a connection does not occur after that time interval, the plug-in marks the server *unavailable* and fails over to one of the other servers defined in the cluster. The default value is `5` seconds.

– **ExtendedHandshake**

Is used when a proxy firewall is between the plug-in and the application server. In such a case, the plug-in is not failing over, as expected. Specify zero or one attribute for each Server.

The plug-in marks a server as down when the connect() method fails. However, when a proxy firewall is in between the plug-in and the application server, the connect() method succeeds, even though the back-end application server is down. This causes the plug-in to not failover correctly to other application servers.

The plug-in performs some handshaking with the application server to ensure that it is started before sending the request. This scenario enables the plug-in to failover in the event the application server is down.

The value can be `true` or `false`.

– **MaxConnections**

Specifies the maximum number of pending connections to an application server that can be flowing through a web server process at any point in time. Specify one element for each Server.

For example, in the following scenario:
- The application server is fronted by five nodes that are running an IBM HTTP Server.
- Each node starts two processes.
- The MaxConnections attribute is set to 50.

In this example, the application server can potentially get up to 500 connections. Multiply the number of nodes, 5, by the number of processes, 2, and then multiply that number by the number specified for the MaxConnections attribute, 50, for a total of 500 connections.

By default, MaxConnections is set to -1. If this attribute is set to either zero or -1, there is no limit to the number of pending connections to the application servers.

– **Transport**

Specifies the transport for reading and writing requests to a particular WebSphere Application Server instance. The transport provides the information that is necessary to determine the location of the application server to which the request is sent. If the server has multiple transports that are defined to use the same protocol, the first one is used. Specify one or more elements for each Server.

It is possible to configure the server to have one non-secure transport and one that uses SSL. In this configuration, a match of the incoming request protocol is performed to determine the appropriate transport to use to send the request to the application server.

- **Hostname**

  Specifies the host name or IP address of the machine on which the WebSphere Application Server instance is running. There is exactly one attribute for each transport.

- **Port**

  Specifies the port on which the WebSphere Application Server instance is listening. There is exactly one attribute for each transport.

- **Protocol**

Specifies the protocol to use when communicating over this transport -- either HTTP or HTTPS. There is exactly one attribute for each transport.

– **Property**

  Specify zero, one, or more elements for each transport. When the protocol of the transport is set to HTTPS, use this element to supply the various initialization parameters, such as password, keyring and stashfile. For example, the portion of the `plugin-cfg.xml` file containing these elements might look like the following code:

```
<Transport Hostname="192.168.1.2" Port="9443" Protocol="HTTPS">
<Property Name="keyring" value="c:/WebSphere/AppServer/keys/keyring.kdb"/>
<Property Name="stashfile" value="c:/WebSphere/AppServer/keys/keyring.sth"/>
<Property Name="password" value="WebAS"/>
```

  - **Name**

    Specifies the name of the property that is being defined. Supported names recognized by the transport are keyring, stashfile, and password.

    Note: The only name that can be specified for the WebSphere HTTP plug-in for z/OS is password. If you specify keyring and stashfile, they are ignored.

    Specify exactly one attribute for each Property.

  - **Value**

    Specifies the value of the property being defined. Specify exactly one attribute for each property.

– **ServerIOTimeout**

  Enables the plug-in to set a timeout value, in seconds, for sending requests to and reading responses from the application server.

  If you set the ServerIOTimeout attribute to a positive value, this attempt to contact the server ends when the timeout occurs. However, the server is not considered unavailable and future requests are still sent to the server on which the unavailable timeout occurred.

  If you set the ServerIOTimeout attribute to a negative value, the server is considered unavailable whenever a timeout occurs, and no future requests are sent to the server on which the timeout occurred.

