IBM WebSphere Application Server for Distributed
Platforms, Version 8.0

Setting up intermediary services

IBM

Compilation date: July 14, 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.

Create WebSphere profiles.

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.

  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 an operation is not enabled when a servlet or JavaServer Pages (JSP) file requiring the operation is accessed, an error message is displayed, such as this one from the IBM HTTP Server:

  `HTTP method POST is not supported by this URL.`

- Make sure the appropriate plug-in file has been installed on your web server and the `configureweb_server_name` script has been run to create and configure the web server definition for this web server.

  If you are using a distributed platform web server, use the web Server Plug-ins Configuration Tool to install the appropriate plug-in file to your web server. Then, run the `configureweb_server_name` script created by the tool to create and configure the web server definition in the WebSphere configuration repository.

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 following steps are performed during the plug-in installation process. See the Plug-in Installation Roadmap for additional information.

1. A web server definition is created.

   You can also use either the administrative console or use the `ConfigurewebServerDefintion.jacl` script to create a web server definition.

2. An application or modules are mapped to a web server. If an application that you want to use with this web server is already installed, the application is automatically mapped to the web server. If the application is not installed, select this web server during the "Map modules to servers" step of the application installation process.

3. The master repository is updated and saved.

When you configure a plug-in, the configuration file for that plug-in is automatically created. You can change or tune the default settings for the properties in this configuration file. If you change any of the settings, you must regenerate the file before your changes take effect.

Generating or regenerating the configuration file 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. If the application server is on the same physical workstation as the web server, the regeneration usually takes about 30 to 60 seconds to complete. The regeneration takes longer if the application server and web server are on different workstations.

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

   gotcha: When the plug-in configuration file is first generated, it does not include admin_host on the list of virtual hosts. The *Installing your application serving environment* PDF describes how to add it to the list.

   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. 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. Install the appropriate GSKIT installation image file on your workstation, if you want to use Secure Sockets Layer (SSL) with this configuration.
3. Tune your web server. See the *Tuning guide* PDF for more information.
4. Propagate the plug-in configuration. The plug-in configuration file, `plugin-cfg.xml`, is automatically propagated to the web server if the web server plug-in configuration service is enabled, and one of the following conditions is true:
   - The web server is a local web server, which means that the web server is located on the same workstation as an application server.
   - The web server is a remote IBM HTTP Server Version 7 that has a running IBM HTTP Server administration server.

   If neither of these conditions are true, you must manually copy the `plugin-cfg.xml` file to the location of the remote web server installation. Copy the `plugin-cfg.xml` file in *<app_server_root>*/profiles/ *<profilename>*/config/cells/../../nodes/../servers/*<webservername>* to the web server host location, which is *<PluginInstallRoot>*/config/*<webservername>*/.

   Important: If you use the FTP function to copy the file, and the configuration reload fails, check the file permissions on the `plugin-cfg.xml` file, and make sure that they are set to `rw-r--r--`. If the file permissions are not correct, the web server is not able to access the new version of the file, which causes the configuration reload to fail.

   If the file permissions are incorrect, issue the following command to change the file permissions to the appropriate settings:

   `chmod 644 plugin-cfg.xml`

       AIX   The AIX® FTP function does not preserve file attributes. Therefore, if you need to manually copy the `plugin-cfg.xml` from an AIX operating system, you might want to use the AIX RCP function instead of the FTP function to copy the file.

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 local web server

This topic describes how to install the web server and the web server plug-in on the machine where you installed WebSphere Application Server.

Before you begin

If the web server that you are setting up is an IBM HTTP Server, and you plan to manage that web server through a node agent that is running as a nonroot user, you must make sure that you adhere to the following requirements:
- The user ID that you designate as the user ID that owns the IBM HTTP Server directories and files, is the same user ID under which the nonroot node agent is running. You cannot run an IBM HTTP Server

as a root user if the node agent that is managing that IBM HTTP Server is running as nonroot node agent because a node agent process that is running as a nonroot user cannot spawn off an IBM HTTP Server that is running as a root user.

- The value you specify for the listener port value must be greater than 1024. An IBM HTTP Server that is running under a nonroot user ID does not start if the port number for its listener port is 1024 or less.

You can ensure that the nonroot node agent and the IBM HTTP Server are using the same user ID if you specify the user ID that you used to install the product as the user ID for the IBM HTTP Server when you install the IBM HTTP Server. However, if, you decide to run the node agent as a nonroot user after you install the IBM HTTP Server and web server plug-in, you can take the following actions to enable both the node agent and the IBM HTTP Server to run as nonroot users:

1. Change the user ID for WebSphere Application Server to a nonroot user ID.
2. Configure the run-as setting for the node agent.
3. Use the administrative console to create a new IBM HTTP Web Server, unless an already defined IBM HTTP Server has the required properties.
4. Change the ownership of the IBM HTTP Server directory and files to the nonroot user ID under which the nonroot node agent is running.

About this task

The following steps create a web server definition in the default profile.

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. Install the WebSphere Customization Toolbox.
6. Configure the web server plug-in using the Web Server Plug-ins Configuration Tool.

   The web server definition is automatically created and configured.
7. Complete the setup by creating the web server definition using the WebSphere Application Server administrative console, or run the plug-in configuration script. The creation of this object is exclusive of the web server installation.

   You must create an application server profile or a custom profile and federate the node before you can use the administrative console of the deployment manager to create a web server definition. The same is true for running the configuration script that the Web Server Plug-ins Configuration Tool created. You must assign the web server to a managed node when you create it. The managed node must exist before running the Web Server Plug-ins Configuration Tool. Otherwise, the installation is considered a remote installation.

   Select one of the following options:
   - **Using the administrative console.**

     Create a web server definition on an existing application server.
     a. Click **Servers > Server Types > Web servers > New** and use the **Create new web server definition** tool to create the web server definition.
     b. Select a template. Select a system template or a user-defined template for the web server that you want to create.
     c. Enter the web server properties:
        – Type: The web server vendor type
        – Port: The existing web server port (default: 80)

- Installation path: The web server installation path. This field is required for IBM HTTP Server only.
  - Service name (Windows operating systems): 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 in which the plug-in is installed.
  d. Confirm the creation of the new web server, and click **Finish**.

  After creating the web server, complete the following steps to verify that the `plugin-key.kdb` file is generated and to configure the web server plug-in with SSL:

  a. Click **Security > SSL certificate and key management**.
  b. Under Configuration settings, click **Manage endpoint security configurations**.
  c. Under Inbound or Outbound, expand *cell_name* > **nodes** > *Web_server_node_name* > **servers** and click *server_name*.
  d. Under Related Items, click **Key stores and certificates**. The administrative console displays the CMSKeyStore configuration with the path to the `plugin-key.kdb` file.
  e. Export the default certificate from `key.p12`, and add it as a signer certificate to the `plugin-key.kdb`.
- **Running the plug-in configuration script.**

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. Install the WebSphere Customization Toolbox.
6. Configure the web server plug-in using the Web Server Plug-ins Configuration Tool.
7. 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.**

8. **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.

9. **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.

10. 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

Install the IBM HTTP Server product and its plug-in, or install a plug-in for another supported web server to enable the web server to work with WebSphere Application Server.

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.

The Web Server Plug-ins Configuration Tool configures supported web servers. You can also 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

The Web Server Plug-ins Configuration Tool configures supported web servers. Use this topic to understand how the tool configures the web server configuration files.

If you must change a configuration for some reason, you can run the Web Server Plug-ins Configuration Tool again to re-configure the web server or you can edit the files.

To support a nondefault profile, edit the configuration to point the web server to the correct location of the plug-in configuration file (`plugin-cfg.xml`). In this case, you would change the profile path from the default profile to the secondary profile.

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:
  - **AIX** **HP-UX** **Linux** **Solaris** Run the command:

    `apachectl -V | grep Architecture`

    This command displays the Architecture value for the web server, which will be either 32-bit or 64-bit.
  - **Windows** Run the following command from the *IBM_HTTP_Server_install_root*\bin directory:

    `apache -V`

    Look for the value of the `Architecture:` line in the output. This value is either 32-bit or 64-bit.

- For other, non-Apache based web servers, use the following techniques. These techniques also work for Apache-based web servers.

  – `AIX` `HP-UX` `Linux` `Solaris` Run the file command against the primary web server application executable file to display its type. For example, run the following command on an AIX system:

    ```
    file /usr/bin/httpd
    ```

    This command results in the following output:

    - For a 64-bit web server: `httpd: 64-bit XCOFF executable ...`
    - For a 32-bit web server: `httpd: executable (RISC System/6000)...`

    In general, if the output of the file command does not indicate `64` in the output, then the application is 32-bit application.

  – `Windows` Open the Microsoft Windows Task Manager when the server is running and locate the server in the process list. A 32-bit application has *32 after its name.

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
- Configure Apache HTTP Server 2.2. See "Configuring Apache HTTP Server V2.2" on page 14.
- Configure Domino® Domino Web Server Version 5 and Version 6.x. See "Configuring Lotus Domino" on page 16.
- Configure IBM HTTP Server powered by Apache 2.x. See "Configuring IBM HTTP Server powered by Apache 2.x" on page 19.
- Configure IBM HTTP Server Version 6.x. See "Configuring IBM HTTP Server Version 6.x" on page 20.
- Configure IBM HTTP Server Version 7.0. See "Configuring IBM HTTP Server Version 7.0" on page 22.
- Configure IBM HTTP Server Version 78.0. See "Configuring IBM HTTP Server Version 8.0" on page 23.
- Configure Microsoft Internet Information Services (IIS). See "Configuring Microsoft Internet Information Services (IIS)" on page 25.
- Configure Sun Java System Web Server (formerly Sun ONE and iPlanet). See "Configuring the Sun Java System Web Server" on page 29.

Results

You can use the Web Server Plug-ins Configuration Tool to configure supported web servers. You can also configure a web servers by editing its configuration.

Configuring Apache HTTP Server V2.0

This topic describes how to change configuration settings for Apache HTTP Server Version 2.0.

Before you begin

After you install the web server plug-ins, you can use the Web Server Plug-ins Configuration Tool to configure a web server plug-in.

This topic describes how to configure the Apache HTTP Server Version 2.0 Web Server. Other procedures in "Editing web server configuration files" on page 10 describe configuring other supported web servers.

gotcha:

- If you are using an Apache HTTP Server that supports 64-bit addressing, you must use the 64-bit CD provided with the WebSphere Application Server product to install the Apache Web Server plug-in binaries. If you use the 32-bit CD, you will receive an error message indicating that the plug-in binaries did not load.

- If you are using an Apache HTTP Server that supports 32-bit addressing, you must use the 32-bit CD provided with the WebSphere Application Server product to install the Apache Web Server plug-in binaries. If you use the 64-bit CD, you will receive an error message indicating that the plug-in binaries did not load.

A sample error message follows:

```
httpd: Syntax error on line XXX of /home/apache/conf/httpd.conf: Cannot
load /home/apache/Plugins/mod_was_ap20_http.sl into server: Invalid argument
```

The plug-in was tested with the threaded worker multi-processing module (MPM) on all platforms except Windows. The plug-in was tested with the default threaded MPM on Windows.

The plug-in works with the Apache 2 prefork MPM but works best with the worker MPM. The plug-in maintains connection pools to backend WebSphere Application Servers and uses in-memory caching. These plug-in functions perform most efficiently when Apache 2.0 is configured to use a single child process with the ThreadsPerChild value equal to the MaxClients value. The plug-in can be used with the prefork MPM or the worker MPM that is configured with multiple child processes, but at reduced efficiency.

Compatibility Statement The plug-in works with versions of the Apache HTTP Server that claim full binary compatibility with Apache 2.0.47 and later, which are built with compilers and compiler options that are compatible with those used to build the plug-in.

About this task

Perform the step that configures Apache 2.0 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 Web Server Plug-ins Configuration Tool installs a dummy `plugin-cfg.xml` file during installation, but this file requires periodic propagation from the real file on the application server machine.

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

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

Procedure

- **AIX** Configure entries in the `httpd.conf` file.

  Use the following examples of the LoadModule and the WebSpherePluginConfig directives as models for configuring your file:

  ```
  LoadModule
        was_ap20_module /usr/IBM/WebSphere/Plugins/bin/mod_was_ap20_http.so
  ```

Example:

```
WebSpherePluginConfig
      /usr/IBM/WebSphere/Plugins/config/webserver1/plugin-cfg.xml
```

- `Linux` `Solaris` Configure entries in the `httpd.conf` file.

  Use the following examples of the LoadModule and the WebSpherePluginConfig directives as models for configuring your file:

```
LoadModule
      was_ap20_module /opt/IBM/WebSphere/Plugins/bin/mod_was_ap20_http.so
```

  Example:

```
WebSpherePluginConfig
      /opt/IBM/WebSphere/Plugins/config/webserver1/plugin-cfg.xml
```

  `Solaris` On the Solaris SPARC 64-bit platform, the Web Server Plug-ins Configuration Tool installs both 32-bit and 64-bit versions of the plug-in for Apache 2.0; however, it configures the web server to use the 32-bit plug-in only. If the web server is 64-bit, you need to configure the LoadModule directive in the `httpd.conf` file to use the 64-bit plug-in as follows:

```
LoadModule
      was_ap20_module /usr/IBM/WebSphere/Plugins/bin/64bits/mod_was_ap20_http.so
```

- `HP-UX` Configure entries in the `httpd.conf` file.

  Use the following examples of the LoadModule and the WebSpherePluginConfig directives as models for configuring your file:

```
LoadModule
      was_ap20_module /opt/IBM/WebSphere/Plugins/bin/mod_was_ap20_http.sl
```

  Example:

```
WebSpherePluginConfig
      /opt/IBM/WebSphere/Plugins/config/webserver1/plugin-cfg.xml
```

- `Windows` Configure entries in the `httpd.conf` file.

  Use the following examples of the LoadModule and the WebSpherePluginConfig directives as models for configuring your file:

```
LoadModule was_ap20_module
      drive:\IBM\WebSphere\Plugins\bin\mod_was_ap20_http.dll
```

  Example:

```
WebSpherePluginConfig
      C:\IBM\WebSphere\Plugins\config\webserver1\plugin-cfg.xml
```

Results

This procedure results in re-configuring the Apache 2.0 Web Server.

What to do next

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 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 Apache HTTP Server V2.2

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

Before you begin

Install Apache 2.2 and the latest version of the web server plug-ins.

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

After you install the web server plug-ins, you can use the Web Server Plug-ins Configuration Tool to configure a web server plug-in.

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 10 describe configuring other supported web servers.

gotcha:
- If you are using an Apache HTTP Server that supports 64-bit addressing, you must use the 64-bit CD provided with the WebSphere Application Server product to install the Apache Web Server plug-in binaries. If you use the 32-bit CD, you will receive an error message indicating that the plug-in binaries did not load.
- If you are using an Apache HTTP Server that supports 32-bit addressing, you must use the 32-bit CD provided with the WebSphere Application Server product to install the Apache Web Server plug-in binaries. If you use the 64-bit CD, you will receive an error message indicating that the plug-in binaries did not load.

A sample error message follows:

```
httpd: Syntax error on line XXX of /home/apache/conf/httpd.conf: Cannot
load /home/apache/Plugins/mod_was_ap22_http.sl into server: Invalid argument
```

The plug-in was tested with the threaded worker multi-processing module (MPM) on all platforms except Windows. The plug-in was tested with the default threaded MPM on Windows.

The plug-in works with the Apache 2.2 prefork MPM but works best with the worker MPM. The plug-in maintains connection pools to backend WebSphere Application Servers and uses in-memory caching. These plug-in functions perform most efficiently when Apache is configured to use a single child process with the ThreadsPerChild value equal to the MaxClients value. The plug-in can be used with the prefork MPM or the worker MPM that is configured with multiple child processes, but at reduced efficiency.

Compatibility Statement The plug-in works with versions of the Apache HTTP Server that claim full binary compatibility with Apache 2.0.47 and later, which are built with compilers and compiler options that are compatible with those used to build the plug-in.

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 Web Server Plug-ins Configuration Tool installs a dummy `plugin-cfg.xml` file during installation, but this file requires periodic propagation from the real file on the application server machine.

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

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

Procedure

- **AIX**   Configure entries in the `httpd.conf` file.

  Use the following examples of the LoadModule and the WebSpherePluginConfig directives as models for configuring your file:

  ```
  LoadModule
        was_ap22_module /usr/IBM/WebSphere/Plugins/bin/mod_was_ap22_http.so
  ```

- **Linux**   **Solaris**   Configure entries in the `httpd.conf` file.

  Use the following examples of the LoadModule and the WebSpherePluginConfig directives as models for configuring your file:

  ```
  LoadModule
        was_ap22_module /opt/IBM/WebSphere/Plugins/bin/mod_was_ap22_http.so
  ```

  Solaris   On the Solaris SPARC 64-bit platform, the Web Server Plug-ins Configuration Tool installs both 32-bit and 64-bit versions of the plug-in for Apache 2.2, however it configures the web server to use the 32-bit plug-in only. If the web server is 64-bit, you need to configure the LoadModule directive in the `httpd.conf` file to use the 64-bit plug-in as follows:

  ```
  LoadModule
        was_ap22_module /usr/IBM/WebSphere/Plugins/bin/64bits/mod_was_ap22_http.so
  ```

- **HP-UX**   Configure entries in the `httpd.conf` file.

  Use the following examples of the LoadModule and the WebSpherePluginConfig directives as models for configuring your file:

  ```
  LoadModule
        was_ap22_module /opt/IBM/WebSphere/Plugins/bin/mod_was_ap22_http.sl
  ```

- **Windows**   Configure entries in the `httpd.conf` file.

  Use the following examples of the LoadModule and the WebSpherePluginConfig directives as models for configuring your file:

  ```
  LoadModule was_ap22_module
        drive:\IBM\WebSphere\Plugins\bin\mod_was_ap22_http.dll
  ```

Results

The Apache 2.2 Web Server is re-configured.

What to do next

The native GSKIT Secure Sockets Layer (SSL) encryption library is used.

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 Lotus Domino

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

Before you begin

The Web Server Plug-ins Configuration Tool configures the web server plug-in.

Other procedures in "Editing web server configuration files" on page 10 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.

    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. | AIX | HP-UX |
    | Linux | Solaris |
    | AIX | HP-UX | Linux | Solaris |

Table 1. Setting the path to the plug-in configuration file. *Set the WAS_PLUGIN_CONFIG_FILE environment variable to the location of the plug-in configuration file using one of the paths in this table.*

If the type of installation is:	Then use this command to set the environment variable:
Remote	`WAS_PLUGIN_CONFIG_FILE=/opt/IBM/WebSphere/Plugins/config/webserver1/plugin-cfg.xml`
Local standalone	`WAS_PLUGIN_CONFIG_FILE=profile_root/config/cells/sa_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.

The `setupPluginCfg.sh` script sets the file path value to the file path that the Web Server Plug-ins Configuration Tool configured originally. If you are reconfiguring the web server to change the original file path, do not use this script. ▨ Windows

▨ Windows

Table 2. Setting the path to the plug-in configuration file. Add the appropriate statement to your lotus_domino_root\ `notes.ini` file.