  If a value is not set for the ServerIOTimeout attribute, the plug-in, by default, uses blocked I/O to write request to and read response from the application server until the TCP connection times out. For example, you might specify the following setting:

```
<Server Name="server1" ServerIOTimeout=300>
```

  In this situation, if an application server stops responding to requests, the plug-in waits 300 seconds (5 minutes) before timing out the TCP connection. Setting the ServerIOTimeout attribute to a reasonable value enables the plug-in to timeout the connection sooner, and transfer requests to another application server when possible.

  When selecting a value for this attribute, remember that sometimes it might take several minutes for an application server to process a request. Setting the value of the ServerIOTimeout attribute too low might cause the plug-in to send a false server error response to the client.

| The default value is `900`, which is equivalent to 15 minutes.

  Note: The ServerIOTimeout limits the amount of time the plug-in waits for each individual read or write operation to return. ServerIOTimeout does not represent a timeout for the overall request.

  For additional recommendations on how to configure the ServerIOTimeout attribute, see the web server plug-in configuration technote on the IBM Support website.

- **ClusterAddress**

  A ClusterAddress is like a server element in that you can specify the same attributes and elements as for a server element. The difference is that you can define only one of them within a ServerCluster. Use a ClusterAddress when you do not want the plug-in to perform any type of load balancing because you already have some type of load balancer in between the plug-in and the application server.

Note: If you include a ClusterAddress tag, you must include the `Name` attribute on that tag. The plug-in uses the `Name` attribute to associate the cluster address with the correct host and port. If you do not specify the `Name` attribute, the plug-in assigns the cluster address the name that is specified for the server that is using the same host and port.

```
<ClusterAddress Name="MyClusterAddr">
<Transport Hostname="192.168.1.2" Port="9080" Protocol="HTTP"/>
<Transport Hostname="192.168.1.2" Port="9443" Protocol="HTTPS">
</ClusterAddress>
```

If a request comes in that does not have affinity established, the plug-in routes it to the cluster address, if defined. If affinity has been established, then the plug-in routes the request directly to the clone, bypassing the cluster address entirely. If no cluster address is defined for the server cluster, then the plug-in load balances across the servers in the primary servers list.

There can be zero or one element for each ServerCluster.

- **PrimaryServers**

  Specifies a list of servers to which the plug-in routes requests for this cluster. If a list of primary servers is not specified, the plug-in routes requests to servers defined for the server cluster. Specify zero or one element for each ServerCluster.

- **BackupServers**

  Specifies a list of servers to which requests are sent if all servers that are specified in the primary servers list are unavailable. The plug-in does not load balance across the backup servers, but traverses the list in order until no servers are left in the list or until a request is successfully sent and a response is received from an application server. Specify zero or one element for each ServerCluster.

VirtualHostGroup

Specifies a group of virtual host names that are specified in the HTTP Host header. Use this property to group virtual host definitions together that are configured to handle similar types of requests.

The following example shows a `VirtualHostGroup` element and associated elements and attributes:

```
<VirtualHostGroup Name="Hosts">
<VirtualHost Name="www.x.com"/>
<VirtualHost Name="www.x.com:443"/>
<VirtualHost Name="*:8080"/>
<VirtualHost Name="www.x.com:*"/>
<VirtualHost Name="*:*"/>
</VirtualHostGroup>
```

- **Name**

  Specifies the logical or administrative name to be used for this group of virtual hosts. Specify exactly one attribute for each VirtualHostGroup.

- **VirtualHost**

  Specifies the name used for a virtual or real machine used to determine if incoming requests must be handled by WebSphere Application Server. Use this element to specify host names that are in the HTTP Host header which must be seen for requests that need to be handled by the application server. You can specify specific host names and ports for incoming requests or specify an asterisk (*) for either the host name, port, or both.

  There can be one or more elements for each VirtualHostGroup.

  – **Name**

    Specifies that the name in the HTTP Host header that matches the name in the VirtualHost. Specify exactly one attribute for each VirtualHost.

    The value is a host name or IP address and port combination, separated by a colon.

    You can configure the plug-in to route requests to the application server based on the incoming HTTP Host header and port for the request. The `Name` attribute specifies those combinations.

You can use a wildcard for this attribute. The only acceptable solutions are either an asterisk (*) for the host name, an asterisk for the port, or an asterisk for both. An asterisk for both means that any request matches this rule. If no port is specified in the definition, the default HTTP port of 80 is assumed.

UriGroup

Specifies a group of URIs that are specified on the HTTP request line. The same application server must be able to handle the URIs. The route compares the incoming URI with the URIs in the group to determine if the application server handles the request.

The following example shows a UriGroup element and associated elements and attributes:

```
<UriGroup Name="Uris">
<Uri Name="/servlet/snoop"/>
<Uri Name="/webapp/*"/>
<Uri Name="*.jsp"/>
</UriGroup>
```

- **Name**

  Specifies the logical or administrative name for this group of URIs. Specify exactly one attribute for each UriGroup.
- **Uri**

  Specifies the virtual path to the resource that is serviced by WebSphere Application Server. Each URI specifies the incoming URLs that the application server needs to handle. You can use a wildcard in these definitions. There can be one or more attributes for each UriGroup.
  - **Name**

    Specifies the actual string to specify in the HTTP request line to match successfully with this URI. You can use a wildcard within the URI definition. You can specify rules such as *.jsp or /servlet/* to be handled by WebSphere Application Server. When you assemble your application, if you specify **File Serving Enabled**, then only a wildcard URI is generated for the web application, regardless of any explicit servlet mappings. If you specify **Serve servlets by classname**, then the following URI is generated: `<Uri Name="`*`Web_application_URI`*`/servlet/*">`

    There is exactly one attribute for each URI.
  - **AffinityCookie**

    Specifies the name of the cookie that the plug-in uses when trying to determine if the inbound request has session affinity. The default value is **JSESSIONID**.

    There can be zero or one attribute for each URI.
  - **AffinityURLIdentifier**

    Specifies the name of the identifier that the plug-in uses when trying to determine if the inbound request has affinity specified in the URL to a particular clone. The default value is **jsessionid**.

    There can be zero or one attribute for each URI.

Route

Specifies a request routing rule by which the plug-in determines if an incoming request must be handled by WebSphere Application Server.

The route definition is the central element of the plug-in configuration. It specifies how the plug-in handles requests, based on certain characteristics of the request. The route definition contains the other main elements: a required ServerCluster, and either a VirtualHostGroup, UriGroup, or both.

Using the information that is defined in the VirtualHostGroup and the UriGroup for the route, the plug-in determines if the incoming request to the web server is sent on to the ServerCluster element that is defined in this route.

See the following example of this element:

```
<Route VirtualHostGroup="Hosts" UriGroup="Uris" ServerCluster="servers/>
```

- **VirtualHostGroup**

  Specifies the group of virtual hosts that are used in route determination. The incoming host header and server port are matched to determine if this request is handled by the application server.

  It is possible to omit this property from the route definition. If it is not present, then every request matches during the virtual host match portion of route determination.

  There can be zero or one attribute for each Route.
- **UriGroup**

  Specifies the group of URIs to use for determining the route. Select zero or one group for each route. The incoming URI for the request is matched to the defined URIs in this group to determine whether this request is handled by the application server.

  It is possible to omit this property from the route definition. If it is not present, then every request matches during the URI match portion of route determination.
- **ServerCluster**

  Specifies the cluster that receives the requests that successfully matches the route. Select exactly one cluster for each route.

  The cluster is used to handle this request. If both the URI and the virtual host matching is successful for this route, then the request is sent to one of the servers that is defined within this cluster.

RequestMetrics

Used to determine whether request metrics are enabled, and how to filter the requests based on the Internet Protocol (IP) and Uniform Resource Identifiers (URI) filters when request metrics are enabled.