If the type of installation is:	Then use this command to set the WebSpherePluginCfg variable:
Remote	`WebSpherePluginCfg=C:\Program Files\IBM\WebSphere\Plugins\config\webserver1\ plugin-cfg.xml`
Local standalone	`WebSpherePluginCfg=`*profile_root*`\config\cells\sa_cell\nodes\webserver1_node\ servers\webserver1\plugin-cfg.xml`

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:

- ▨ AIX
  - `/usr/lotus/notes/50091/ibmpow/Notes.jar`

– `/usr/notesdata/names.nsf`
- **Solaris**
  – `/opt/lotus/notes/5080/sunspa/Notes.jar`
  – `/opt/notesdata/names.nsf`
- **Windows**
  – `c:\Program Files\lotus\notes\Notes.jar`
  – `c:\Program Files\lotus\notes\data\names.nsf`

Solaris

The Domino Server plug-in might fail to configure on a Solaris platform

If this occurs during installation, a `dsapi_stderr.txt` file is created in the logs directory and you get the following error messages:

```
lotus.notes.NotesException: Could not load dll for system name SUNOS
        at lotus.notes.NotesThread.load(NotesThread.java:210)
        at lotus.notes.NotesThread.<clinit>(NotesThread.java:24)
java.lang.UnsatisfiedLinkError: NnotesInitThread
        at lotus.notes.NotesThread.NnotesInitThread(Native Method)
        at lotus.notes.NotesThread.initThread(NotesThread.java:99)
        at lotus.notes.NotesThread.run(NotesThread.java:133)
```

You can configure the WebSphere Application Server or Domino Server plug-in manually using the Domino Server Web Administration tool. Use the following procedures:

1. Start the Domino Server.
2. Enter the URL for the Domino Server Web Administration site using a browser. For example, `http://`*host_name*`/names.nsf`. Enter the administrator user name and password.
3. Double-click **Server-Servers**.
4. Double-click **WebServer** to configure.
5. Double-click **Edit Server**.
6. Double-click **Internet Protocol**.
7. Add the WebSphere Application Server DSAPI plug-in to the **DSAPI** field. For example, *app_server_root*`/bin/libdomino5_http.so`

   If there are already DSAPI filter files specified, use a space to delimit the WebSphere Application Server plug-in file.
8. Double-click **Save and Close**.
9. Restart the Domino Server.

AIX **HP-UX** **Linux** **Solaris**

Avoiding a DSAPI filter-loading error when the Lotus Domino Server starts

On operating systems such as AIX or Linux, if the Lotus Domino Web Server starts using a nonroot user, you are likely to generate a DSAPI filter-loading error when the Lotus Domino Server starts:

`Error loading DSAPI filter.`

`Filter not loaded: `*app_server_root*`/bin/libdomino6_http.a`

Manually change the WebSphere Application Server `bin` directory permissions from `750` to `755` to run Lotus Domino Server as a nonroot user and not generate the error.

You must also change permissions on the WebSphere Application Server `logs` directory to `777` to allow Lotus Domino Server to write to the log. However, this change poses a security risk.

If the Lotus Domino Server is started as root, the problem does not occur.

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

After you install the web server plug-ins, you can use the Web Server Plug-ins Configuration Tool to configure a web server plug-in.

This topic describes how to configure the IBM HTTP Server. Other procedures in "Editing web server configuration files" on page 10 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 Plug-ins Configuration Tool installs a dummy `plugin-cfg.xml` file during installation, but this file requires periodic propagation from the real file on the spplication server machine.

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

Procedure

* **AIX** Configure entries in the `httpd.conf` file.

  Use the following examples of the LoadModule and the WebSpherePluginConfig directives as models for configuring your file:

  ```
  LoadModule
       was_ap20_module /usr/IBM/WebSphere/Plugins/bin/mod_was_ap20_http.so
  ```

  Example:

  ```
  WebSpherePluginConfig
       /usr/IBM/WebSphere/Plugins/config/webserver1/plugin-cfg.xml
  ```

* **Linux** **Solaris** Configure entries in the `httpd.conf` file.

  Use the following examples of the LoadModule and the WebSpherePluginConfig directives as models for configuring your file:

  ```
  LoadModule
       was_ap20_module /opt/IBM/WebSphere/Plugins/bin/mod_was_ap20_http.so
  ```

  Example:

  ```
  WebSpherePluginConfig
       /opt/IBM/WebSphere/Plugins/config/webserver1/plugin-cfg.xml
  ```

* **HP-UX** Configure entries in the `httpd.conf` file.

  Use the following examples of the LoadModule and the WebSpherePluginConfig directives as models for configuring your file:

  ```
  LoadModule
       was_ap20_module /opt/IBM/WebSphere/Plugins/bin/mod_was_ap20_http.sl
  ```

  Example:

  ```
  WebSpherePluginConfig
       /opt/IBM/WebSphere/Plugins/config/webserver1/plugin-cfg.xml
  ```

- **Windows** Configure entries in the `httpd.conf` file.

  Use the following examples of the LoadModule and the WebSpherePluginConfig directives as models for configuring your file:

  ```
  LoadModule was_ap20_module
        drive:\IBM\WebSphere\Plugins\bin\mod_was_ap20_http.dll
  ```

  Example:

  ```
  WebSpherePluginConfig
        C:\IBM\WebSphere\Plugins\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.

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 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 6.x

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

Before you begin

The Web Server Plug-ins Configuration Tool configures the web server plug-in.

See "Installing and configuring web server plug-ins" on page 50.

This topic describes how to configure IBM HTTP Server, Version 6.x. Other procedures in "Editing web server configuration files" on page 10 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 Web Server Plug-ins Configuration Tool installs a dummy `plugin-cfg.xml` file during installation, but this file requires periodic propagation from the real file on the spplication server machine.

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

Procedure

- AIX   Configure entries in the `httpd.conf` file.

  Use the following examples of the LoadModule and the WebSpherePluginConfig directives as models for configuring your file:

  ```
  LoadModule
        was_ap20_module /usr/IBM/HTTPServer/Plugins/bin/mod_was_ap20_http.so
  ```

  Example:

  ```
  WebSpherePluginConfig
        /usr/IBM/HTTPServer/Plugins/config/webserver1/plugin-cfg.xml
  ```

- Linux   Solaris   Configure entries in the `httpd.conf` file.

  Use the following examples of the LoadModule and the WebSpherePluginConfig directives as models for configuring your file:

  ```
  LoadModule
        was_ap20_module /opt/IBM/HTTPServer/Plugins/bin/mod_was_ap20_http.so
  ```

  Example:

  ```
  WebSpherePluginConfig
        /opt/IBM/HTTPServer/Plugins/config/webserver1/plugin-cfg.xml
  ```

- HP-UX   Configure entries in the `httpd.conf` file.

  Use the following examples of the LoadModule and the WebSpherePluginConfig directives as models for configuring your file:

  ```
  LoadModule
        was_ap20_module /opt/IBM/HTTPServer/Plugins/bin/mod_was_ap20_http.sl
  ```

  Example:

  ```
  WebSpherePluginConfig
        /opt/IBM/HTTPServer/Plugins/config/webserver1/plugin-cfg.xml
  ```

- Windows   Configure entries in the `httpd.conf` file.

  Use the following examples of the LoadModule and the WebSpherePluginConfig directives as models for configuring your file:

  ```
  LoadModule was_ap20_module
        drive:\Program Files\IBM\HTTPServer\Plugins\bin\mod_was_ap20_http.dll
  ```

  Example:

  ```
  WebSpherePluginConfig
        C:\Program Files\IBM\HTTPServer\Plugins\config\webserver1\plugin-cfg.xml
  ```

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

The Web Server Plug-ins Configuration Tool configures the web server plug-in.

This topic describes how to configure IBM HTTP Server, Version 7.0. Other procedures in "Editing web server configuration files" on page 10 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 Web Server Plug-ins Configuration Tool installs a dummy `plugin-cfg.xml` file during installation, but this file requires periodic propagation from the real file on the application server machine.

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

Procedure

- **AIX** Configure entries in the `httpd.conf` file.

  Use the following examples of the LoadModule and the WebSpherePluginConfig directives as models for configuring your file:

  ```
  LoadModule
        was_ap22_module /usr/IBM/HTTPServer/Plugins/bin/mod_was_ap22_http.so
  ```

  Example:

  ```
  WebSpherePluginConfig
        /usr/IBM/HTTPServer/Plugins/config/webserver1/plugin-cfg.xml
  ```

- **Linux** **Solaris** Configure entries in the `httpd.conf` file.

  Use the following examples of the LoadModule and the WebSpherePluginConfig directives as models for configuring your file:

  ```
  LoadModule
        was_ap22_module /opt/IBM/HTTPServer/Plugins/bin/mod_was_ap22_http.so
  ```

  Example:

  ```
  WebSpherePluginConfig
        /opt/IBM/HTTPServer/Plugins/config/webserver1/plugin-cfg.xml
  ```

- `HP-UX` Configure entries in the `httpd.conf` file.

  Use the following examples of the LoadModule and the WebSpherePluginConfig directives as models for configuring your file:

  ```
  LoadModule
        was_ap22_module /opt/IBM/HTTPServer/Plugins/bin/mod_was_ap22_http.sl
  ```

  Example:

  ```
  WebSpherePluginConfig
        /opt/IBM/HTTPServer/Plugins/config/webserver1/plugin-cfg.xml
  ```

- `Windows` Configure entries in the `httpd.conf` file.

  Use the following examples of the LoadModule and the WebSpherePluginConfig directives as models for configuring your file:

  ```
  LoadModule was_ap22_module
        drive:\Program Files\IBM\HTTPServer\Plugins\bin\mod_was_ap22_http.dll
  ```

  Example:

  ```
  WebSpherePluginConfig
        C:\Program Files\IBM\HTTPServer\Plugins\config\webserver1\plugin-cfg.xml
  ```

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

The Web Server Plug-ins Configuration Tool configures the web server plug-in.

This topic describes how to configure IBM HTTP Server, Version 8.0. Other procedures in "Editing web server configuration files" on page 10 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 Web Server Plug-ins Configuration Tool installs a dummy `plugin-cfg.xml` file during installation, but this file requires periodic propagation from the real file on the application server machine.

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

Procedure

- AIX Configure entries in the `httpd.conf` file.

  Use the following examples of the LoadModule and the WebSpherePluginConfig directives as models for configuring your file:

```
LoadModule
     was_ap22_module /usr/IBM/HTTPServer/Plugins/bin/mod_was_ap22_http.so
```

  Example:

```
WebSpherePluginConfig
     /usr/IBM/HTTPServer/Plugins/config/webserver1/plugin-cfg.xml
```

- Linux Solaris Configure entries in the `httpd.conf` file.

  Use the following examples of the LoadModule and the WebSpherePluginConfig directives as models for configuring your file:

```
LoadModule
     was_ap22_module /opt/IBM/HTTPServer/Plugins/bin/mod_was_ap22_http.so
```

  Example:

```
WebSpherePluginConfig
     /opt/IBM/HTTPServer/Plugins/config/webserver1/plugin-cfg.xml
```

- HP-UX Configure entries in the `httpd.conf` file.

  Use the following examples of the LoadModule and the WebSpherePluginConfig directives as models for configuring your file:

```
LoadModule
     was_ap22_module /opt/IBM/HTTPServer/Plugins/bin/mod_was_ap22_http.sl
```

  Example:

```
WebSpherePluginConfig
     /opt/IBM/HTTPServer/Plugins/config/webserver1/plugin-cfg.xml
```

- Windows Configure entries in the `httpd.conf` file.

  Use the following examples of the LoadModule and the WebSpherePluginConfig directives as models for configuring your file:

```
LoadModule was_ap22_module
     drive:\Program Files\IBM\HTTPServer\Plugins\bin\mod_was_ap22_http.dll
```

  Example:

```
WebSpherePluginConfig
     C:\Program Files\IBM\HTTPServer\Plugins\config\webserver1\plugin-cfg.xml
```

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 Microsoft Internet Information Services (IIS)

This topic describes manual configuration settings for Internet Information Services (IIS).

Before you begin

The Web Server Plug-ins Configuration Tool configures the web server. This topic describes how to configure the Internet Information Services (IIS) Web Server manually. Other procedures in "Editing web server configuration files" on page 10 describe configuring other supported web servers.

You must have read/write access to the *plugins_root* directory to perform this task.

About this task

Use the following procedure to manually reproduce how the Web Server Plug-ins Configuration Tool configures the Microsoft Internet Information Services Web Server.

Procedure

- Configure IIS Version 6.0.
    1. Start the IIS application and create a new virtual directory for the website instance that you intend to work with WebSphere Application Server. These instructions assume that you are using the Default Web Site.

        Click **Programs > Administrative Tools > Internet Information Services (IIS) Manager** on a Windows Server 2003 Standard Edition system, for example.
    2. Expand the tree on the left until you see **Default Web Site**.

        Right-click **Default Web Site > New > Virtual Directory** to create the directory with a default installation.
    3. Type **sePlugins** in the **Alias** field in the Virtual Directory Alias panel, then click **Next**.
    4. Browse to the *plugins_root*\bin\IIS_*web_server_name* directory in the Path field of the Web Site Content Directory panel , then click **Next**.

        For example, select the C:\Program Files\IBM\WebSphere\Plugins\bin\IIS_webserver1 directory.
    5. Select the appropriate permission check boxes in the Virtual Directory Access Permissions panel.

        Select the **Read** check box and the **Execute (such as ISAPI applications or CGI)** check box, for example.
    6. Click **Next** to add the sePlugins virtual directory to your default website.
    7. Click **Finish** when the success message displays.
    8. Copy the plug-in binaries to the *plugins_root* \bin\IIS_*web_server_name* directory.

        For example. copy the plug-in binary files to the C:\Program Files\IBM\WebSphere\Plugins\bin\IIS_webserver1 directory.

        The plugin-cfg.loc file resides in this directory. The first line of the plugin-cfg.loc file identifies the location of the plugin-cfg.xml file.
    9. Expand the **Web Sites** folder in the left pane navigation tree of the IIS Manager panel.

10. Right-click **Default Web Site** in the navigation tree and click **Properties**.

11. Add the Internet Services Application Programming Interface (ISAPI) filter into the IIS configuration.

    In the Default Web Site Properties panel, perform the following steps:

    a. Click the **ISAPI Filters** tab.

    b. Click **Add** to open the **Add/Edit Filter Properties** dialog window.

    c. Type **iisWASPlugin** in the **Filter name** field.

    d. Click **Browse** to select the `C:\Program Files\IBM\WebSphere\Plugins\bin\IIS_webserver1\`
       `iisWASPlugin_http.dll` file for the value of the **Executable** field.

       Browse to your *plugins_root* `\bin\IIS_`*web_server_name* directory to select the
       iisWASPlugin_http.dll file.

    e. Click **OK** to close the **Add/Edit Filter Properties** dialog window.

    f. Click **OK** to close the **Default Web Site Properties** window.

12. Set the value in the `plugin-cfg.loc` file to the location of the configuration file.

    Set the location to the *plugins_root* `\config\` *webserver_name* `\plugin-cfg.xml` file, which might
    be `C:\Program Files\IBM\WebSphere\Plugins\config\IIS_webserver1\plugin-cfg.xml` 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.

    Example:

    `"C:\IBM\WebSphere\Plugins\config\webserver1\plugin-cfg.xml"`

13. Configure the web server to run WebSphere Application Server extensions:

    a. Expand the left pane navigation tree and click on the **Web Service Extensions** folder in the IIS
       Manager panel.

    b. Click **Add a new web service extension** to open the **New Web Service Extension** dialog
       window.

    c. In the **Extension name** field, type `WASPlugin` as the name of the new web service extension.

    d. Click **Add** to open the **Add file** dialog window.

    e. In the **Path to file** field, type the path or click **Browse** to navigate to the correct
       `iisWASPlugin_http.dll` file that the new web service extension requires, and click **OK**.

    f. Select the **Set extension status to Allowed** check box to automatically set the status of the
       new web service extension to Allowed and click **OK**.

- Configure IIS Version 7.x.

  Note: ⬛ 2008 ⬛ You can use IIS Version 7 with Windows Server 2008 and IIS Version 7.5 with
  Windows Server 2008 R2.

1. Install IIS Version 7.x with the necessary IIS Version 6.0 Management Compatibility components.

   IIS Version 6.0 Management Compatibility components are not automatically installed by default.
   Complete the following steps to install IIS Version 7.x with the necessary IIS Version 6.0
   Management Compatibility components.

   – Complete the following steps to bring up the Server Manager window on Windows Server 2008:

     a. Click **Start > Administrative Tools > Server Managers**.

     b. Click **Action > Add Roles**, and then click **Next**.

     c. On the Select Server Roles page, select the Web Server (IIS) role, and then click **Next**.

     d. If a prompt for the Windows Process Activation Service feature displays, click **Add Feature >
        Next**, and then click **Next** on the IIS introduction page

   – When the Role Services window displays, verify that the following options are selected in addition
     to the default options that are already selected.

     - Internet Information Services: Management Tools

- IIS Version 6.0 Management Compatibility: IIS Version 6.0 Management Console, IIS Version 6.0 Scripting Tools, IIS Version 6.0 WMI Compatibility, and IIS Metabase compatibility
    - Application Development: ISAPI Extensions, ISAPI Filters
  – Click **Next** to enable the selected options, and then click **Install** on the next window to perform the installation.
  – When the installation finishes, click **Close** on the Installation Results window.
2. Install the web server plug-ins.

   If you are using an already installed web server plug-in, go to the next step, and re-configure IIS Version 7.x to use that web server plug-in.
3. Optional: Re-configure IIS Version 7.x if the web server plug-in is already installed:

   The following steps are completed automatically during web server plug-in installation. You only need to complete these steps are if you are re-configuring IIS Version 7.x to use an existing web server plug-in.

   Complete the following steps to re-configure IIS Version 7.x:

   a. Click **Start > All Programs > Administrative Tools > Internet Information Services (IIS) Manager** on a Windows Server 2008 operating system. This action starts the IIS application, and creates a new virtual directory for the website instance that you intend to use with WebSphere Application Server. These instructions assume that you are using the default website.

   b. Expand the tree on the left until you see Default Web Site.

   c. Right-click **Default Web Site > Add Virtual Directory** to create the directory with a default installation.

   d. Type sePlugins in the **Alias** field on the Virtual Directory Alias window.

   e. Browse to the *plugins_root*\bin\IIS_*web_server_name* directory in the **Physical Path** field of the Web Site Content Directory window, and then click **OK**. For example, select the `C:\Program Files\IBM\WebSphere\Plugins\bin\IIS_webserver1` directory.

   f. Click the Test Settings button. If the settings test fails, then either change the permissions of the physical directory, or select **Connect As**, and let IIS connect as a Windows user account that has authority to files in that physical path.

   Attention: When you click the Test Settings button, you might encounter the following warning message if you use the default "Pass-thru authentication" setting:

   ```
   Cannot verify access to path
   ```

   For more information, refer to the Microsoft information on this subject.

   g. Click **OK** to add the sePlugins virtual directory to your default website.

   h. In the navigation tree, select the sePlugins virtual directory that you just created.

   i. On the Features panel, double-click **Handler Mappings**, and then click **Edit Feature Permissions** on the Actions panel.

   j. Select **Script** and **Execute**, if they are not already selected.

   k. Click **OK**.

   l. Manually copy the plug-in binaries to the *plugins_root*\bin\IIS_*web_server_name* directory. For example, copy the plug-in binary files to the `C:\Program Files\IBM\WebSphere\Plugins\bin\IIS_webserver1` directory.

   The plugin-cfg.loc file resides in this directory. The first line of the plugin-cfg.loc file identifies the location of the plugin-cfg.xml file.

   m. Return to the IIS Manager window, and expand the Web Sites folder in the left-hand navigation tree of that window.

   n. Select **Default Web Site** in the navigation tree.

   o. Add the Internet Services Application Programming Interface (ISAPI) filter into the IIS configuration.

On the Default Web Site Properties panel, complete the following steps:

1) Double-click the ISAPI Filters tab.

2) Click to open the Add/Edit Filter Properties dialog window.

3) Type iisWASPlugin in the Filter name field.

4) Click **Browse** to select the plug-in file located in the *plugins_root* `\bin\` `IIS`*web_server_name*\iisWASPlugin_http.dll directory.

5) Click **OK** to close the Add/Edit Filter Properties dialog window.

p. In the navigation tree, select the top level server node.

q. On the Features panel, double-click **ISAPI and CGI Restrictions**, and then, on the Actions panel, click **Add**.

To determine the value to specify for the **ISAPI or CGI Path** property, browse to, and then select the same plug-in file that you selected in the previous step. For example:

`plugins_root\bin\IIS_web_server_name\iisWASPlugin_http.dll`

Then type WASPlugin in the **Description** field, select **Allow extension path to execute**, and then click **OK** to close the **ISAPI and CGI Restrictions** dialog window.

r. Set the value in the plugin-cfg.loc file to the location of the configuration file at *plugins_root* `\config\`*webserver_name*`\plugin-cfg.xml`.

Following is the default location:

`C:\Program Files\IBM\WebSphere\Plugins\config\IIS_webserver1\plugin-cfg.xml`

The location varies depending on how you have configured your system. If the web server, and WebSphere Application Server are on separate machines, you have a remote installation. If the web server, and WebSphere Application Server are on the same machine, then you have a local installation, and the correct location of the configuration file might be set. If the two servers are on the same machine, and the application server is federated, you have a local distributed installation.

Local distributed example:

`C:\IBM\WebSphere\AppServer\profiles\custom01\config\cells\dmgrcell\nodes`
`\managed_node\servers\webserver1\plugin-cfg.xml`

Local example:

`C:\IBM\WebSphere\Plugins\config\webserver1\`
`plugin-cfg.xml`

s. Restart IIS Version 7.x and your WebSphere Application Server profile.

- Enable IIS Version 6.0 or IIS Version 7.x to communicate with a web server plug-in that is running in 32–bit mode.

The web server plug-in for IIS is available in both 32-bit, and 64-bit versions. When using the 32-bit version plug-in on a Microsoft Windows 64-bit operating system, the following steps should be taken to enable the native 64-bit IIS to run the plug-in under a 32-bit worker process.

The Windows Server TechNet topic Running 32-bit Applications on 64-bit Windows describes how to enable the native 64-bit IIS Version 6.0 to run the web server plug-in under a 32-bit worker process.

Complete the following steps to enable the native 64-bit IIS Version 7.x to run the web server plug-in under a 32-bit worker process:

1. Launch the IIS Version 7.x administrative console.

2. On the connections page, expand the **Sites** node, and select the website that is intended for the web server plug-in.

3. On the actions page, click **Basic Settings**, and make a note of the Application Pool name.

4. Click **Cancel** , and then select the **Application Pools** node on the connections page.

5. On the features page, right-click the application pool that you noted in the earlier step, and then choose **Advanced Settings**.

6. Set the **Enable 32-bit Applications** property to `True`.

7.   Click **OK** to complete the configuration change.

8.   Restart the corresponding application pool.

- Optional: Configure multiple websites. Given:
  - There are two websites defined: website1, website2.
  - The DLL files are already created as `bin/website1/iisWASPlugin_http.dll` and `bin/website2/iisWebsite2/iisWASPlugin_http.dll`.
  - The `plugin-cfg.loc` files are created in the same folder as the DLL files.

1.   Run IIS in worker process isolation mode (default).

   To enable worker process in isolation mode:

   a.   Open the IIS Manager console and expand the local computer by clicking the **plus sign**.

   b.   Expand the **Web Sites** folder, then right-click the **Default Web Sites** folder.

   c.   Click **Properties**, then click the **Service** tab.

   d.   Under Isolation mode, clear the Run web service in IIS 5.0 isolation mode check box to enable worker process isolation mode.

2.   Define two application pools; one for website1 and the other for website2. Do not use the pre-defined application pool DefaultAppPool.

3.   Define the two websites, including the filter setting, virtual host setting, and extension settings.

4.   Assign an application pool for each website.

   a.   Under each website folder, right click on the *website name*.

   b.   Click **Property**, and select the **Home Directory** tab. 2.

   c.   At the bottom of the application settings, select the application pool you defined for website 1 from the drop-down list of application pools.

   d.   Click **OK**.

   e.   Repeat the previous steps for the second website and select the application pool you defined for website 2.

5.   Start the IIS service and start each website.

Results

This procedure results in re-configuring the Internet Information Services (IIS) Web Server.

Note: On some editions of the Windows operating system, the `http_plugin.log` file is not created automatically when the plug-in is installed and the IIS Web Server is started. If the `http_plugin.log` file is not created after performing the procedure described above, take the following steps:

   1.   Open a Windows Explorer window.

   2.   Browse to the *plugins_root*`\logs\`*web_server_name* directory.

   3.   Share the folder and give full-control permission to everyone.

What to do next

You can now install applications on the configured Web server. 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 the Sun Java System Web Server

This topic describes how to change configuration settings for the Sun Java System Web Server (formerly Sun ONE Web Server and iPlanet Web Server), Version 6.0 and later.

Before you begin

The Web Server Plug-ins Configuration Tool configures the web server plug-in. This topic describes how to configure the Sun Java System Web Server if you must change something in the existing configuration. Other procedures in "Editing web server configuration files" on page 10 describe configuring other supported web servers.

About this task

Configure the Sun ONE Web Server 6.0 or Sun Java System Web Server, Version 6.1 and later.

Examples and messages are sometimes shown on more than one line for ease of presentation. Verify that each directive in a web server configuration file is on one line.

Procedure

1. Configure entries in the `obj.conf` configuration file and in the `magnus.conf` configuration file for Version 6.0 and later of Sun Java System Web Server.

   a. Add two directives to the `obj.conf` file after the `<Object name=default>` tag:

      ```
      Service fn="as_handler"
      AddLog fn="as_term"
      ```

   b. Add two directives at the end of the `magnus.conf` file:

      The location for the `bootstrap.properties` directive 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.

      gotcha: The following examples reference the libns61_http.so file, which is only supported for the Sun Java System Web Server 7.0 and 6.1. If you are using a Sun ONE Web Server 6.0, change these references to libns41_http.so.

      - AIX   HP-UX   Linux   Solaris

        Example:

        ```
        Init fn="load-modules"
             funcs="as_init,as_handler,as_term"
             shlib="/opt/IBM/WebSphere/Plugins/bin/libns61_http.so"
        Init fn="as_init"
             bootstrap.properties="/opt/IBM/WebSphere/Plugins/config/webserver1/plugin-cfg.xml"
        ```

      - Windows

        Example:

        ```
        Init fn="load-modules"
             funcs="as_init,as_handler,as_term"
             shlib="C:\IBM\WebSphere\Plugins\bin\ns41_http.dll"
        Init fn="as_init"
             bootstrap.properties="C:\IBM\WebSphere\Plugins\config\webserver1\plugin-cfg.xml"
        ```

2. Set the shared library path on HP-UX machines. On some installations of Sun Java System Web Server on an HP-UX machine, it is necessary to manually set the `SHLIB_PATH` variable to `/usr/lib` before starting Sun Java System Web Server with a plug-in that is configured for Secured Sockets Layer (SSL). For example, in the korn shell, issue the following command before invoking the command to start the Sun Java System Web Server:

   ```
   export SHLIB_PATH=/usr/lib:$SHLIB_PATH
   ```

3. Disable the feature of Sun Java System Web Server Version 6.1 that supports servlets and JavaServer Pages files by default. Disable this feature so that the WebSphere Application Server plug-in can handle the requests.

   Perform the following steps to disable the feature:

   a. Remove or comment out the following two lines from the `obj.conf` configuration file:

```
NameTrans fn="ntrans-j2ee" name="j2ee"
Error fn="error-j2ee"
```

b. Remove or comment out the following line from the `magnus.conf` configuration file: `AIX`

`HP-UX` `Linux` `Solaris`

```
Init fn="load-modules"
    shlib="C:/Sun/WebServer6.1/bin/https/bin/j2eeplugin.so"
        shlib_flags="(global|now)"
```

`Windows`

```
Init fn="load-modules"
    shlib="C:\Sun\WebServer6.1\bin\https\bin\j2eeplugin.dll"
        shlib_flags="(global|now)"
```

4. If you are configuring a 64-bit version of the Sun Java System Web Server, modify the loadModule entry in the `magnus.conf` configuration file to point to the *plugin_root*`/bin/64bit/libns61_http.so` file.

Results

This procedure results in editing and re-configuring the Sun Java System Web Server.

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.

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.

The Web Server Plug-ins Configuration Tool creates a web server definition 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.

   To stop server1, open a command window and navigate to the *profile_root*/bin directory. Then issue the following command:

   `./stopServer.sh 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.sh server1`

   When you receive a message that is similar to the following message, the server1 process is running:

   `Server server1 open for e-business; process id is 1719`

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

```
<VirtualHostGroup Name="admin_host">
      <VirtualHost Name="*:9060"/>
      <VirtualHost Name="*:80"/>
      <VirtualHost Name="*:9043"/>
  </VirtualHostGroup>
  ...
  ...
  ...
  <ServerCluster Name="server1_SERVER1HOSTserver1_Cluster">
      <Server LoadBalanceWeight="1" Name="SERVER1HOSTserver1_dmgr">
          <Transport Hostname="SERVER1HOST" Port="9060" Protocol="http"/>
      </Server>

      <PrimaryServers>
          <Server Name="SERVER1HOSTserver1_dmgr"/>
      </PrimaryServers>
  </ServerCluster>
  ...
  ...
  ...
  <UriGroup Name="admin_host_server1_SERVER1HOSTserver1_Cluster_URIs">
      <Uri AffinityCookie="JSESSIONID"
          AffinityURLIdentifier="jsessionid" Name="/ibm/console/*"/>
  </UriGroup>
  <Route ServerCluster="server1_SERVER1HOSTserver1_Cluster"
      UriGroup="admin_host_server1_SERVER1HOSTserver1_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.

After installing a supported web server, you must install a binary plug-in module for the web server by installing the Web Server Plug-ins. The plug-in module lets the web server communicate with the application server. The Web Server Plug-ins Configuration Tool allows you to configure the web server and create a web server definition in the configuration of the application server. The Web Server Plug-ins Configuration Tool 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 **default (temporary) plug-in configuration file, `plugin-cfg.xml`**, on the web server machine.
- The **`configure`_`web_server_name`_ script** that you copy from the web server machine to the application server machine.

See the following descriptions of each file.

Web server configuration file

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

The Web Server Plug-ins Configuration Tool must re-configure the configuration file for a supported 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.

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.

The standalone application server regenerates the `plugin-cfg.xml` file in the _profile_root_/`config/cells/`_cell_name_/`nodes/`_web_server_name_`_node/servers/`_web_server_name_ directory. Regeneration occurs whenever a change occurs in the application server configuration that affects deployed applications.

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.

The web server plug-in configuration service automatically regenerates the `plugin-cfg.xml` file after certain events that change the configuration. The configuration service automatically propagates the `plugin-cfg.xml` file to an IBM HTTP Server machine when the file is regenerated. You must manually copy the file on other web servers.

Default plug-in configuration file, `plugin-cfg.xml`

The Web Server Plug-ins Configuration Tool creates the temporary `plugin-cfg.xml` file in the _plugins_root_/`config/`_web_server_name_ directory. The tool creates the file for every remote installation scenario.

The default file is a placeholder that you must replace with the `plugin-cfg.xml` file from the web server definition on the application server. The default file is a replica of the file that the application server creates for a default standalone application server.

Run the `configureweb_server_name` script from the *app_server_root*/`bin` directory of the application server machine for a remote installation or directly from the *plugins_root*/`bin` directory for a local installation. The script creates the web server definition in the configuration files of the default profile. To configure a different profile than the default, edit the `configureweb_server_name` script. Use the -profileName parameter to identify a profile other than the default profile.

After the web server definition is created, the web server plug-in configuration service within the application server creates the first `plugin-cfg.xml` file in the web server definition on the application server machine. If you install an application, create a virtual host, or do anything that changes the configuration, you must propagate the updated `plugin-cfg.xml` file from the application server machine to the web server machine to replace the default file.

Configure*web_server_name* script for the web server definition

The Web Server Plug-ins Configuration Tool creates the `configureweb_server_name` script on the web server machine in the *plugins_root*/`bin` directory. If one machine in a remote scenario is running under an operating system like AIX or Linux and the other machine is running under Windows, use the script created in the *plugins_root*/`bin/crossPlatformScripts` 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 on a remote application server machine. You do not have to copy the script on a local installation. Run the script to create a web server definition in the configuration of the application server.

When using the IBM HTTP Server, configure the IBM HTTP Administration Server also. The IBM HTTP Administration Server works with the administrative console to manage web server definitions. Also, 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

Important: Always open a new command window before running this script. You can avoid a potential problem by doing so.

The problem is a potential conflict between a shell environment variable, the WAS_USER_SCRIPT environment variable, and the actual default profile. The script always works against the default profile. If the WAS_USER_SCRIPT environment variable is set, however, a conflict arises as the script attempts to work on the profile identified by the variable.

The variable is easy to set accidentally. Issue any command from the *profile_root*/`bin` directory of any profile and the variable is set to that profile.

If you have more than one profile on your system, the potential exists that the default profile and the profile identified by the variable are different profiles. If so, a conflict occurs and the script might not create the web server definition in the correct profile, or might not create the web server definition at all.

Reset the variable in either of two ways:

- Close the command window where the variable is set and open a new one.
- Change directories to the *profile_root*/`bin` directory of the default profile and source the `setupCmdLine.sh` script:

Windows

1. Open a command prompt window.
2. Change directories to the *app_server_root*\`bin` directory.
3. Issue the `setupCmdLine.bat` command.

AIX **HP-UX** **Linux** **Solaris**

1. Open a command shell window.
2. Change directories to the *app_server_root*/`bin` directory.
3. Issue the `. ./setupCmdLine.sh` command. Notice the space between the periods. The special format for this command sources the command to make the setting active for all processes started from the command shell.