See the following example of this element:

```
<RequestMetrics armEnabled="false"  loggingEnabled="true"
   rmEnabled="false" traceLevel="PERF_DEBUG">
```

- **armEnabled**

  Specifies whether the ARM 4 agent is enabled in the plug-in. When it is set to `true`, the ARM 4 agent is called.

  Note: For the SunOne (iPlanet) web Server the following directive must be included in the `obj.conf` file to enable ARM 4 support:

  ```
  AddLog fn="as_term"
  ```

  If this directive is not included, the arm_stop procedure is never called.

  Select zero or one attribute for RequestMetrics
- **loggingEnabled**

  Specifies whether request metrics logging is enabled in the plug-in. When it is set to `true` and the traceLevel is not set to NONE, the request response time, and other request information, is logged. When it is set to `false`, there is no request logging. The value of loggingEnabled depends on the value specified for the system property, com.ibm.websphere.pmi.reqmetrics.loggingEnabled. When this system property is not present, loggingEnable is set to `true`. Specify exactly one attribute for RequestMetrics.
- **rmEnabled**

  Specifies whether the request metrics are enabled in the plug-in. When it is set to `true`, the plug-in, request metrics, inspects the filters and logs the request trace record in the plug-in log file. This action is performed if a request passes the filters. When this attribute is set to `false`, the rest of the request metrics attributes are ignored. Specify exactly one attribute for RequestMetrics.
- **traceLevel**

Indicates how much information is logged when the `rmEnabled` attribute is set to `true`. When this attribute is set to `NONE`, no request logging is performed. When this attribute is not set to `NONE`, and loggingEnabled is set to `true`, the request response time, and other request information, is logged when the request is done. Specify exactly one attribute for RequestMetrics.

- **filters**

  When rmEnabled is `true`, the filters control which requests are traced. Specify zero, one, or two attributes for RequestMetrics.

  - **enable**

    When enable is `true`, the type of filter is on and requests must pass the filter. Specify exactly one attribute for each filter.

  - **type**

    There are two types of filters: SOURCE_IP (for example, client IP address) and URI. For the SOURCE_IP filter type, requests are filtered based on a known IP address. You can specify a mask for an IP address using the asterisk (*). If the asterisk is used, the asterisk must always be the last character of the mask, for example 127.0.0.*, 127.0.*, 127*. For performance reasons, the pattern matches character by character, until either an asterisk is found in the filter, a mismatch occurs, or the filters are found as an exact match.

    For the URI filter type, requests are filtered based on the URI of the incoming HTTP request. The rules for pattern matching are the same as matching SOURCE_IP address filters.

    If both URI and client IP address filters are enabled, request metrics requires a match for both filter types. If neither is enabled, all requests are considered a match.

    There is exactly one attribute for each filter.

  - **filterValues**

    Specifies the detailed filter information. Specify one or multiple attributes for each filter.

    - **value**

      Specifies the filter value for the corresponding filter type. This value might be either a client IP address or a URI. Specify exactly one attribute for each filterValue.

  - **enableESIToPassCookies**

    Specifies whether to allow forwarding of session cookies to WebSphere Application Server when processing ESI include requests. If the value is set to `true`, this custom property is enabled. If the value is set to `false`, the custom property is disabled. By default, the value is set to `false`.

  - **GetDWLMTable**

    Specifies whether to allow a newly created plug-in process to proactively request a partition table from WebSphere Application Server before it handles any HTTP requests. This custom property is used only when memory-to-memory session management is configured. If the value is set to `true`, this custom property is enabled. If the value is set to `false`, the custom property is disabled. By default, the value is set to `false`.

Web server plug-in custom properties

If you are using a web server plug-in, you can add one or more of the following custom properties to the configuration settings for that plug-in.

Complete these steps to add a web server plug-ins custom property.

1. In the administrative console, select **Servers > Server Types > Web servers >** *web_server_name* **> Plug-in properties > Custom properties > New**.
2. Under **General Properties**, specify the name of the custom property in the **Name** field and a value for this property in the **Value** field. You can also specify a description of this property in the **Description** field.
3. Click **Apply** or **OK**.
4. Click **Save** to save your configuration changes.