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
```

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.

The application server must have the following values in the actual `plugin-cfg.xml` file. If so, the default file can successfully configure the binary plug-in module. Then, the plug-in module can successfully communicate with the web server and the application server.

Suppose that the application server does not have the following values in the actual `plugin-cfg.xml` file. In that case, the default file configures the binary plug-in module incorrectly. The plug-in module can always communicate with the web server. But with an improper configuration file, the plug-in module cannot communicate successfully with the application server.

The following are fixed parameter values in the temporary plug-in configuration file.

- **Virtual host name**

  Default value: default_host

This virtual host is configured to serve the DefaultApplication. This value is probably the same as the value in the real `plugin-cfg.xml` file. However, suppose that you create another virtual host for serving applications and install the DefaultApplication on it. If so, the actual `plugin-cfg.xml` file is regenerated. The web server cannot access the DefaultApplication. (The application includes the snoop servlet and the hitcount servlet.)

To access applications on the new virtual host, propagate the real `plugin-cfg.xml` file. Propagation is copying the updated file from the application server machine to the web server machine.

- **HTTP transport port**

  Default value: 9080

  The `9080` value is the default value for the HTTP transport port for the default_host virtual host. This value is probably the same as the value in the updated file. However, this value changes for every profile on the application server machine. The HTTP transport port value must be unique for every application server.

  To communicate over a different port, propagate the real `plugin-cfg.xml` file.

- **Web server listening port**

  Default value: 80

  The `80` value is the default value for the port that controls communication with the web server. However, each application server profile must have a unique port value to communicate to a web server. The actual port value might be 81 or another number.

  To communicate over a different port, propagate the real `plugin-cfg.xml` file.

- **HTTPS transport port**

  Default value: 9443

  The `9443` value is the default value for the HTTPS (secure) transport port for the default_host virtual host. This value is probably the same as the value in the updated file. However, this value changes for every profile on the application server machine. The HTTPS transport port value must be unique for every application server.

  To communicate over a different secure port, propagate the real `plugin-cfg.xml` file.

- **Applications installed on the server1 application server**

  All of the default servlets and applications are included in the default file.

  To serve an application that you developed with the web server, propagate the real `plugin-cfg.xml` file.

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.

Button	Resulting Action
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 3. Status indicators.   The following table describes the status indicators.

⬦	**Started**	The web server is running.
✖	**Stopped**	The web server is not running.
⦿	**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.

Important: You can also determine the IBM HTTP Server version using the versionInfo command.

Procedure

1. Change the directory to the installation root of the Web server.

   For example, it is `/opt/IBM/HTTPServer` on a Solaris machine.
2. Find the subdirectory that contains the executable. The executable for IBM HTTP Server is:
   - **Windows** httpd.exe (the previous command, apache.exe is deprecated)
   - **Linux** httpd
   - **AIX** **HP-UX** **Solaris** apachectl
3. Issue the command with the -v option to display the version information.

   Windows
   ```
   httpd.exe -v
   ```
   Linux
   ```
   ./httpd -v
   ```
   AIX **HP-UX** **Solaris**
   ```
   ./apachectl -v
   ```

Results

The version is shown in the "Server version" field and will look something like the following:
```
IBM_HTTP_Server/7.0.0.0 (Windows)
Server built: Jul 31 2008 08:41:58
```

or
```
Server version: IBM_HTTP_Server/7.0.0.0 (Unix)
Server built: Jul 31 2008 08:41:58
```

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).
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.

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.

The Web Server Plug-ins Configuration Tool configures the web server for communicating with the application server and creates a web server configuration definition in the application server if possible.

Some topologies, such as the web server on one system and the application server on another system, prevent the Web Server Plug-ins Configuration Tool from creating the web server definition in the application server configuration directly on the remote system. In such a case, the Web Server Plug-ins Configuration Tool creates a script that you can copy to the application server system. Run the script to create the web server configuration definition within the application server configuration.

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.

The Web Server Plug-ins Configuration Tool configures the web server and the application server to allow communication between the servers.

Tip: As an alternative to using the Web Server Plug-ins Configuration Tool, you can use the pct command-line tool with a response file to configure a web server. Read "Configuring a web server plug-in using the WCT command-line utility" on page 109 for more information.

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.

When multiple profiles exist, you can select the profile that the Web Server Plug-ins Configuration Tool configures. See "Plug-ins configuration" on page 78 for a description of the flow of logic that determines how to select the profile to configure.

Procedure

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

  See "Configuring a web server and an application server profile on the same machine" on page 83 for the procedure that explains how to create this web server topology for an application server profile.

- **Scenario 2: Remote** The application server and the web server are on separate machines or logical partitions.

  See "Configuring a web server and an application server on separate machines (remote)" on page 89 for the procedure that explains how to create this web server topology.

- **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.
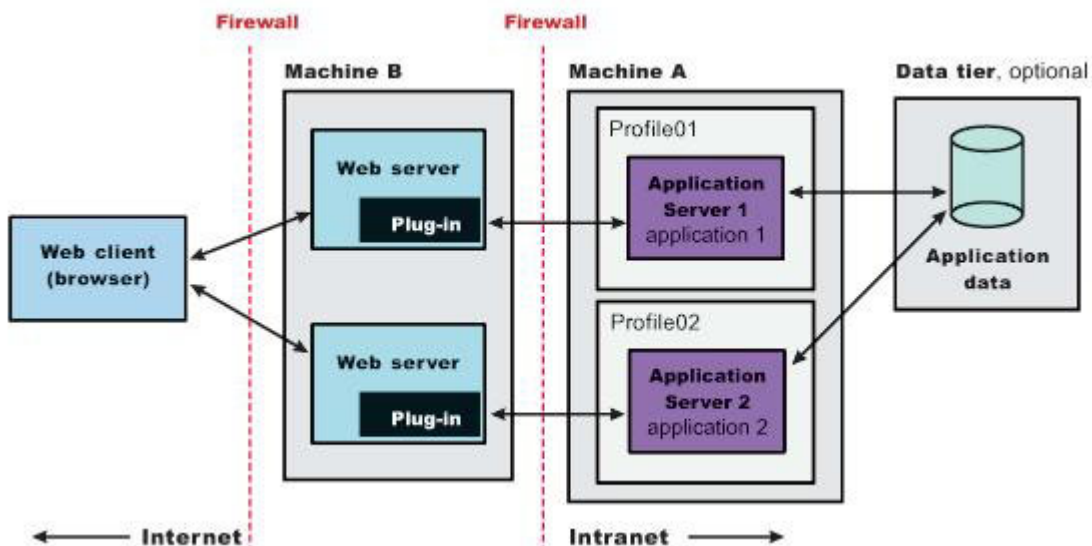
  See "Configuring multiple web servers and remote standalone application servers" on page 96 for the procedure that explains how to create this web server topology.

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.

See "Web server configuration" on page 34 for more information about the files involved in configuring a web server.

See Editing Web server configuration files for information about how the Web Server Plug-ins Configuration Tool configures supported web servers.

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.

About this task

The Web Server Plug-ins Configuration Tool has three main tasks:
* Configures the web server configuration file on the web server machine to point to the binary plug-in module and to the XML configuration file for the binary module.
* Installs a temporary XML configuration file for the binary module (`plugin-cfg.xml`) on the web server machine in remote scenarios.
* Creates the configuration for a web server definition on the application server machine. The Web Server Plug-ins Configuration Tool processes the creation of the web server definition differently depending on the scenario:
  - Recommended remote standalone application server installation:

    Creates a configuration script that you run on the application server machine. Install the web server and configure its plug-in on a different machine than the application server. This configuration is recommended for a production environment.
  - Local standalone application server installation:

    Configures the default profile on a local application server machine and creates the Web server definition for it directly. Install the web server and configure its plug-in on the same machine with the application server. This configuration is for development and test environments.

Tip: As an alternative to using the Web Server Plug-ins Configuration Tool, you can use the pct command-line tool with a response file to configure a web server. Read "Configuring a web server plug-in using the WCT command-line utility" on page 109 for more information.
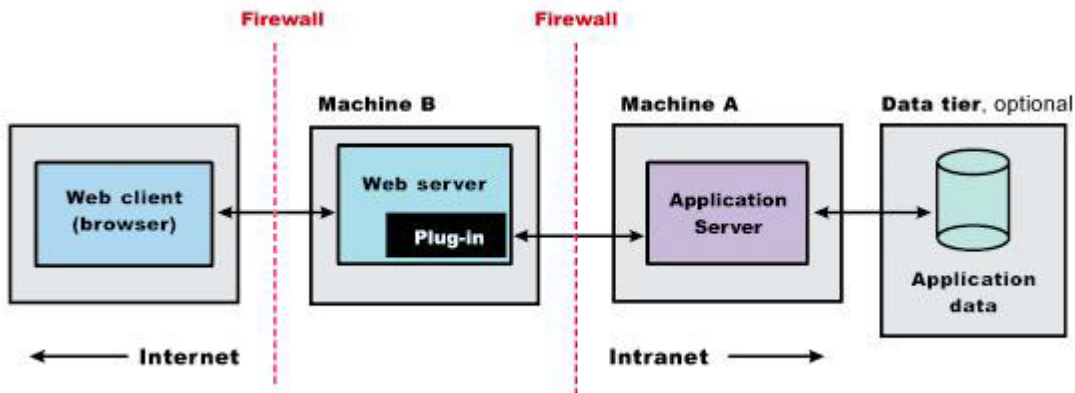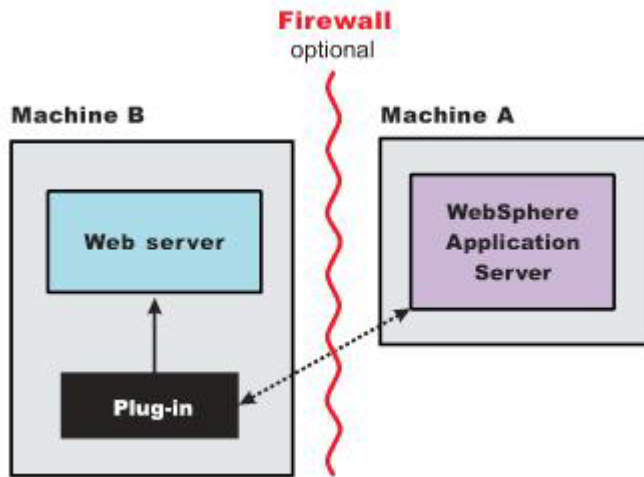
Procedure
* **Set up a remote web server installation.**

  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 4. Installation and configuration.   Remote installation scenario

Step	Machine	Task
1	A	Install Installation Manager.
2	A	Use Installation Manager to install the WebSphere Application Server product.
3	A	Create a standalone application server profile.
4	B	Install Installation Manager.
5	B	Use Installation Manager to install the following: • Web Server Plug-ins for WebSphere Application Server • WebSphere Customization Toolbox
6	B	Use Installation Manager to install IBM HTTP Server, or install another supported web server.
7	B	Open the WebSphere Customization Toolbox, and launch the Web Server Plug-ins Configuration Tool to configure the web server plug-in. The script for creating and configuring the web server is created under the *plugins_root*/`bin` directory.
8	B	Copy the `configureweb_server_name` script to paste on Machine A. If one machine is running under an operating system such as AIX or Linux and the other machine is running under Windows, copy the script from the *plugins_root*/`bin/crossPlatformScripts` directory.
9	A	Paste the `configureweb_server_name` script from Machine B to the *app_server_root*/`bin` directory on Machine A.
10	A	Start the application server.
11	A	Run the script from a command line.
12	A	Verify that the application server is running. Open the administrative console and save the changed configuration.
13	B	Start the web server. `AIX` `HP-UX` `Linux` `Solaris` Source the *plugins_root*/`setupPluginCfg.sh` script for a Domino Web Server before starting a Domino Web Server.
14	B	Run the Snoop servlet. Access the following URL in your browser: `http://host_name_of_machine_B:http_transport_port/Snoop` To verify with your own application, regenerate and propagate the `plugin-cfg.xml` file after installing the application.

Regeneration of the `plugin-cfg.xml` file

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

During configuration, the temporary `plugin-cfg.xml` file is installed on Machine B in the *plugins_root*/`config`/*web_server_name* directory. To use the actual `plugin-cfg.xml` file from the application server, propagate the `plugin-cfg.xml` file as described in the next section.

Propagation of the `plugin-cfg.xml` file

The web server plug-in configuration service propagates the `plugin-cfg.xml` file automatically for IBM HTTP Server Version 6.0 or later. For all other web servers, propagate the plug-in configuration file manually. Copy the `plugin-cfg.xml` file from the *profile_root*/ `config/cells`/*cell_name*/`nodes`/ *web_server_name*_`node/servers`/*web_server_name* directory on Machine A. Paste the file into the *plugins_root*/`config`/*web_server_name* directory on Machine B.

- **Set up a local web server configuration.**

  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 5. Installation and configuration. Local installation scenario

Step	Machine	Task
1	A	Install Installation Manager.
2	A	Use Installation Manager to install the following: • WebSphere Application Server product • Web Server Plug-ins for WebSphere Application Server • WebSphere Customization Toolbox
3	A	Use Installation Manager to install IBM HTTP Server, or install another supported web server.
4	A	Create a standalone application server profile.
5	A	Open the WebSphere Customization Toolbox, and launch the Web Server Plug-ins Configuration Tool to configure the web server plug-in and create the web server definition. The web server definition is automatically created and configured during the configuration of the plug-in.
6	A	Start the application server.
7	A	Verify that the application server is running. Open the administrative console and save the changed configuration.
8	A	Start the web server. AIX HP-UX Linux Solaris Run the *plugins_root*/`setupPluginCfg.sh` script for a Domino Web Server before starting a Domino Web Server.

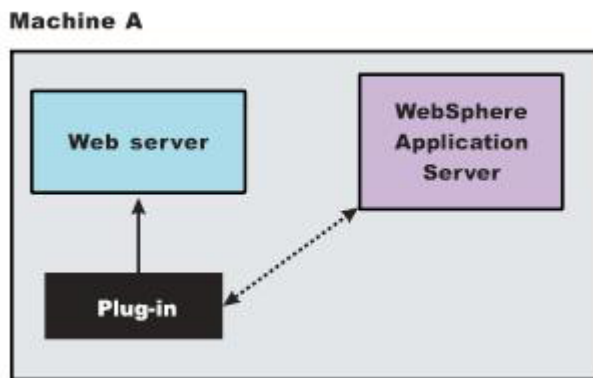| Step	Machine	Task
9	A	Run the Snoop servlet.
		Access the following URL in your browser:
		`http://host_name_of_machine_A:http_transport_port/Snoop`
		To verify with your own application, regenerate and propagate the `plugin-cfg.xml` file after installing the application.

Regeneration of the `plugin-cfg.xml` file

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

The `plugin-cfg.xml` file is generated in the *profile_root*/`config/cells/`*cell_name*/`nodes/`
web_server_name`_node/servers/`*web_server_name* directory. The generation occurs when the web server definition is created.

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 "Web server configuration" on page 34 for more information about the files involved in configuring a web server.

See "Plug-ins configuration" on page 78 for information about the logic behind the processing scenarios for the Web Server Plug-ins Configuration Tool.

See "Editing web server configuration files" on page 10 for information about how the Web Server Plug-ins Configuration Tool configures supported web servers.

See "Installing and configuring web server plug-ins" on page 50 for information about other installation scenarios for installing Web Server Plug-ins.

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

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

Before you begin

Note: The Web Server Plug-ins for WebSphere Application Server Version 8.0 are now installed by Installation Manager rather than by the programs based on InstallShield MultiPlatform (ISMP) that are used to install, update, and uninstall previous versions.

Restrictions:

- Solaris The Installation Manager GUI is not supported on Solaris 10 x64 systems. Perform the following actions to install or uninstall the product on these systems:
  – Use the Installation Manager GUI on a supported system to record a response file that will allow you to install or uninstall the product silently.
  – Edit the recorded response file if necessary.

– Use the response file to install or uninstall the product silently on your system.

- `Linux` For any Linux system that is enabled for Security Enhanced Linux (SELinux), such as Red Hat Enterprise Linux Version 5 or SUSE Linux Enterprise Server Version 11, you must identify the Java shared libraries in the Installation Manager Version 1.4.3 or earlier installation image to the system. Also, you must identify the Java shared libraries in the Installation Manager Version 1.4.3 or earlier installation after it has been installed. For example:

```
chcon -R -t texrel_shlib_t ${IM_Image}/jre_5.0.3.sr8a_20080811b/jre/bin
chcon -R -t texrel_shlib_t ${IM_Install_root}/eclipse/jre_5.0.3.sr8a_20080811b/jre/bin
```

- Installation Manager console mode, which is included in Installation Manager Version 1.4.3 and later, does not work with WebSphere Application Server Version 8.0 offerings.

About this task

Perform one of these procedures to install, update, roll back, or uninstall the Web Server Plug-ins using Installation Manager.

Procedure

Results

Notes on logging and tracing:

- An easy way to view the logs is to open Installation Manager and go to **File > View Log**. An individual log file can be opened by selecting it in the table and then clicking the **Open log file** icon.
- Logs are located in the `logs` directory of Installation Manager's application data location. For example:
  - `Windows` **Administrative installation:**

```
C:\Documents and Settings\All Users\Application Data\IBM\Installation Manager\logs
```

  - `Windows` **Non-administrative installation:**

```
C:\Documents and Settings\user_name\Application Data\IBM\Installation Manager\logs
```

  - `AIX` `HP-UX` `Linux` `Solaris` **Administrative installation:**

```
/var/ibm/InstallationManager/logs
```

  - `AIX` `HP-UX` `Linux` `Solaris` **Non-administrative installation:**

```
user_home/var/ibm/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.
- The `log.properties` file in the `logs` directory specifies the level of logging or tracing that Installation Manager uses.

Notes on troubleshooting:

- `HP-UX` By default, some HP-UX systems are configured to not use DNS to resolve host names. This could result in Installation Manager not being able to connect to an external repository.

You can ping the repository, but nslookup does not return anything.

Work with your system administrator to configure your machine to use DNS, or use the IP address of the repository.

- Installation Manager might display a warning message during the uninstallation process.

  Uninstalling the Web Server Plug-ins using Installation Manager requires that the data repositories remain valid and available.

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

  Read the release notes to learn more about the latest version of Installation Manager. To access the release notes, complete the following task:

  – `Windows` Click **Start > Programs > IBM Installation Manager > Release Notes**.

  – `AIX` `HP-UX` `Linux` `Solaris` Go to the documentation subdirectory in the directory where Installation Manager is installed, and open the `readme.html` file.

Installing the Web Server Plug-ins using the GUI

You can use the Installation Manager GUI to install the Web Server Plug-ins.

Before you begin

Install Installation Manager:

1. Perform one of the following procedures:

   - If you want to use the Installation Manager that is included with this product, perform the following actions:

     a. 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.

          1) 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.

          2) 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 files from the Passport Advantage site.

          2) 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.

          3) 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.

          1) 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.

2) 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.

Notes:

– 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.

– With the Packaging Utility, you can create and manage packages for installation repositories. You can copy multiple packages into one repository or copy multiple disks for one product into a repository. You can copy packages from Passport Advantage into a repository for example. For more information on the Packaging Utility, go to the IBM Installation Manager Information Center.

b. Change to the location containing the Installation Manager installation files, and run one of the following commands:

Administrative installation:

– **Windows** `install.exe`

– **AIX** **HP-UX** **Linux** **Solaris** `./install`

Non-administrative installation:

– **Windows** `userinst.exe`

– **AIX** **HP-UX** **Linux** **Solaris** `./userinst`

Group-mode installation:

AIX **HP-UX** **Linux** **Solaris** `./groupinst`

Notes on group mode:

– Group mode allows multiple users to use a single instance of IBM Installation Manager to manage software packages.

– **Windows** Group mode is not available on Windows operating systems.

– If you do not install Installation Manager using group mode, you will not be able to use group mode to manage any of the products that you install later using this Installation Manager.

– Make sure that you change the installation location from the default location in the current user's home directory to a location that is accessible by all users in the group.

– Set up your groups, permissions, and environment variables as described in the Group mode road maps in the IBM Installation Manager Information Center before installing in group mode.

– For more information on using group mode, read the Group mode road maps in the IBM Installation Manager Information Center.

The installer opens an **Install Packages** window.

c. Make sure that the Installation Manager package is selected, and click **Next**.

d. Accept the terms in the license agreements, and click **Next**.

The program creates the directory for your installation.

  e. Click **Next**.

  f. Review the summary information, and click **Install**.

    – If the installation is successful, the program displays a message indicating that installation is successful.

    – If the installation is not successful, click **View Log File** to troubleshoot the problem.

- 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.

    a. Download the product repositories from the Passport Advantage site.

    b. 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.

  Notes:

    – 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.

    – With the Packaging Utility, you can create and manage packages for installation repositories. You can copy multiple packages into one repository or copy multiple disks for one product into a repository. You can copy packages from Passport Advantage into a repository for example. For more information on the Packaging Utility, go to the IBM Installation Manager Information Center.

2. Add the product repository to your Installation Manager preferences.

  a. Start Installation Manager.

  b. In the top menu, click **File > Preferences**.

  c. Select **Repositories**.

  d. Perform the following actions:

    1) Click **Add Repository**.

    2) Enter the path to the `repository.config` file in the location containing the repository files.

      For example:

      - **Windows** `C:\repositories\`*product_name*`\local-repositories`

      - **AIX** **HP-UX** **Linux** **Solaris** `/var/repositories/`*product_name*`/local-repositories`

      or

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

    3) Click **OK**.

e. Deselect any locations listed in the Repositories window that you will not be using.

f. Click **Apply**.

g. Click **OK**.

h. Click **File > Exit** to close Installation Manager.

About this task

Perform this procedure to use the Installation Manager GUI to install the Web Server Plug-ins.

Procedure

1. Start Installation Manager.

   Tip: [AIX] [HP-UX] [Linux] [Solaris] You can start Installation Manager in group mode with the ./IBMIM command.
   - Group mode allows users to share packages in a common location and manage them with the same instance of Installation Manager.
   - For more information on using group mode, read the Group mode road maps in the IBM Installation Manager Information Center.

2. Click **Install**.

   Note: If you are prompted to authenticate, use the IBM ID and password that you registered with on the program website.

   Installation Manager searches its defined repositories for available packages.

3. Perform the following actions.

   a. Select **Web Server Plug-ins for IBM WebSphere Application Server** and the appropriate version.

      Note: If you are installing the ILAN version of this product, select **Web Server Plug-ins for IBM WebSphere Application Server (ILAN)**.

      If you already have the Web Server Plug-ins installed on your system, a message displays indicating that the Web Server Plug-ins are already installed. To create another installation of the Web Server Plug-ins in another location, click **Continue**.

      Tip: If the **Search service repositories during installation and updates** option is selected on the Installation Manager Repository preference page and you are connected to the Internet, you can click **Check for Other Versions and Extensions** to search for updates in the default update repositories for the selected packages. In this case, you do not need to add the specific service-repository URL to the Installation Manager Repository preference page.

   b. Select the fixes to install.

      Any recommended fixes are selected by default.

      If there are recommended fixes, you can select the option to show only recommended fixes and hide non-recommended fixes.

   c. Click **Next**.

   Note: If you try to install a newer level of the Web Server Plug-ins with a previous version of Installation Manager, Installation Manager might prompt you to update to the latest level of Installation Manager when it connects to the repository. Update to the newer version before you continue if you are prompted to do so. Read Installing updates in the Installation Manager information center for information about automatic updates.

4. Accept the terms in the license agreements, and click **Next**.

5. Specify the installation root directory for the product binaries, which are also referred to as the core product files or system files.

The panel also displays the shared resources directory and disk-space information.

Restrictions:

- Deleting the default target location and leaving an installation-directory field empty prevents you from continuing.
- Do not use symbolic links as the destination directory.

  Symbolic links are not supported.
- Do not use a semicolon in the directory name.

  The Web Server Plug-ins cannot install properly if the target directory includes a semicolon.

  `Windows` A semicolon is the character used to construct the class path on Windows systems.
- `Windows` The maximum path length on the Windows Server 2008, Windows Vista, and Windows 7 operating systems is 60 characters.

6. Click **Next**.
7. If you are installing on a 64-bit system, choose between a 32-bit and 64-bit IBM Runtime Environment for Java and click **Next**.

   Notes:

   - This option displays only if you are installing on a 64-bit system.
   - You must select one of the two options.
   - You cannot modify this installation later and change this selection.

8. Review the summary information, and click **Install**.
   - If the installation is successful, the program displays a message indicating that installation is successful.

     Note: The program might also display important post-installation instructions as well.
   - If the installation is not successful, click **View Log File** to troubleshoot the problem.

9. Click **Finish**.
10. Click **File > Exit** to close Installation Manager.

Installing the Web Server Plug-ins silently

You can use Installation Manager to install the Web Server Plug-ins silently.

Before you begin

Install Installation Manager on each of the systems onto which you want to install the product.

1. Perform one of the following procedures:
   - If you want to use the Installation Manager that is included with this product, perform the following actions:
     a. 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.
          1) 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.
          2) 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 files from the Passport Advantage site.

2) 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.

3) 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.

1) 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.

2) 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.

Notes:

– 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.

– With the Packaging Utility, you can create and manage packages for installation repositories. You can copy multiple packages into one repository or copy multiple disks for one product into a repository. You can copy packages from Passport Advantage into a repository for example. For more information on the Packaging Utility, go to the IBM Installation Manager Information Center.

b. Change to the location containing the Installation Manager installation files, and run one of the following commands to install Installation Manager silently:

Administrative installation:

– **Windows** `installc.exe -acceptLicense -log` *log_file_path_and_name*

– **AIX** **HP-UX** **Linux** **Solaris** `./installc -acceptLicense -log` *log_file_path_and_name*

Non-administrative installation:

– **Windows** `userinstc.exe -acceptLicense -log` *log_file_path_and_name*

– **AIX** **HP-UX** **Linux** **Solaris** `./userinstc -acceptLicense -log` *log_file_path_and_name*

Group-mode installation:

AIX **HP-UX** **Linux** **Solaris** `./groupinstc -acceptLicense -dataLocation` *application_data_location* `-log` *log_file_path_and_name*

Notes on group mode:

– Group mode allows multiple users to use a single instance of IBM Installation Manager to manage software packages.

– **Windows** Group mode is not available on Windows operating systems.

- If you do not install Installation Manager using group mode, you will not be able to use group mode to manage any of the products that you install later using this Installation Manager.
- Make sure that you change the installation location from the default location in the current user's home directory to a location that is accessible by all users in the group.
- Set up your groups, permissions, and environment variables as described in the Group mode road maps in the IBM Installation Manager Information Center before installing in group mode.
- For more information on using group mode, read the Group mode road maps in the IBM Installation Manager Information Center.

- 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.

  a. Download the product repositories from the Passport Advantage site.

  b. 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.

  Notes:

  - 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.
  - With the Packaging Utility, you can create and manage packages for installation repositories. You can copy multiple packages into one repository or copy multiple disks for one product into a repository. You can copy packages from Passport Advantage into a repository for example. For more information on the Packaging Utility, go to the IBM Installation Manager Information Center.

2. Add the product repository to your Installation Manager preferences.

   a. Start Installation Manager.

   b. In the top menu, click **File > Preferences**.

   c. Select **Repositories**.

   d. Perform the following actions:

      1) Click **Add Repository**.

      2) Enter the path to the `repository.config` file in the location containing the repository files.

         For example:

- `Windows`   `C:\repositories\`*`product_name`*`\local-repositories`
- `AIX`   `HP-UX`   `Linux`   `Solaris`   `/var/repositories/`*`product_name`*`/local-repositories`

      or

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

     3) Click **OK**.

  e. Deselect any locations listed in the Repositories window that you will not be using.

  f. Click **Apply**.

  g. Click **OK**.

  h. Click **File > Exit** to close Installation Manager.

About this task

Using Installation Manager, you can work with response files to install the Web Server Plug-ins silently in a variety of ways. You can record a response file using the GUI as described in the following procedure, or you can generate a new response file by hand or by taking an example and modifying it.

Procedure

1. Optional: **Record a response file to install the Web Server Plug-ins:** On one of your systems, perform the following actions to record a response file that will install the Web Server Plug-ins.

  a. From a command line, change to the eclipse subdirectory in the directory where you installed Installation Manager.

  b. Start Installation Manager from the command line using the -record option.

    For example:

    - `Windows` **Administrator or non-administrator:**

```
IBMIM.exe -skipInstall "C:\temp\imRegistry"
  -record C:\temp\install_response_file.xml
```

    - `AIX`   `HP-UX`   `Linux`   `Solaris`   **Administrator:**

```
./IBMIM -skipInstall /var/temp/imRegistry
  -record /var/temp/install_response_file.xml
```

    - `AIX`   `HP-UX`   `Linux`   `Solaris`   **Non-administrator:**

```
./IBMIM -skipInstall user_home/var/temp/imRegistry
  -record user_home/var/temp/install_response_file.xml
```

    Tip: When you record a new response file, you can specify the -skipInstall parameter. Using this parameter has the following benefits:

      - No files are actually installed, and this speeds up the recording.
      - If you use a temporary data location with the -skipInstall parameter, Installation Manager writes the installation registry to the specified data location while recording. When you start Installation Manager again without the -skipInstall parameter, you then can use your response file to install against the real installation registry.

      The -skipInstall operation should not be used on the actual agent data location used by Installation Manager. This is unsupported. Use a clean writable location, and re-use that location for future recording sessions.

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

  c. Add the appropriate repositories to your Installation Manager preferences.

    1) In the top menu, click **File > Preferences**.

    2) Select **Repositories**.

    3) Perform the following actions for each repository:

      a) Click **Add Repository**.

b) Enter the path to the `repository.config` file in the remote web-based repository or the local directory into which you unpacked the repository files.

For example:

- Remote repositories:

`https://downloads.mycorp.com:8080/WAS_80_repository`

or

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

- Local repositories:

  - **Windows** `C:\repositories\plugins\local-repositories`
  - **AIX** **HP-UX** **Linux** **Solaris** `/var/repositories/plugins/local-repositories`

c) Click **OK**.

4) Click **Apply**.

5) Click **OK**.

d. Click **Install**.

Note: If you are prompted to authenticate, use the IBM ID and password that you registered with on the program website.

Installation Manager searches its defined repositories for available packages.

e. Perform the following actions.

1) Select **Web Server Plug-ins for IBM WebSphere Application Server** and the appropriate version.

Note: If you are installing the ILAN version of this product, select **Web Server Plug-ins for IBM WebSphere Application Server (ILAN)**.

If you already have the Web Server Plug-ins installed on your system, a message displays indicating that the Web Server Plug-ins are already installed. To create another installation of the Web Server Plug-ins in another location, click **Continue**.

2) Click **Next**.

f. Accept the terms in the license agreements, and click **Next**.

g. Specify the installation root directory for the Web Server Plug-ins binaries, which are also referred to as the core product files or system files.

The panel also displays the shared resources directory and disk-space information.

Restrictions:

- Deleting the default target location and leaving an installation-directory field empty prevents you from continuing.
- Do not use symbolic links as the destination directory.

  Symbolic links are not supported.

- Do not use a semicolon in the directory name.

  The Web Server Plug-ins cannot install properly if the target directory includes a semicolon.

  Windows A semicolon is the character used to construct the class path on Windows systems.

- **Windows** The maximum path length on the Windows Server 2008, Windows Vista, and Windows 7 operating systems is 60 characters.

h. Click **Next**.

      i. If you are installing on a 64-bit system, choose between a 32-bit and 64-bit IBM Runtime Environment for Java and click **Next**.

        Notes:

            • This option displays only if you are installing on a 64-bit system.

            • You must select one of the two options.

            • You cannot modify this installation later and change this selection.

      j. Review the summary information, and click **Install**.

         • If the installation is successful, the program displays a message indicating that installation is successful.

          Note: The program might also display important post-installation instructions as well.

         • If the installation is not successful, click **View Log File** to troubleshoot the problem.

      k. Click **Finish**.

      l. Click **File > Exit** to close Installation Manager.

      m. Optional: If you are using an authenticated remote repository, create a keyring file for silent installation.

        1) From a command line, change to the eclipse subdirectory in the directory where you installed Installation Manager.

        2) Start Installation Manager from the command line using the -record option.

           For example:

            • **Windows** **Administrator or non-administrator:**

```
IBMIM.exe -skipInstall "C:\temp\imRegistry"
  -keyring C:\IM\im.keyring
  -record C:\temp\keyring_response_file.xml
```

            • **AIX** **HP-UX** **Linux** **Solaris** **Administrator:**

```
./IBMIM -skipInstall /var/temp/imRegistry
  -keyring /var/IM/im.keyring
  -record /var/temp/keyring_response_file.xml
```

            • **AIX** **HP-UX** **Linux** **Solaris** **Non-administrator:**

```
./IBMIM -skipInstall user_home/var/temp/imRegistry
  -keyring user_home/var/IM/im.keyring
  -record user_home/var/temp/keyring_response_file.xml
```

        3) When a window opens that requests your credentials for the authenticated remote repository, enter the correct credentials and **save** them.

        4) Click **File > Exit** to close Installation Manager.

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

2. **Use the response files to install the Web Server Plug-ins silently:**

      a. Optional: **Use the response file to install the keyring silently:** Go to a command line on each of the systems on which you want to install the product, change to the `eclipse/tools` subdirectory in the directory where you installed Installation Manager, and install the keyring silently.

        For example:

         • **Windows** **Administrator or non-administrator:**

```
imcl.exe -acceptLicense
  input C:\temp\keyring_response_file.xml
  -log C:\temp\keyring_log.xml
```

         • **AIX** **HP-UX** **Linux** **Solaris** **Administrator:**

```
./imcl -acceptLicense
  input /var/temp/keyring_response_file.xml
  -log /var/temp/keyring_log.xml
```

         • **AIX** **HP-UX** **Linux** **Solaris** **Non-administrator:**

```
./imcl -acceptLicense
  input user_home/var/temp/keyring_response_file.xml
  -log user_home/var/temp/keyring_log.xml
```

b. **Use the response file to install the product silently:** Go to a command line on each of the systems on which you want to install the product, change to the `eclipse/tools` subdirectory in the directory where you installed Installation Manager, and install the product silently.

For example:

- **Windows** **Administrator or non-administrator:**

```
imcl.exe -acceptLicense
  input C:\temp\install_response_file.xml
  -log C:\temp\install_log.xml
  -keyring C:\IM\im.keyring
```

- **AIX** **HP-UX** **Linux** **Solaris** **Administrator:**

```
./imcl -acceptLicense
  input /var/temp/install_response_file.xml
  -log /var/temp/install_log.xml
  -keyring /var/IM/im.keyring
```

- **AIX** **HP-UX** **Linux** **Solaris** **Non-administrator:**

```
./imcl -acceptLicense
  input user_home/var/temp/install_response_file.xml
  -log user_home/var/temp/install_log.xml
  -keyring user_home/var/IM/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.
- The program might write important post-installation instructions to standard output.

Read the IBM Installation Manager Information Center for more information.

Example

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

```
<?xml version="1.0" encoding="UTF-8"?>

<!-- ##### Copyright ##################################################
# Licensed Materials - Property of IBM (c) Copyright IBM Corp. 2011.
# All Rights Reserved. US Government Users Restricted Rights-Use, duplication
# or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
################################################################# -->

<!-- ##### Frequently Asked Questions ###################################
# The latest information about using Installation Manager is
# located in the online Information Center. There you can find
# information about the commands and attributes used in
# silent installation response files.
#
#     Installation Manager Information Center can be found at:
#     http://publib.boulder.ibm.com/infocenter/install/v1r4/index.jsp
#
# Question 1. How do I record a response file using Installation Manager?
# Answer 1. Start Installation Manager from the command line under the
# eclipse subdirectory with the record parameter and it will generate a
# response file containing actions it performed, repositories it used, and
# its preferences settings. Optionally use the -skipInstall parameter if
# you do not want the product to be installed to the machine. Specify a
# new agentDataLocation location value when doing a new installation. Do
# not use an existing agentDataLocation for an installation because it might
# damage the installation data and prevent you from modifying, updating,
# rolling back, or uninstalling the installed packages.
#
# Windows: IBMIM -record <responseFile> -skipInstall <agentDataLocation>
# Linux or UNIX: ./IBMIM -record <responseFile> -skipInstall <agentDataLocation>
#
# For example:
#   Windows = IBMIM.exe -record c:\temp\responsefiles\WASv8.install.Win32.xml
#     -skipInstall c:\temp\skipInstall\WebSphere_Temp_Registry
#   Linux or UNIX = ./IBMIM -record /home/user/responsefiles/WASv8.install.RHEL64.xml
#     -skipInstall c:\temp\skipInstall\WebSphere_Temp_Registry
#
# Question 2. How do I run Installation Manager silently using response file?
# Answer 2. Create a silent installation response file and run the following command
# from the eclipse\tools subdirectory in the directory where you installed
# Installation Manager:
#
#   Windows = imcl.exe -acceptLicense -showProgress
#     input <response_file_path_and_name> -log <log_file_path_and_name>
```

```
#    Linux, UNIX, IBM i and z/OS = ./imcl -acceptLicense -showProgress
#       input <response_file_path_and_name> -log <log_file_path_and_name>
#
# For example:
#    Windows = imcl.exe -acceptLicense -showProgress
#       input c:\temp\responsefile\WASv8.install.Win32.xml
#    Linux, UNIX, IBM i and z/OS = ./imcl -acceptLicense -showProgress
#       input /home/user/responsefile/WASv8.install.RHEL64.xml
#
# The -acceptLicense command must be included to indicate acceptance of all
#     license agreements of all offerings being installed, updated or modified.
# The -showProgress command shows progress when running in silent mode.
# Additional commands can be displayed by requesting help:  IBMIM -help
#
# Question 3. How do I store and pass credentials to repositories that
# require authentication?
# Answer 3. Installation Manager uses a key ring file to store encrypted
# credentials for authenticating with repositories. Follow this two-step
# process for creating and using a key ring file with Installation Manager.
#
# First, create a key ring file with your credentials by starting
# Installation Manager from the command line under eclipse subdirectory
# with the keyring parameter.
# Use the optional password parameter to password protect your file.
#
#    Windows = IBMIM.exe -keyring <path and file name> -password <password>
#    Linux, UNIX, IBM i and z/OS = ./IBMIM -keyring <path and file name>
#                                     -password <password>
#
# Installation Manager will start in graphical mode. Verify that the
# repositories to which you need to authenticate are included in the
# preferences, File / Preferences / Repositories. If they are not
# listed, then click Add Repositories to add the URL or UNC path.
# Installation Manager will prompt for your credentials. If the repository
# is already in the list, then any attempt to access the repository location,
# such as clicking the Test Connections button, will also prompt for your
# credentials. Enter the correct credential and check the Save password
# checkbox. The credentials are saved to the key ring file you specified.
#
# Second, when you start a silent installation, run imcl under eclipse/tools
# subdirectory, and provide Installation Manager with the location of the key
# ring file and the password if the file is protected. For example:
#
#    Windows = imcl.exe -acceptLicense -showProgress
#       input <path and file name of response file>
#       -keyring <path and name of key ring file> -password <password>
#    Linux, UNIX, IBM i and z/OS = ./imcl -acceptLicense -showProgress
#       input <path and file name of response file>
#       -keyring <path and name of key ring file> -password <password>
#
################################################################### -->

<!-- ##### Agent Input ##########################################
#
# Note that the "acceptLicense" attribute has been deprecated.
# Use "-acceptLicense" command line option to accept license agreements.
#
# The clean and temporary attributes specify the repositories and other
# preferences Installation Manager uses and whether those settings
# should persist after the installation finishes.
#
# Valid values for clean:
#      true = only use the repositories and other preferences that are
#             specified in the response file.
#      false = use the repositories and other preferences that are
#             specified in the response file and Installation Manager.
#
# Valid values for temporary:
#      true = repositories and other preferences specified in the
#             response file do not persist in Installation Manager.
#      false = repositories and other preferences specified in the
#             response file persist in Installation Manager.
#
################################################################### -->

<agent-input clean="true" temporary="true">

<!-- ##### Repositories #########################################
# Repositories are locations that Installation Manager queries for
# installable packages. Repositories can be local (on the machine
# with Installation Manager) or remote (on a corporate intranet or
# hosted elsewhere on the internet).
#
# If the machine using this response file has access to the internet,
# then include the IBM WebSphere Live Update Repositories in the list
# of repository locations.
#
# If the machine using this response file cannot access the internet,
# then comment out the IBM WebSphere Live Update Repositories and
# specify the URL or UNC path to custom intranet repositories and
```

```
# directory paths to local repositories to use.
#
################################################################ -->

<server>
    <!-- ##### IBM WebSphere Live Update Repositories ####################
      # These repositories contain Web Server Plug-ins for IBM WebSphere
      # Application Server offerings, and updates for those offerings
      #
      # To use the secure repository (https), you must have an IBM ID,
      # which can be obtained by registering at: http://www.ibm.com/account
      # or your Passport Advantage account.
      #
      # And, you must use a key ring file with your response file.
      ############################################################## -->
<repository location="http://www.ibm.com/software/repositorymanager/com.ibm.websphere.PLG.v80" />
    <!-- <repository location="https://www.ibm.com/software/rational/repositorymanager/repositories/websphere" /> -->