5. Re-generate and propagate the `plugin-cfg.xml` file.

Note: You can update the *global* `plugin-cfg.xml` file using the administrative console or running the GenPluginCfg command for all of the clusters in a cell. However, you must delete the `config/cells/plugin-cfg.xml` file before you update the *global* `plugin-cfg.xml` file. If you do not delete the `config/cells/plugin-cfg.xml` file, only the new properties and their values are added to the *global* `plugin-cfg.xml` file. Any updates to existing plug-in property values are not added to the *global* `plugin-cfg.xml` file.

CertLabel

Specifies the label of the certificate within the keyring that the plug-in is to use when the web container requests a client certificate from the plug-in. This custom property does not apply to any client certificate that is coming from the SSL connection with the browser. If you are using an SSL co-processor, and the plug-in is not running on a z/OS or IBM i system, if you specify the token label, followed by a colon, as the value for this custom property the entire CertLabel value is used as the keyring label.

gotcha: You can only use this custom property if you are running on Version 7.0.0.3 or later.

Data type	Boolean
Default	False

GetDWLMTable

Specifies whether the plug-in should prefetch the partition table. When this custom property is enabled, the plugin prefetches the partition table so that affinity requests are maintained. The GetDWLMTable custom property must be enabled when memory-to-memory session management is configured for WebSphere Application Server.

Data type	String
Default	False

HTTPMaxHeaders

Specifies the maximum number of headers that can be included in a request or response that passes through the web server plug-in. If a request or response contains more than the allowable number of headers, the web server plug-in drops the extra headers.

Data type	Integer
Range	1 - 4000
Default	300

If you prefer, instead of adding this custom property, you can manually add the following values to the plugin-cfg.xml file:

```
HTTPMaxHeaders = "<value>" in the Config tag. Example :
<Config ASDisableNagle="false" AcceptAllContent="false"
AppServerPortPreference="HostHeader" ChunkedResponse="false"
FIPSEnable="false" HTTPMaxHeaders="2500"
IISDisableNagle="false" IISPluginPriority="High"
IgnoreDNSFailures="false" RefreshInterval="60"
ResponseChunkSize="64" VHostMatchingCompat="false">
```

SSLConsolidate

Specifies whether the web server plug-in is to compare the setup of each new SSL transport with the setup of other SSL transports that are already defined in the configuration file. When you set this property to `true`, and the plug-in determines that the keyring and CertLabel values specified for the new SSL transport match the values specified for an already defined SSL transport, the plug-in uses the existing SSL environment instead of creating a new SSL environment. Creating fewer SSL environments means that the plug-in requires less memory, and the plug-in initialization time decreases, thereby optimizing your overall GSkit environment.

Data type	Boolean
Default	False

TrustedProxyEnable

Permits the web server plug-in to interface with the proxy servers and load balancers that are listed for the TrustedProxyList custom property. When this property is set to `true`, the proxy servers and load balancers in this trusted proxy list can set values for the $WSRA and $WSRH internal headers. The $WSRA internal header is the IP address of the remote host, which is typically the browser, or an internal address that is obtained by Network Address Translation (N.A.T.). The $WSRH internal header is the host name of the remote host. This header information enables the web server plug-in to interface with that specific proxy server or load balancer.

When you use this custom property you must also use the TrustedProxyList custom property to specify a list of trusted proxy servers and load balancers. Also, you must clear the Remove special headers check box on the Request Routing panel within the administrative console. For more information, see the documentation on web server plug-in request routing properties.

Data type	Boolean
Default	False

TrustedProxyList

Specifies a comma delimited list of all proxy servers or load balancers that have permission to interface with this web server plug-in. You must use this property with the `TrustedProxyEnable=true` custom property setting. If the TrustedProxyEnable custom property is set to `false`, this list is ignored.

Example:

```
TrustedProxyList = myProxyServer1.myDomain.com,myProxyServer2.com,192.168.0.1
```

Data type	String
Default	None

Web server plug-in configuration properties

Web server plug-in configuration properties are set on various panels of the administrative console. The table provided indicates on which panel a particular property is set.

Table 6. Web server plug-in configuration properties. The following table indicates which panel in the administrative console you need to use to manually configure a web server plug-in property.

Administrative console panel	Field name	Configuration property name

Table 6. Web server plug-in configuration properties (continued). The following table indicates which panel in the administrative console you need to use to manually configure a web server plug-in property.

In the administrative console, click **Servers** > **Server Types** > **Web servers** > *web_server_name* > **Plug-in properties**	Refresh configuration interval	RefreshInterval
In the administrative console, click **Servers** > **Server Types** > **Web servers** > *web_server_name* > **Plug-in properties**	Plug-in log file name	Log->name
In the administrative console, click **Servers** > **Server Types** > **Web servers** > *web_server_name* > **Plug-in properties**	Plug-in logging	Log->LogLevel
In the administrative console, click **Servers** > **Server Types** > **Web servers** > *web_server_name* > **Plug-in properties**	Ignore DNS failures during web server startup	IgnoreDNSFailures
In the administrative console, click **Servers** > **Server Types** > **Web servers** > *web_server_name* > **Plug-in properties**	KeyringLocation	Keyring
In the administrative console, click **Servers** > **Server Types** > **Web servers** > *web_server_name* > **Plug-in properties**	StashfileLocation	Stashfile
In the administrative console, click **Servers** > **Server Types** > **Web servers** > *web_server_name* > **Plug-in properties** > **Custom properties** > **New**	FIPSEnable	FIPSEnable
In the administrative console, click **Servers** > **Web Servers** > *Web_server_name*> **Plug-in properties** > **Custom properties** > **New**	TrustedProxyEnable	TrustedProxyEnable
In the administrative console, click **Servers** > **Web Servers** > *Web_server_name*> **Plug-in properties** > **Custom properties** > **New**	TrustedProxyList	TrustedProxyList
In the administrative console, click **Servers** > **Web Servers** > *Web_server_name*> **Plug-in properties** > **Custom properties** > **New**	SSLConsolidate	SSLConsolidate
In the administrative console, click **Servers** > **Server Types** > **Web servers** > *Web_server_name* > **Plug-in properties** > **Request routing**	Load balancing option	LoadBalance

Table 6. Web server plug-in configuration properties (continued). The following table indicates which panel in the administrative console you need to use to manually configure a web server plug-in property.

In the administrative console, click **Servers > Server Types > Web servers >** *web_server_name* **> Plug-in properties > Request Routing**	Clone separator change	CloneSeparatorChange
In the administrative console, click **Servers > Server Types > Web servers >** *web_server_name* **> Plug-in properties > Request Routing**	Retry interval	RetryInterval
In the administrative console, click **Servers > Server Types > Web servers >** *web_server_name* **> Plug-in properties > Request routing**	Maximum size of request content	PostSizeLimit
In the administrative console, click **Servers > Server Types > Web servers >** *web_server_name* **> Plug-in properties > Request routing**	Size of the buffer that is used to cache POST requests	PostBufferSize
In the administrative console, click **Servers > Server Types > Web servers >** *web_server_name* **> Plug-in properties > Request routing**	Remove special headers	RemoveSpecialHeaders
In the administrative console, click **Servers > Server Types > WebSphere application servers >** *server_name* **> Web server plug-in properties**	Server role	PrimaryServers and BackupServers list
In the administrative console, click **Servers > Server Types > WebSphere application servers >** *server_name* **> Web server plug-in properties**	Connect timeout	Server ConnectTimeout
In the administrative console, click **Servers > Server Types > WebSphere application servers >** *server_name*, and then, in the Additional Properties section, click **Web server plug-in properties**.	The read and write timeouts for all the connections to the application server	ServerIOTimeout
In the administrative console, click **Servers > Server Types > WebSphere application servers >** *server_name* **> Web server plug-in properties**	Use extended handshake to check whether Application Server is running	Server Extended Handshake
In the administrative console, click **Servers > Server Types > WebSphere application servers >** *server_name* **> Web server plug-in properties**	Send the header "100 Continue" before sending the request content	WaitForContinue