    <!-- ##### Custom Repositories ###################################
      # Uncomment and update the repository location key below
      # to specify URLs or UNC paths to any intranet repositories
      # and directory paths to local repositories to use.
      ############################################################## -->
    <!-- <repository location='https:\\w3.mycompany.com\repositories\'/> -->
    <!-- <repository location='/home/user/repositories/websphere/'/> -->

    <!-- ##### Local Repositories ###################################
      # Uncomment and update the following line when using a local
      # repository located on your own machine to install a
      # Web Server Plug-ins offering.
      ##################################################### -->
    <!-- <repository location='insert the full directory path inside single quotes'/> -->
</server>

<!-- ##### Install Packages ##################################
#
# Install Command
#
# Use the install command to inform Installation Manager of the
# installation packages to install.
#
# The modify attribute is optional and can be paired with an install
# command to add features or paired with an uninstall command to
# remove commands. If omitted, the default value is set to false.
#    false = indicates not to modify an existing install by adding
#             or removing features.
#    true = indicates to modify an existing install by adding or
#             removing features.
#
# The offering ID attribute is required because it specifies the
# offering to be installed. The offering listed must be present in
# at least one of the repositories listed earlier. The example
# command below contains the offering ID for the Web Server Plug-ins.
#
# The version attribute is optional. If a version number is provided,
# then the offering will be installed at the version level specified
# as long as it is available in the repositories. If the version
# attribute is not provided, then the default behavior is to install
# the latest version available in the repositories. The version number
# can be found in the repository.xml file in the repositories.
# For example, <offering ... version='8.0.0.20110617_2222'>.
#
# The profile attribute is required and typically is unique to the
# offering. If modifying or updating an existing installation, the
# profile attribute must match the profile ID of the targeted installation
# of Web Server Plug-ins.
#
# The features attribute is optional. Offerings always have at least
# one feature; a required core feature which is installed regardless
# of whether it is explicitly specified. If other feature names
# are provided, then only those features will be installed.
# Features must be comma delimited without spaces.
#
# The feature values for Web Server Plug-ins include:
#  com.ibm.jre.6_32bit,com.ibm.jre.6_64bit
#
# On 32-bit machines, the 32-bit jre feature will be install
# automatically even if it is not specified in the response file.
#
# On 64-bit machines, one and only one of the Java Runtime Environment
# (JRE) features must be specified.
#
# The installFixes attribute indicates whether fixes available in
# repositories are installed with the product. By default, all
# available fixes will be installed with the offering.
#
# Valid values for installFixes:
#      none = do not install available fixes with the offering.
#      recommended = installs all available recommended fixes with the offering.
#      all = installs all available fixes with the offering.
```

```
#
# Interim fixes for offerings also can be installed while they
# are being installed by including the offering ID for the interim
# fix and specifying the profile ID. A commented out example is
# provided in the install command below.
#
# Installation Manager supports installing multiple offerings at once.
# Additional offerings can be included in the install command,
# with each offering requiring its own offering ID, version, profile value,
# and feature values.
#
# Profile Command
#
# A separate profile command must be included for each offering listed
# in the install command. The profile command informs Installation
# Manager about offering specific properties or configuration values.
#
# The installLocation specifies where the offering will be installed.
# If the response file is used to modify or update an existing
# installation, then ensure the installLocation points to the
# location where the offering was installed previously.
#
# The eclipseLocation data key should use the same directory path to
# Web Server Plug-ins as the installationLocation attribute.
#
# Include data keys for product specific profile properties.
#
#################################################################### -->

<install modify='false'>
<offering id='com.ibm.websphere.PLG.v80'
 profile='Web Server Plug-ins for IBM WebSphere Application Server V8.0'
 features='core.feature,com.ibm.jre.6_32bit' installFixes='none'/>
<!-- <offering id='PM12345_WAS80' profile='Web Server Plug-ins for IBM WebSphere Application Server V8.0'/> -->
</install>

<profile id='Web Server Plug-ins for IBM WebSphere Application Server V8.0'
 installLocation='C:\Program Files\IBM\WebSphere\Plugins'>
<data key='eclipseLocation' value='C:\Program Files\IBM\WebSphere\Plugins'/>
<data key='user.import.profile' value='false'/>
<data key='cic.selector.nl' value='en'/>
</profile>

<!-- ##### Shared Data Location #######################################
# Uncomment the preference for eclipseCache to set the shared data
# location the first time you use Installation Manager to do an
# installation.
#
# Eclipse cache location can be obtained from the installed.xml file found in
# Linux/Unix: /var/ibm/InstallationManager
# Windows: C:\Documents and Settings\All Users\Application Data\IBM\Installation Manager
# from the following property:
# <property name='cacheLocation' value='C:\Program Files\IBM\IMShared'/>
#
# Open the installed.xml file in a text editor because the style sheet
# might hide this value if opened in a web browser.
# For further information on how to edit preferences, refer to the public library at:
# http://publib.boulder.ibm.com/infocenter/install/v1r4/index.jsp?topic=/com.ibm.silentinstall12.doc/topics/r_silent_prefs.html
#
# After the shared data location is set, it cannot be changed
# using a response file or the graphical wizard.
#
# Ensure that the shared data location is a location that can be written
# to by all user accounts that are expected to use Installation Manager.
#
# By default, Installation Manager saves downloaded artifacts to
# the shared data location. This serves two purposes.
#
# First, if the same product is installed a more than once to the machine,
# then the files in the shared data location will be used rather than
# downloading them again.
#
# Second, during the rollback process, the saved artifacts are used.
# Otherwise, if the artifacts are not saved or are removed, then
# Installation Manager must have to access the repositories used to
# install the previous versions.
#
# Valid values for preserveDownloadedArtifacts:
#     true = store downloaded artifacts in the shared data location
#     false = remove downloaded artifacts from the shared data location
#
#################################################################### -->

<!--
<preference name='com.ibm.cic.common.core.preferences.eclipseCache' value='C:\Program Files\IBM\IMShared'/>
<preference name='com.ibm.cic.common.core.preferences.preserveDownloadedArtifacts' value='true'/>
-->


<!-- ##### Preferences Settings #######################################
```

```
# Additional preferences for Installation Manager can be specified.
# These preference correspond to those that are located in the graphical
# interface under File / Preferences.
#
# If a preference command is omitted from or commented out of the response
# file, then Installation Manager uses the preference value that was
# previously set or the default value for the preference.
#
# Preference settings might be added or deprecated in new versions of
# Installation Manager. Consult the online Installation Manager
# Information Center for the latest set of preferences and
# descriptions about how to use them.
#
# http://publib.boulder.ibm.com/infocenter/install/v1r4/index.jsp
#
################################################################ -->

<!--
<preference name='com.ibm.cic.common.core.preferences.connectTimeout' value='30'/>
<preference name='com.ibm.cic.common.core.preferences.readTimeout' value='45'/>
<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.keepFetchedFiles' value='false'/>
<preference name='PassportAdvantageIsEnabled' value='false'/>
<preference name='com.ibm.cic.common.core.preferences.searchForUpdates' value='false'/>
<preference name='com.ibm.cic.agent.ui.displayInternalVersion' value='false'/>
 -->

</agent-input>
```

Important: ▐ AIX ▌ ▐ Linux ▌ ▐ Windows ▌ If you are installing on a 64-bit system, you must include one of the options for an IBM Runtime Environment for Java.

- If you want to use the 32-bit IBM Runtime Environment for Java, include com.ibm.jre.6_32bit as a feature in the response file.

  For example:

```
<offering profile='Web Server Plug-ins for IBM WebSphere Application Server V8.0'
  features='core.feature,com.ibm.jre.6_32bit' id='com.ibm.websphere.PLG.v80'/>
```

- If you want to use the 64-bit IBM Runtime Environment for Java, include com.ibm.jre.6_64bit as a feature in the response file.

  For example:

```
<offering profile='Web Server Plug-ins for IBM WebSphere Application Server V8.0'
  features='core.feature,com.ibm.jre.6_64bit' id='com.ibm.websphere.PLG.v80'/>
```

Follow these guidelines:

- Include this feature only if you are installing on a 64-bit system; do not include it if you are installing on a 32-bit system.
- You must include one of the two options if you are installing on a 64-bit system.
- You cannot modify this installation later and change the selection.

Installing fix packs on the Web Server Plug-ins using the Installation Manager GUI

You can use Installation Manager to update the Web Server Plug-ins to a later version.

Before you begin

Make sure that the web-based or local service repository location is listed and checked or that the **Search service repositories during installation and updates** option is selected on the Repositories panel in your Installation Manager preferences. For more information on using service repositories with Installation Manager, read the Installation Manager Information Center.

About this task

Perform this procedure to use Installation Manager to update the Web Server Plug-ins.

Procedure

1. Start Installation Manager.
2. Click **Update**.

   Note: If you are prompted to authenticate, use the IBM ID and password that you use to access protected IBM software websites.
3. Select the package group to update.
4. Click **Next**.
5. Select the version to which you want to update under **Web Server Plug-ins for IBM WebSphere Application Server**.
6. Click **Next**.
7. Accept the terms in the license agreements, and click **Next**.
8. Review the summary information, and click **Update**.
   - If the installation is successful, the program displays a message indicating that installation is successful.
   - If the installation is not successful, click **View Log File** to troubleshoot the problem.
9. Click **Finish**.
10. Click **File > Exit** to close Installation Manager.

Uninstalling fix packs from the Web Server Plug-ins using the Installation Manager GUI

You can use Installation Manager to roll back the Web Server Plug-ins to an earlier version.

Before you begin

During the rollback process, Installation Manager must access files from the earlier version of the package. By default, these files are stored on your computer when you install a package. If you change the default setting or delete the files using the **Remove Saved Files** option, Installation Manager requires access to the repository that was used to install the earlier version.

About this task

Perform this procedure to use Installation Manager to roll back the Web Server Plug-ins to an earlier version.

Procedure

1. Stop all servers on the WebSphere Application Server installation that is being modified.
2. Start Installation Manager.
3. Click **Roll Back**.

   Note: If you are prompted to authenticate, use the IBM ID and password that you use to access protected IBM software websites.
4. Select the package group to roll back.
5. Click **Next**.
6. Select the version to which you want to roll back under **Web Server Plug-ins for IBM WebSphere Application Server**.
7. Click **Next**.
8. Review the summary information, and click **Roll Back**.
   - If the rollback is successful, the program displays a message indicating that the rollback is successful.
   - If the rollback is not successful, click **View Log File** to troubleshoot the problem.

9. Click **Finish**.
10. Click **File > Exit** to close Installation Manager.

Uninstalling Web Server Plug-ins using the GUI

Use the Installation Manager GUI to uninstall the Web Server Plug-ins.

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.

Procedure

1. Start Installation Manager.
2. Click **Uninstall**.
3. In the **Uninstall Packages** window, perform the following actions.
   a. Select **Web Server Plug-ins for IBM WebSphere Application Server** and the appropriate version.

      Note: If you are uninstalling the ILAN version of this product, select **Web Server Plug-ins for IBM WebSphere Application Server (ILAN)**.
   b. Click **Next**.
4. Review the summary information.
5. Click **Uninstall**.
   - If the uninstallation is successful, the program displays a message that indicates success.
   - If the uninstallation is not successful, click **View log** to troubleshoot the problem.
6. Click **Finish**.
7. Click **File > Exit** to close Installation Manager.

Uninstalling the Web Server Plug-ins silently

You can use Installation Manager to uninstall the Web Server Plug-ins silently.

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.

Optional: Perform or record the installation of Installation Manager and installation of the Web Server Plug-ins to a temporary installation registry on one of your systems so that you can use this temporary registry to record the uninstallation without using the standard registry where Installation Manager is installed.

Read the following for more information:
- "Installing the Web Server Plug-ins using the GUI" on page 57
- "Installing the Web Server Plug-ins silently" on page 61

About this task

Using Installation Manager, you can work with response files to uninstall the Web Server Plug-ins silently in a variety of ways. You can record a response file using the GUI as described in the following procedure, or you can generate a new response file by hand or by taking an example and modifying it.

Procedure

1. Optional: **Record a response file to uninstall the Web Server Plug-ins:** On one of your systems, perform the following actions to record a response file that will uninstall the Web Server Plug-ins:

   a. From a command line, change to the eclipse subdirectory in the directory where you installed Installation Manager.

   b. Start Installation Manager from the command line using the -record option.

   For example:

   • **Windows** **Administrator or non-administrator:**

```
IBMIM.exe -skipInstall "C:\temp\imRegistry"
 -record C:\temp\uninstall_response_file.xml
```

   • **AIX** **HP-UX** **Linux** **Solaris** **Administrator:**

```
./IBMIM -skipInstall /var/temp/imRegistry
 -record /var/temp/uninstall_response_file.xml
```

   • **AIX** **HP-UX** **Linux** **Solaris** **Non-administrator:**

```
./IBMIM -skipInstall user_home/var/temp/imRegistry
 -record user_home/var/temp/uninstall_response_file.xml
```

   Tip: If you choose to use the -skipInstall parameter with a temporary installation registry created as described in "Before you begin," Installation Manager uses the temporary installation registry while recording the response file. It is important to note that when the -skipInstall parameter is specified, no packages are installed or uninstalled. All of the actions that you perform in Installation Manager simply update the installation data that is stored in the specified temporary registry. After the response file is generated, it can be used to uninstall the Web Server Plug-ins, removing the Web Server Plug-ins files and updating the standard installation registry.

   The -skipInstall operation should not be used on the actual agent data location used by Installation Manager. This is unsupported. Use a clean writable location, and re-use that location for future recording sessions.

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

   c. Click **Uninstall**.

   d. In the **Uninstall Packages** window, perform the following actions.

      1) Select **Web Server Plug-ins for IBM WebSphere Application Server** and the appropriate version.

      Note: If you are uninstalling the ILAN version of this product, select **Web Server Plug-ins for IBM WebSphere Application Server (ILAN)**.

      2) Click **Next**.

   e. Review the summary information.

   f. Click **Uninstall**.

      • If the uninstallation is successful, the program displays a message that indicates success.

      • If the uninstallation is not successful, click **View log** to troubleshoot the problem.

   g. Click **Finish**.

   h. Click **File > Exit** to close Installation Manager.

2. **Use the response file to uninstall the Web Server Plug-ins silently:** 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 response file that you created to silently uninstall the Web Server Plug-ins.

   For example:

   • **Windows** **Administrator or non-administrator:**

```
imcl.exe
 input C:\temp\uninstall_response_file.xml
 -log C:\temp\uninstall_log.xml
```

- **AIX** **HP-UX** **Linux** **Solaris** **Administrator:**

```
./imcl
  input /var/temp/uninstall_response_file.xml
  -log /var/temp/uninstall_log.xml
```

- **AIX** **HP-UX** **Linux** **Solaris** **Non-administrator:**

```
./imcl
  input user_home/var/temp/uninstall_response_file.xml
  -log user_home/var/temp/uninstall_log.xml
```

Go to the IBM Installation Manager Information Center for more information.

Windows

Example

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

```
<?xml version="1.0" encoding="UTF-8"?>

<!-- ##### Copyright #################################################
# Licensed Materials - Property of IBM (c) Copyright IBM Corp. 2011.
# All Rights Reserved. US Government Users Restricted Rights-Use, duplication
# or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
################################################################# -->

<!-- ##### Frequently Asked Questions #################################
# The latest information about using Installation Manager is
# located in the online Information Center. There you can find
# information about the commands and attributes used in
# silent installation response files.
#
#     Installation Manager Information Center can be found at:
#     http://publib.boulder.ibm.com/infocenter/install/v1r4/index.jsp
#
# Question 1. How do I record a response file using Installation Manager?
# Answer 1. Start Installation Manager from the command line under the
# eclipse subdirectory with the record parameter and it will generate a
# response file containing actions it performed, repositories it used, and
# its preferences settings. Optionally use the -skipInstall parameter if
# you do not want the product to be installed to the machine. Specify a
# new agentDataLocation location value when doing a new installation. Do
# not use an existing agentDataLocation for an installation because it might
# damage the installation data and prevent you from modifying, updating,
# rolling back, or uninstalling the installed packages.
#
# Windows: IBMIM -record <responseFile> -skipInstall <agentDataLocation>
# Linux or UNIX: ./IBMIM -record <responseFile> -skipInstall <agentDataLocation>
#
# For example:
#   Windows = IBMIM.exe -record c:\temp\responsefiles\WASv8.install.Win32.xml
#     -skipInstall c:\temp\skipInstall\WebSphere_Temp_Registry
#   Linux or UNIX = ./IBMIM -record /home/user/responsefiles/WASv8.install.RHEL64.xml
#     -skipInstall c:\temp\skipInstall\WebSphere_Temp_Registry
#
# Question 2. How do I run Installation Manager silently using response file?
# Answer 2. Create a silent installation response file and run the following command
# from the eclipse\tools subdirectory in the directory where you installed
# Installation Manager:
#
#   Windows = imcl.exe -acceptLicense -showProgress
#     input <response_file_path_and_name> -log <log_file_path_and_name>
#   Linux, UNIX, IBM i and z/OS = ./imcl -acceptLicense -showProgress
#     input <response_file_path_and_name> -log <log_file_path_and_name>
#
# For example:
#   Windows = imcl.exe -acceptLicense -showProgress
#     input c:\temp\responsefile\WASv8.install.Win32.xml
#   Linux, UNIX, IBM i and z/OS = ./imcl -acceptLicense -showProgress
#     input /home/user/responsefile/WASv8.install.RHEL64.xml
#
# The -acceptLicense command must be included to indicate acceptance of all
#     license agreements of all offerings being installed, updated or modified.
# The -showProgress command shows progress when running in silent mode.
# Additional commands can be displayed by requesting help:  IBMIM -help
#
################################################################# -->

<!-- ##### Agent Input #####################################
# The clean and temporary attributes specify the repositories and other
# preferences Installation Manager uses and whether those settings
# should persist after the uninstall finishes.
#
# Valid values for clean:
#     true = only use the repositories and other preferences that are
#            specified in the response file.
#     false = use the repositories and other preferences that are
```

```
#            specified in the response file and Installation Manager.
#
# Valid values for temporary:
#      true = repositories and other preferences specified in the
#             response file do not persist in Installation Manager.
#      false = repositories and other preferences specified in the
#             response file persist in Installation Manager.
#
################################################################### -->

<agent-input clean='true' temporary='true'>

<!-- ##### Repositories ##################################################
# Repositories are locations that Installation Manager queries for
# installable packages. Repositories can be local (on the machine
# with Installation Manager) or remote (on a corporate intranet or
# hosted elsewhere on the internet).
#
# If the machine using this response file has access to the internet,
# then include the IBM WebSphere Live Update Repositories in the list
# of repository locations.
#
# If the machine using this response file cannot access the internet,
# then comment out the IBM WebSphere Live Update Repositories and
# specify the URL or UNC path to custom intranet repositories and
# directory paths to local repositories to use.
#
################################################################### -->

<server>
    <!-- ##### IBM WebSphere Live Update Repositories ####################
     # These repositories contain Web Server Plug-ins for WebSphere
  # Application Server offerings, and updates for those offerings
     #
     # To use the secure repository (https), you must have an IBM ID,
     # which can be obtained by registering at: http://www.ibm.com/account
     # or your Passport Advantage account.
     #
     # And, you must use a key ring file with your response file.
     ############################################################## -->
    <repository location="http://www.ibm.com/software/repositorymanager/ccom.ibm.websphere.PLG.v80" />
    <!-- <repository location="https://www.ibm.com/software/rational/repositorymanager/repositories/websphere" /> -->

    <!-- ##### Custom Repositories ###################################
     # Uncomment and update the repository location key below
     # to specify URLs or UNC paths to any intranet repositories
     # and directory paths to local repositories to use.
     ########################################################### -->
    <!-- <repository location='https:\\w3.mycompany.com\repositories\'/> -->
    <!-- <repository location='/home/user/repositories/websphere/'/> -->

    <!-- ##### Local Repositories ###################################
     # Uncomment and update the following line when using a local
     # repository located on your own machine to install a
     # Web Server Plug-ins offering.
     ######################################################### -->
    <!-- <repository location='insert the full directory path inside single quotes'/> -->
</server>

<!-- ##### Uninstall Packages ########################################
#
# Uninstall Command
#
# Use the uninstall command to inform Installation Manager of the
# installation packages to uninstall.
#
# The modify attribute is optional and can be paired with an install
# command to add features or paired with an uninstall command to
# remove commands. If omitted, the default value is set to false.
#    false = indicates not to modify an existing install by adding
#            or removing features.
#    true = indicates to modify an existing install by adding or
#            removing features.
#
# The offering ID attribute is required because it specifies the
# offering to be uninstalled. The example command below contains the
# offering ID for Web Server Plug-ins.
#
# The version attribute is optional. If a version number is provided,
# then the offering will be uninstalled at the version level specified
# If the version attribute is not provided, then the default behavior is
# to uninstall the latest version. The version number can be found in
# the repository.xml file in the repositories.
# For example, <offering ... version='8.0.0.20110617_2222'>.
#
# The profile attribute is required and must match the package group
# name for the offering to be uninstalled.
#
# The features attribute is optional. If there is no feature attribute,
# then all features are uninstalled. If features are specified, then
```

```
# only those features will be uninstalled.
# Features must be comma delimited without spaces.
#
# The feature values for Web Server Plug-ins include:
#   com.ibm.jre.6_32bit,com.ibm.jre.6_64bit
#
# Installation Manager supports uninstalling multiple offerings at once.
# Additional offerings can be included in the uninstall command,
# with each offering requiring its own offering ID, version, profile value,
# and feature values.
#
# Profile Command
#
# A separate profile command must be included for each offering listed
# in the install command. The profile command informs Installation
# Manager about offering specific properties or configuration values.
#
# The installLocation specifies where the offering will be installed.
# If the response file is used to modify or update an existing
# installation, then ensure the installLocation points to the
# location where the offering was installed previously.
#
# The eclipseLocation data key should use the same directory path to
# Web Server Plug-ins as the installationLocation attribute.
#
# Include data keys for product specific profile properties.
#
################################################################# -->
<uninstall modify='false'>
<offering id='com.ibm.websphere.PLG.v80'
 profile='Web Server Plug-ins for IBM WebSphere Application Server V8.0'
 features='core.feature,com.ibm.jre.6_32bit'/>
</uninstall>

<profile id='Web Server Plug-ins for IBM WebSphere Application Server V8.0'
 installLocation='C:\Program Files\IBM\WebSphere\Plugins'>
<data key='eclipseLocation' value='C:\Program Files\IBM\WebSphere\Plugins'/>
<data key='user.import.profile' value='false'/>
<data key='cic.selector.nl' value='en'/>
</profile>

<!-- ##### Shared Data Location #########################################
# Uncomment the preference for eclipseCache to set the shared data
# location the first time you use Installation Manager to do an
# installation.
#
# Eclipse cache location can be obtained from the installed.xml file found in
# Linux/Unix: /var/ibm/InstallationManager
# Windows: C:\Documents and Settings\All Users\Application Data\IBM\Installation Manager
# from the following property:
# <property name='cacheLocation' value='C:\Program Files\IBM\IMShared'/>
#
# Open the installed.xml file in a text editor because the style sheet
# might hide this value if opened in a web browser.
# For further information on how to edit preferences, refer to the public library at:
# http://publib.boulder.ibm.com/infocenter/install/v1r4/index.jsp?topic=/com.ibm.silentinstall12.doc/topics/r_silent_prefs.html
#
# After the shared data location is set, it cannot be changed
# using a response file or the graphical wizard.
#
# Ensure that the shared data location is a location that can be written
# to by all user accounts that are expected to use Installation Manager.
#
# By default, Installation Manager saves downloaded artifacts to
# the shared data location. This serves two purposes.
#
# First, if the same product is installed a more than once to the machine,
# then the files in the shared data location will be used rather than
# downloading them again.
#
# Second, during the rollback process, the saved artifacts are used.
# Otherwise, if the artifacts are not saved or are removed, then
# Installation Manager must have to access the repositories used to
# install the previous versions.
#
# Valid values for preserveDownloadedArtifacts:
#     true = store downloaded artifacts in the shared data location
#     false = remove downloaded artifacts from the shared data location
#
################################################################# -->

<!--
<preference name='com.ibm.cic.common.core.preferences.eclipseCache' value='C:\Program Files\IBM\IMShared'/>
<preference name='com.ibm.cic.common.core.preferences.preserveDownloadedArtifacts' value='true'/>
-->

<!-- ##### Preferences Settings #########################################
# Additional preferences for Installation Manager can be specified.
# These preference correspond to those that are located in the graphical
# interface under File / Preferences.
```

```
#
# If a preference command is omitted from or commented out of the response
# file, then Installation Manager uses the preference value that was
# previously set or the default value for the preference.
#
# Preference settings might be added or deprecated in new versions of
# Installation Manager. Consult the online Installation Manager
# Information Center for the latest set of preferences and
# descriptions about how to use them.
#
# http://publib.boulder.ibm.com/infocenter/install/v1r4/index.jsp
#
#################################################################### -->

<!--
<preference name='com.ibm.cic.common.core.preferences.connectTimeout' value='30'/>
<preference name='com.ibm.cic.common.core.preferences.readTimeout' value='45'/>
<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.keepFetchedFiles' value='false'/>
<preference name='PassportAdvantageIsEnabled' value='false'/>
<preference name='com.ibm.cic.common.core.preferences.searchForUpdates' value='false'/>
<preference name='com.ibm.cic.agent.ui.displayInternalVersion' value='false'/>
 -->