Table 6. Web server plug-in configuration properties (continued). The following table indicates which panel in the administrative console you need to use to manually configure a web server plug-in property.

In the administrative console, click **Servers > Server Types > WebSphere application servers >** *server_name* **> Web server plug-in properties**	Maximum number of connections that can be handled by the Application Server	Server MaxConnections
In the administrative console, click **Servers > Server Types > Web servers >** *web_server_name* **> Plug-in properties > Request and Response**	Application server port preference	AppServerPortPreference
In the administrative console, click **Servers > Web Servers >** *Web_server_name* **> Plug-in properties > Request and Response**	Enable Nagle algorithm for connections to the Application Server	ASDisableNagle
In the administrative console, click **Servers > Server Types > Web servers >** *web_server_name* **> Plug-in properties > Request and Response**	Enable Nagle Algorithm for the IIS Web Server	IISDisableNagle
In the administrative console, click **Servers > Server Types > Web servers >** *web_server_name* **> Plug-in properties > Request and Response**	Virtual host matching	VHostMatchingCompat
In the administrative console, click **Servers > Server Types > Web servers >** *web_server_name* **> Plug-in properties > Request and Response**	Maximum chunk size used when reading the response body	ResponseChunkSize
In the administrative console, click **Servers > Server Types > Web servers >** *web_server_name* **> Plug-in properties > Request and Response**	Accept content for all requests	AcceptAllContent
In the administrative console, click **Servers > Server Types > Web servers >** *web_server_name* **> Plug-in properties > Request and Response**	Chunk HTTP response to the client	ChunkedResponse
In the administrative console, click **Servers > Server Types > Web servers >** *web_server_name* **> Plug-in properties > Request and Response**	Priority used by the IIS Web Server when loading the plug-in configuration file	IISPluginPriority
In the administrative console, click **Servers > Server Types > Web servers >** *web_server_name* **> Plug-in properties > Caching**	Enable Edge Side Include (ESI) processing to cache the responses	ESIEnable
In the administrative console, click **Servers > Server Types > Web servers >** *web_server_name* **> Plug-in properties > Caching**	Maximum cache size	ESIMaxCacheSize

Table 6. Web server plug-in configuration properties (continued). The following table indicates which panel in the administrative console you need to use to manually configure a web server plug-in property.

In the administrative console, click **Servers** > **Server Types** > **Web servers** > *web_server_name* > **Plug-in properties** > **Caching**	Enable invalidation monitor to receive notifications	ESIInvalidationMonitor
In the administrative console, click **Servers** > **Web Servers** > *Web_server_name* > **Plug-in properties** > **Caching**	Forwarding of session cookies to WebSphere Application Server.	enableESIToPassCookies

Web server plug-in tuning tips

Important tips for web server plug-in tuning include how to balance workload and improve performance in a high stress environment. Balancing workloads among application servers in a network fronted by a web server plug-in helps improve request response time.

Limiting the number of connections that can be established with an application server works best for web servers that follow use a single, multithreaded process for serving requests.

```
ServerLimit          1
ThreadLimit          1024
StartServers         1
MaxClients           1024
MinSpareThreads      1
MaxSpareThreads      1024
ThreadsPerChild      1024
MaxRequestsPerChild  0
```

Improving performance in a high stress environment

To tune the TcpTimedWaitDelay setting, change the value of the tcp_time_wait_interval parameter from the default value of 240 seconds, to 30 seconds:

1. Locate in the Windows Registry:

   ```
   HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\tcpip\Parameters\TcpTimedWaitDelay
   ```

   If this entry does not exist in your Windows Registry, create it by editing this entry as a new DWORD item.
2. Specify, in seconds, a value between 30 and 300 inclusive for this entry. (It is recommended that you specify a value of 30.)

To tune the MaxUserPort setting:

1. Locate in the Windows Registry:

   ```
   HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\tcpip\Parameters\MaxUserPort
   ```

   If this entry does not exist in your Windows Registry, create it by editing this entry as a new DWORD item.
2. Set the maximum number of ports to a value between 5000 and 65534 ports, inclusive. (It is recommended that you specify a value of 65534,)

See the Microsoft website for more information about these settings.

Appendix. Directory conventions

References in product information to *app_server_root*, *profile_root*, and other directories imply specific default directory locations. This topic describes the conventions in use for WebSphere Application Server.

Default product locations - IBM i

These file paths are default locations. You can install the product and other components in any directory where you have write access. You can create profiles in any valid directory where you have write access. Multiple installations of WebSphere Application Server products or components require multiple locations.

app_client_root
: The default installation root directory for the Application Client for IBM WebSphere Application Server is the `/QIBM/ProdData/WebSphere/AppClient/V8/client` directory.

app_client_user_data_root
: The default Application Client for IBM WebSphere Application Server user data root is the `/QIBM/UserData/WebSphere/AppClient/V8/client` directory.

app_client_profile_root
: The default Application Client for IBM WebSphere Application Server profile root is the `/QIBM/UserData/WebSphere/AppClient/V8/client/profiles/`*`profile_name`* directory.

app_server_root
: The default installation root directory for WebSphere Application Server - Express is the `/QIBM/ProdData/WebSphere/AppServer/V8/Express` directory.

java_home

Table 7. Root directories for supported Java Virtual Machines.

This table shows the root directories for all supported Java Virtual Machines (JVMs).

JVM	Directory
32–bit IBM Technology for Java	`/QOpenSys/QIBM/ProdData/JavaVM/jdk60/32bit`
64–bit IBM Technology for Java	`/QOpenSys/QIBM/ProdData/JavaVM/jdk60/64bit`

plugins_profile_root
: The default Web Server Plug-ins profile root is the `/QIBM/UserData/WebSphere/Plugins/V8/webserver/profiles/`*profile_name* directory.

plugins_root
: The default installation root directory for Web Server Plug-ins is the `/QIBM/ProdData/WebSphere/Plugins/V8/webserver` directory.

plugins_user_data_root
: The default Web Server Plug-ins user data root is the `/QIBM/UserData/WebSphere/Plugins/V8/webserver` directory.

product_library
product_lib
: This is the product library for the installed product. The product library for each Version 8.0 installation on the system contains the program and service program objects (similar to `.exe`, `.dll`, `.so` objects) for the installed product. The product library name is `QWAS8`*`x`* (where *x* is A, B, C, and so on). The product library for the first WebSphere Application Server Version 8.0 product installed on the system is `QWAS8A`. The *`app_server_root`*`/properties/product.properties` file contains the value for the product library of the installation, `was.install.library`, and is located under the *`app_server_root`* directory.

profile_root
: The default directory for a profile named *profile_name* for WebSphere Application Server - Express is the `/QIBM/UserData/WebSphere/AppServer/V8/Express/profiles/`*`profile_name`* directory.

shared_product_library
: The shared product library, which contains all of the objects shared by all installations on the system, is `QWAS8`. This library contains objects such as the product definition, the subsystem description, the job description, and the job queue.

user_data_root
: The default user data directory for WebSphere Application Server - Express is the `/QIBM/UserData/WebSphere/AppServer/V8/Express` directory.

  The `profiles` and `profileRegistry` subdirectories are created under this directory when you install the product.

web_server_root
: The default web server path is /www/*web_server_name*.

Notices

Trademarks and service marks

Index