</agent-input>
```

Plug-ins configuration

The Web Server Plug-ins Configuration Tool configures an application server for a web server type and creates a web server definition in the configuration of the application server. This overview shows the different processing paths that the Web Server Plug-ins Configuration Tool can use.

This topic describes the two ways that the Web Server Plug-ins Configuration Tool can configure a Web server and create the `plugin-cfg.xml` file, which is the plug-in configuration file.

Tip: As an alternative to using the Web Server Plug-ins Configuration Tool, you can use the pct command-line tool with a response file to configure a web server. Read "Configuring a web server plug-in using the WCT command-line utility" on page 109 for more information.

Configuration flows

The Web Server Plug-ins Configuration Tool resolves all configurations of the web server and WebSphere Application Server to two scenarios: remote plug-in configuration and local plug-in configuration. The logic implemented in determining which scenario applies to a configuration is shown in the following diagram.

Web server plug-ins for WebSphere Application Server
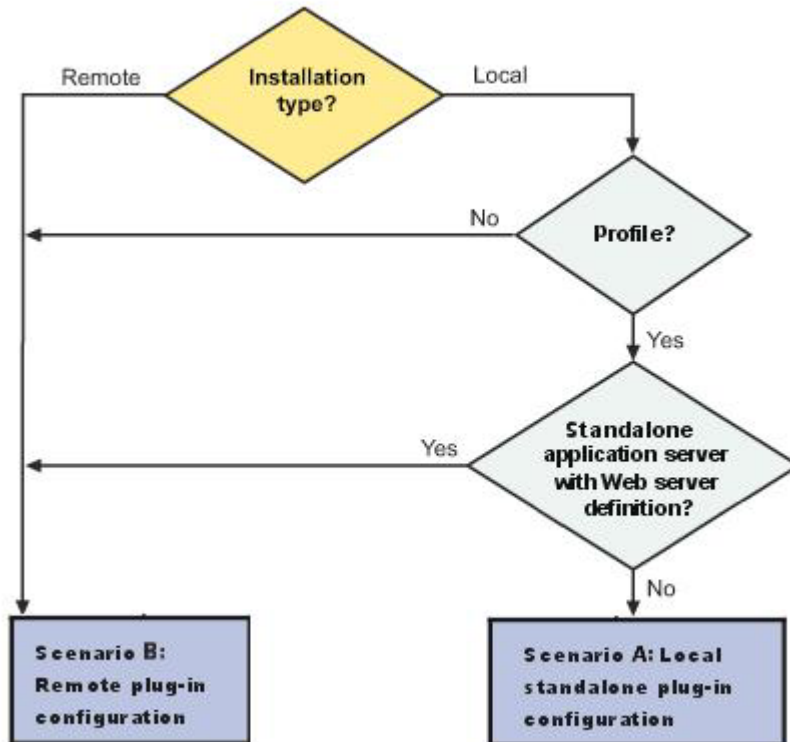
Legend:

Installation type?

> The installation type is either remote or local.

> When the Web server and application server are not on the same computer, choose the remote scenario. When both the web server and application server are on the same computer, choose the local scenario.

Profile?

> If the product is installed but the profile is accidentally deleted or otherwise missing, the scenario is considered to be a remote installation. Create a profile before running the script.

Standalone application server with web server definition?

> If the profile has an existing web server definition, the installation is considered a remote plug-in configuration. You cannot have more than one web server definition in a standalone application server.

Scenario A. Local standalone plug-in configuration

In this scenario, the Web Server Plug-ins Configuration Tool creates a web server definition within the application server profile directly, without the use of a script.

The Web Server Plug-ins Configuration Tool configures the web server to use the `plugin-cfg.xml` file that is within the application server profile. The application server regenerates the `plugin-cfg.xml` file in the

profile_root/`config/cells/` *cell_name*/`nodes/` *web_server_name*_`node/servers/` *web_server_name*
directory. Regeneration occurs whenever a change occurs in the application server configuration that
affects deployed applications.

After configuring the application server for the local web server, you can start the application server and
the web server immediately.

Table 6. Configuration that qualifies for the local application server scenario.

Profile type	Automatic creation of web server definition?	Web server already defined in application server configuration?
Application server	Yes	No

Redirection to Scenario B

A application server profile that has an existing web server definition should be processed as a remote
plug-in configuration. An application server can have just one web server definition.

See **Scenario B** for a description of this type of node.

Overview of the verification procedure

The following overview shows the procedure for verifying the web server configuration:
1. Start the web server with the proper procedure for your web server.

   For example, start the IBM HTTP Server from a command line:
   - `AIX` `HP-UX` `Linux` `Solaris` `./`*IHS_root*`/bin/apachectl start`
   - `Windows` *IHS_root*`\bin\apache`
2. Start the application server.

   Change directories to the *profile_root*/`bin` directory and run the startServer command:
   - `AIX` `HP-UX` `Linux` `Solaris` `./`*profile_root*`/bin/startServer.sh server1`
   - `Windows` *profile_root*`\bin\startServer server1`

   Open the administrative console, and save the changed configuration.
3. Point your browser to `http://localhost:9080/snoop` to test the internal HTTP transport provided by
   the Application Server. Point your browser to `http://`*Host_name_of_Web_server_machine*`/snoop` to test
   the web server plug-in.
4. Verify that both web addresses display the Snoop Servlet - Request/Client Information page.

Scenario B. Remote plug-in configuration

The Web Server Plug-ins Configuration Tool does not automatically create a web server definition within
the distributed profile on a remote machine. The tool creates the `configure`*web_server_name* script instead.

The Web Server Plug-ins Configuration Tool configures the web server to use the `plugin-cfg.xml` file that
is maintained on the web server machine in the *plugins_root*/`config`/*web_server_name* directory. This file
requires periodic propagation. Propagation is copying the current `plugin-cfg.xml` file from the application
server machine to replace the *plugins_root*/`config`/*web_server_name*/`plugin-cfg.xml` file.

After installing the binary plug-in for the local web server, you do not have to run the script before you can
start the application server and the web server. However, you do not have the benefits of a Web server
definition in the application server node until you run the script.

Table 7. Configurations that qualify for the remote application server scenario.

Profile type	Creation of web server definition?	Web server already defined in application server configuration?
Any profile anywhere if you select a remote installation type in the Web Server Plug-ins Configuration Tool	By script	N/A
No profile	By script	N/A
Standalone application server profile with an existing web server definition	By script	Yes

Testing the application server without a web server definition: The following overview shows the procedure for verifying the temporary *plugins_root*/config/*web_server_name*/plugin-cfg.xml file.

The web server communicates with the remote application server using the temporary plugin-cfg.xml file.

If the application server has an HTTP Transport port assignment other than 9080, the test is not successful. Continue to the next section to create the web server definition on the application server and to complete your test of the configuration.

1. Start the web server with the proper procedure for your web server.

   For example, start the IBM HTTP Server from a command line:
   - **AIX** **HP-UX** **Linux** **Solaris** `./IHS_root/bin/apachectl start`
   - **Windows** `IHS_root\bin\apache`

2. Start the application server on the remote machine.

   Change directories to the *profile_root*/bin directory and run the startServer command:
   - **AIX** **HP-UX** **Linux** **Solaris** `./profile_root/bin/startServer.sh server1`
   - **Windows** `profile_root\bin\startServer server1`

3. Point your browser to http://localhost:9080/snoop to test the internal HTTP transport provided by the application server. Point your browser to http://*Host_name_of_Web_server_machine*/snoop to test the web server plug-in.

4. Verify that both web addresses display the Snoop Servlet - Request/Client Information page.

Completing the installation by configuring a web server definition: The following overview shows the procedure for completing the configuration. The configuration is not complete until the Web server definition exists in the configuration of the application server node. The web server definition is a central element in the regeneration of a valid plug-in configuration file, plugin-cfg.xml.

1. Create the web server definition in the application server.

   Run the script to manually create the web server definition within the configuration of the application server node:

   a. Copy the script from the *plugins_root*/bin directory to the remote *app_server_root*/bin directory.

   b. Open a command window and run the script:
      - **AIX** **HP-UX** **Linux** **Solaris** `./configureweb_server_name.sh`
      - **Windows** `configureweb_server_name.bat`

   The configureweb_server_name script can take three parameters: *profile_name*, *Admin_Console_Username* and *Admin_Console_Password*.

   - *profile_name* indicates the name of the profile used to create the web server definition.
   - *Admin_Console_Username* indicates the username of the administrative console. The profile with the administrative console deployed must have administrative-console security turned on. This parameter can not be used if *profile_name* is blank.
   - *Admin_Console_Password* indicates the password corresponding to the username. This parameter cannot be used if both *profile_name* and *Admin_Console_Username* are blank.

Note: The *webserverNodeName* value in the script is a concatenation of the nickname that you have chosen for the Web server and the suffix *-node*. It is automatically created during plug-in configuration and cannot be changed. For example, if you named your web server *myserver* during plug-in configuration, the value for the associated web server definition created after you ran the script would be *myserver-node*.

If you have enabled security or changed the default JMX connector type, edit the script and include the appropriate parameters.

2. Copy the current plug-in configuration file, `plugin-cfg.xml`, in the *profile_root*/`config/cells/`*cell_name*/`nodes/`*web_server_name*`_node/servers/`*web_server_name* directory. Paste the file on the web server machine to replace the temporary *plugins_root*/`config/`*web_server_name*/`plugin-cfg.xml` file. The IBM HTTP Server supports automatic propagation. Other web servers require manual propagation.

3. Start the web server with the proper procedure for your web server. Open the administrative console, and save the changed configuration.

4. Point your browser to `http://localhost:9080/snoop` to test the internal HTTP transport provided by the application server. Point your browser to `http://`*Host_name_of_Web_server_machine*/`snoop` to test the web server plug-in.

5. Verify that both web addresses display the Snoop Servlet - Request/Client Information page.

Summary

Two scenarios exist for Web Server Plug-ins. Each scenario revolves around a unique location for the plug-in configuration file, `plugin-cfg.xml`.

The application server generates the plug-in configuration file. The purpose of the file is to publish the location of all of the application server objects that are relevant to a web server and to control binary plug-in configuration options. The file identifies objects such as applications and virtual hosts for serving applications.

If the web server cannot access the file on the application server machine, you must copy the file to the web server. That process is called "propagation." Propagation is reserved for the remote plug-in configuration scenario, which is **Scenario B** in this topic.

In the local scenario, the web server can access the `plugin-cfg.xml` file because the web server is on the same machine as the file.

All **Scenario A** configurations have the web server definition within its own web server node.

Limited management options do not let you create or delete the one web server definition in the administrative console of a standalone application server. The inability of a standalone application server to create a web server definition is the basis for the configuration scripts created by the Web Server Plug-ins Configuration Tool. Without the scripts, you could not easily create a web server definition on a standalone application server node.

The location of the `plugin-cfg.xml` file for each configuration described in this topic is shown in the following table:

Table 8. Plug-in configuration file locations.

		Location of the `plugin-cfg.xml` file	
Scenario	Profile type	*plugins_root*	*profile_root*: **within the web server node**
A	Application server profile		X

Table 8. Plug-in configuration file locations (continued).

Scenario	Profile type	Location of the `plugin-cfg.xml` file	
		plugins_root	*profile_root*: within the web server node
B	Any profile anywhere if you select a remote installation type in the Web Server Plug-ins Configuration Tool	X	
	No profile	X	
	Application server profile with an existing web server definition	X	

Legend:

plugins_root

plugins_root/config/*web_server_name*/plugin-cfg.xml

profile_ root: **within the web server node**

profile_root/config/cells/*cell_name*/nodes/*web_server_name*_node/servers/*web_server_name*/plugin-cfg.xml

Configuring a web server and an application server profile on the same machine

This topic describes configuring a web server plug-in that WebSphere Application Server provides to communicate with a particular brand of web server. This procedure describes installing the web server and its web server plug-in for WebSphere Application Server and the application server on the same machine.

Before you begin

When multiple profiles exist, you can select the profile that the Web Server Plug-ins Configuration Tool configures. See "Plug-ins configuration" on page 78 for a description of the flow of logic that determines how to select the profile to configure.

If the WebSphere Application Server product family supports a particular brand of web server, such as IBM HTTP Server or Microsoft Internet Information Services (IIS), your WebSphere Application Server product provides a binary plug-in for the web server that you must install.

If the WebSphere Application Server product family does not provide a binary plug-in for a particular brand of web server, then the web server is not supported. The purpose of the binary plug-in is to provide the communication protocol between the web server and the application server.

Suppose that you create a new profile and you also want to use a web server. You must install a new web server for the new profile, install the Web Server Plug-ins, and use the Web Server Plug-ins Configuration Tool to configure both the web server and the application server.

If the web server is not already installed, you can still install the Web Server Plug-ins for future use.

This procedure configures the application server profile that is the default profile on the machine. A one-to-one relationship exists between a web server and the application server.

However, a standalone application server profile and a managed profile can each have multiple web servers defined, each in a separate web server definition.

This article describes how to create the following topology:
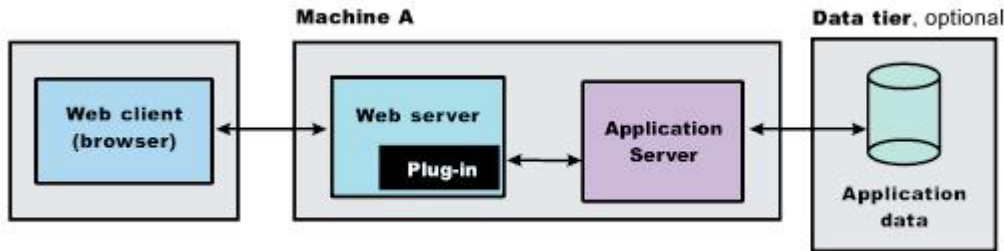
Note: Nonroot installation for the plug-in component is only supported if the application server was also installed by the same nonroot user. Otherwise, the web server configuration scripts will fail to run against the application server installation.

About this task

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.

The Web Server Plug-ins Configuration Tool configures the plug-in for the supported web server after collecting the following information:

- Type of web server to configure
- Architecture of your installed target web server (64 bit or 32 bit)
- Location of the configuration file or files for the web server to be configured
- Web server port
- For IBM HTTP Server, the following information:
    - Port number for optional IBM HTTP Server administrative server setup
    - User ID and password to authenticate to the optional IBM HTTP Server administrative server from the administrative console
    - AIX HP-UX Linux Solaris System user ID and group to have write permission to IBM HTTP Server, the IBM HTTP Server administrative server, and the web server plug-in configuration files
    - Windows User ID and password if you choose to run the IBM HTTP Server administrative server as a Window service
- Name of the web server definition
- Configuration scenario to be used
    - If it is a remote scenario, the tool collects the host name or IP address of the application server.
    - If it is a local scenario, the tool collects the installation root directory of the WebSphere Application Server product.
- Profile to be configured with the web server plug-in

The Web Server Plug-ins Configuration Tool edits the configuration file or files for a web server by creating directives that point to the location of the binary plug-in module and the plug-in configuration file.

The name of the binary plug-in module varies per web server type. The plug-in configuration file is always the `plugin-cfg.xml` file.

The Web Server Plug-ins Configuration Tool creates a web server definition in the configuration of the application server unless one already exists.

You can use the administrative console to manage the web server configuration. For example, when you install an application on the application server, you can also choose to install it on the web server definition. If so, the updated `plugin-cfg.xml` file shows that the new application is available. When the web server reads the updated plug-in configuration file, the web server becomes aware of the new application that it can serve to web clients.

If you choose not to install the new application on the web server definition, the application is not added to the plug-in configuration file. The web server is not aware of the application and cannot serve it to web clients.

Tip: As an alternative to using the Web Server Plug-ins Configuration Tool, you can use the pct command-line tool with a response file to configure a web server. Read "Configuring a web server plug-in using the WCT command-line utility" on page 109 for more information.

Use this procedure to install the web server plug-in, configure the web server, and create a web server definition in the default application server profile.

Procedure

Configure a standalone application server.

1. Log on to the operating system.

   If you are installing as a nonroot or non-administrative user, then there are certain limitations.

   AIX **HP-UX** **Linux** **Solaris** In addition, select a umask that allows the owner to read/write to the files, and allows others to access them according to the prevailing system policy. For root, a umask of 022 is recommended. For nonroot users, a umask of 002 or 022 could be used, depending on whether or not the users share the group. To verify the umask setting, issue the following command:

`umask`

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

`umask 022`

   Windows When installing as an administrative user on a Windows operating system, a Windows service is automatically created to autostart the application server. The installer user account must have the following advanced user rights:

   - Act as part of the operating system
   - Log on as a service

   For example, on some Windows operating systems, click **Control Panel > Administrative Tools > Local Security Policy > Local Policies > User Rights Assignments** to set the advanced options. See your Windows operating system documentation for more information.

   Windows If you plan to run the application server as a Windows service, do not install from a user ID that contains spaces. A user ID with spaces cannot be validated. Such a user ID is not allowed to continue the installation. To work around this restriction, install with a user ID that does not contain spaces.

2. Install Installation Manager.

3. Use Installation Manager to install the following:

   - WebSphere Application Server
   - Web Server Plug-ins for WebSphere Application Server

- Websphere Customization Toolbox
4. Use Installation Manager to install the IBM HTTP Server, or install another supported web server.
5. Open the Websphere Customization Toolbox, and launch the Web Server Plug-ins Configuration Tool.
6. Select a Web Server Plug-ins runtime location.

   If the location of a previously installed web server plug-in that you want to use is not in the list, perform the following actions to add the location to your working set:

   a. Click **Add**.

   b. Enter a name for the web server plug-in location.

   c. Perform one of the following actions:
      - Enter the location.
      - Click **Browse**, find the location, and click **OK**.

7. Click **Create**.
8. Select the type of web server that you are configuring, and click **Next**.
9. Select the architecture of your installed target web server (64 bit or 32 bit) and click **Next** if you are asked.
10. Click **Browse** to select the configuration file or files for your web server, verify that the web server port is correct, and then click **Next** when you are finished.

    Select the file and not just the directory of the file. Some web servers have two configuration files and require you to browse for each file.

    The following list shows configuration files for supported web servers:

    Apache HTTP Server
    > *apache_root*/config/httpd.conf

    Domino Web Server
    > names.nsf and Notes.jar

    > The wizard prompts for the notes.jar file. The actual name is Notes.jar.

    > The Web Server Plug-ins Configuration Tool verifies that the files exist but the tool does not validate either file.

    IBM HTTP Server
    > *IHS_root*/conf/httpd.conf

    Microsoft Internet Information Services (IIS)
    > The Web Server Plug-ins Configuration Tool can determine the correct files to edit.

    Sun Java System Web Server (formerly Sun ONE Web Server and iPlanet Web Server) Version 6.0 and later
    > obj.conf and magnus.conf

11. If you are configuring an IBM HTTP web server plug-in, perform the following actions.

    a. Optionally, set up the administration server configuration to administer the web server.
       1) Select **Setup IBM HTTP Server Administration Server**.
       2) Specify a port number on which the IBM HTTP administration server will communicate.
       3) Optionally, select **Create a user ID for IBM Server Administration Server authentication** and enter a user ID and password to authenticate to the IBM HTTP Server administrative server from the administrative console.

    b. Click **Next**.

    c. `AIX` `HP-UX` `Linux` `Solaris` Specify the system user ID and group to have write permission to IBM HTTP Server, the IBM HTTP Server administrative server, and the web server plug-in configuration files.

       Select **Create a new unique system user ID and group using the credentials** if necessary.

d. **Windows** Optionally, set up the IBM HTTP Server Administration Server to run as a Window service.

   1) Select **Run IBM HTTP Server Administration Server as a Windows Service**.

   2) Perform one of the following actions:

      - Select **Log on as a local system account**.

      - Select **Log on as a specified user account**, and enter the user ID and password for that account.

        The user ID requires the following advanced user rights:

        – Act as part of the operating system

        – Log on as a service

   3) Choose whether your startup type will be automatic or manual.

   e. Click **Next**.

12. Specify a unique name for the web server definition, and click **Next**.

13. Select the configuration scenario.

    a. Choose the local scenario.

    b. Perform one of the following actions:

       - Enter the installation location of WebSphere Application Server (*app_server_root*).

       - Click **Browse**, find the installation location of WebSphere Application Server (*app_server_root*), and click **OK**.

    c. Click **Next**.

14. Select the profile to configure with the current web server plug-in, and click **Next**.

15. Review the summary information, and click **Configure** to begin configuring the web server, web server plug-in, and application server profile.

16. Verify the success of the installation on the summary panel, and click **Finish**.

    If a problem occurs and the installation is unsuccessful, examine the logs in the *plugins_root*/`logs` directory. Correct any problems and re-configure.

17. **Domino Web Server only:** Set the WAS_PLUGIN_CONFIG_FILE environment variable.

    On platforms such as AIX or Linux, sourcing a script to the parent shell allows child processes to inherit the exported variables. On Windows systems, run the script as you would run any other command. Sourcing is automatic on Windows systems.

    a. Open a command window.

    b. Change directories to the plug-ins installation root directory.

    c. Issue the appropriate command for the *plugins_root*/`bin/setupPluginCfg.sh` script:

       - **AIX** **HP-UX** **Solaris** `. `*plugins_root*/`bin/setupPluginCfg.sh` (Notice the space between the period and the installation root directory.)

       - **Linux** `source `*plugins_root*/`bin/setupPluginCfg.sh`

    The script is also in the *lotus_root*/`notesdata` directory on operating systems such as AIX or Linux.

    Issue the appropriate command for the script before starting the Domino Web Server.

18. Start the Snoop servlet to verify the ability of the web server to retrieve an application from the application server.

    Test your environment by starting your application server, your web server, and using the Snoop servlet with an IP address.

    a. Start the application server.

       Change directories to the *profile_root*/`bin` directory and run the startServer command:

       - **AIX** **HP-UX** **Linux** **Solaris** `./startServer.sh server1`

       - **Windows** `startServer server1`

    b. Start the IBM HTTP Server or the web server that you are using.

Use a command window to change the directory to the IBM HTTP Server installed image, or to the installed image of your web server. Issue the appropriate command to start the web server, such as these commands for IBM HTTP Server:

To start the IBM HTTP Server from the command line:

Access the apache and apachectl commands in the `IBMHttpServer/bin` directory.

- `AIX` `HP-UX` `Linux` `Solaris` `./apachectl start`
- `Windows` `apache`

c. Point your browser to `http://localhost:9080/snoop` to test the internal HTTP transport provided by the application server. Point your browser to `http://Host_name_of_Web_server_machine/snoop` to test the web server plug-in.

The HTTP Transport port is 9080 by default and must be unique for every profile. The port is associated with a virtual host named default_host, which is configured to host the installed DefaultApplication. The Snoop servlet is part of the DefaultApplication. Change the port to match your actual HTTP Transport port.

d. Verify that Snoop is running.

Either web address should display the Snoop Servlet - Request/Client Information page.

> **Tip:** In the event of a verification failure where an HTTP error code of 500 appears, go to **IIS Manager > Default Web Site > sePlugins**. Right click, and choose to edit permissions. Click on the sharing tab, and choose to share with everyone (permissions level: read/write).

e. **Remote IBM HTTP Server only:**

Verify that the automatic propagation function can work on a remote IBM HTTP Server by using the following steps. This procedure is not necessary for local web servers.

1) Create a user=adminUser, password=adminPassword in the *IHS_root* `/conf/admin.passwd` file. For example: `c:\ws\ihs80\bin\htpasswd -cb c:\ws\ihs80\conf\admin.passwd adminUser adminPassword`

2) Use the administrative console of the application server to enter the User ID and password information that you created for the administrative user of IBM HTTP Server. Go to **Servers > Web server >** *web_server_definition* **> Remote web server administration**. Set the following values: admin Port=8008, User Id=adminUser, Password=adminPassword.

3) Set the correct read/write permissions for the `httpd.conf` file and the plugin-cfg.xml file. See the *IHS_root* `/logs/admin_ERROR.LOG` file for more information.

Automatic propagation of the plug-in configuration file requires the IBM HTTP administrative server to be up and running. If you are managing an IBM HTTP Server using the WebSphere Application Server administrative console, the following error might display:

`"Could not connect to IHS Administration server error"`

Perform the following procedure to correct the error:

1) Verify that the IBM HTTP Server administration server is running.

2) Verify that the web server host name and the port that is defined in the WebSphere Application Server administrative console matches the IBM HTTP Server administration host name and port.

3) Verify that the fire wall is not preventing you from accessing the IBM HTTP Server administration server from the WebSphere Application Server administrative console.

4) Verify that the user ID and password that is specified in the WebSphere Application Server administrative console under remote managed, is created in the `admin.passwd` file, using the htpasswd command.

5) If you are trying to connect securely, verify that you export the IBM HTTP Server administration server keydb personal certificate into the WebSphere Application Server key database as a signer certificate. This key database is specified by the com.ibm.ssl.trustStore directive in the `sas.client.props` file in the profile where your administrative console is running. This consideration is primarily for self-signed certificates.

6) If you still have problems, check the IBM HTTP Server `admin_ERROR.LOG` file and the WebSphere Application Server logs (`trace.log` file) to determine the cause of the problem.

Results

The installation of the Web Server Plug-ins results in the creation of the `Plugins` directory and several subdirectories. The following directories are among those created on a Linux system, for example:

- *plugins_root*/`bin` contains the binary plug-ins for all supported web servers
- *plugins_root*/`logs` contains log files
- *plugins_root*/`properties` contains version information

The Web Server Plug-ins Configuration Tool creates a web server definition within the application server profile unless one already exists.

The Web Server Plug-ins Configuration Tool configures the web server to use the *profile_root*/`plugin-cfg.xml` file.

The application server regenerates the web server plug-in configuration file, `plugin-cfg.xml` whenever an event occurs that affects the file. Such events include the addition or removal of an application, server, or virtual host. The standalone application server regenerates the file in the following location:

```
profile_root
   /config/cells/cell_name/nodes/
   web_server_name_node/servers/
   web_server_name/plugin-cfg.xml
```

What to do next

You can start a standalone application server and the web server immediately after configuring the plug-in for the local web server. Open the administrative console of the application server after you start the server and save the changed configuration.

See "Selecting a web server topology diagram and roadmap" on page 52 for an overview of the installation procedure.

See "Plug-ins configuration" on page 78 for information about the location of the plug-in configuration file.

See "Web server configuration" on page 34 for information about the files involved in configuring a web server.

See "Editing web server configuration files" on page 10 for information about how the Web Server Plug-ins Configuration Tool configures supported web servers.

See "Installing and configuring web server plug-ins" on page 50 for information about other installation scenarios for installing web server plug-ins.

Configuring a web server and an application server on separate machines (remote)

This topic describes installing a web server plug-in that WebSphere Application Server provides to communicate with a particular brand of web server. This procedure describes installing the web server and its web server plug-in for WebSphere Application Server on one machine and configuring the application server in the default profile on another machine to communicate with the web server.

Before you begin

When multiple profiles exist, you can select the profile that the Web Server Plug-ins Configuration Tool configures. See "Plug-ins configuration" on page 78 for a description of the flow of logic that determines how to select the profile to configure.

If the WebSphere Application Server product family supports a particular brand of web server, such as IBM HTTP Server or Microsoft Internet Information Services (IIS), your WebSphere Application Server product provides a binary plug-in for the web server that you must install.

If the WebSphere Application Server product family does not provide a binary plug-in for a particular brand of web server, then the web server is not supported. The purpose of the binary plug-in is to provide the communication protocol between the web server and the application server.

Suppose that you create a new profile and you also want to use a web server. You must install a new web server for the new profile, install the Web Server Plug-ins, and use the Web Server Plug-ins Configuration Tool to configure both the web server and the application server.

If the web server is not already installed, you can still install the Web Server Plug-ins for future use.

About this task

Installing the Web Server Plug-ins installs the plug-in module. The Web Server Plug-ins Configuration Tool configures the web server for communicating with the application server and creates a web server configuration definition in the application server, if possible.

Tip: As an alternative to using the Web Server Plug-ins Configuration Tool, you can use the pct command-line tool with a response file to configure a web server. Read "Configuring a web server plug-in using the WCT command-line utility" on page 109 for more information.

This procedure configures the application server profile that is the default profile on the machine. A one-to-one relationship exists between a web server and the application server.

This article describes how to create the following topology:
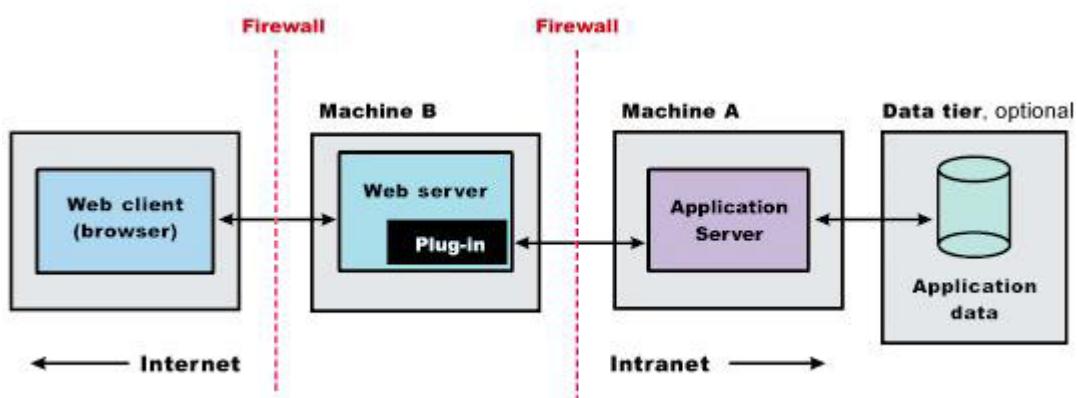
This article describes the installation of a web server on one machine and the application server on a separate machine. In this situation, the Web Server Plug-ins Configuration Tool on one machine cannot create the web server definition in the application server configuration on the other machine.

In such a case, the Web Server Plug-ins Configuration Tool creates a script on the web server machine that you can copy to the application server machine. Run the script on the application server machine to create the web server configuration definition within the application server configuration.

Perform the following procedure to install the plug-in and configure both the web server and the application server.

Procedure

1. Install Installation Manager on Machine A and Machine B.
2. Use Installation Manager to install WebSphere Application Server on Machine A.
3. Create a standalone application server on Machine A.
4. Optional: Create a new host alias for the default virtual host.

   If you configured the web server to use a port other than port 80, then you must add a new host alias for that port for the default host. For example, when running as nonroot, IBM HTTP Server is configured with a default port value of 8080.
5. Use Installation Manager to install the following on Machine B.
   - Web Server Plug-ins for WebSphere Application Server
   - Websphere Customization Toolbox
6. Use Installation Manager to install the IBM HTTP Server on Machine B, or install another supported web server on Machine B.
7. Open the WebSphere Customization Toolbox, and launch the Web Server Plug-ins Configuration Tool on the machine with the web server.
8. Select a web server plug-in runtime location.

   If the location of a previously installed web server that you want to use is not in the list, perform the following actions to add the location to your working set:
   a. Click **Add**.
   b. Enter a name for the web server plug-in location.
   c. Perform one of the following actions:
      - Enter the location.
      - Click **Browse**, find the location, and click **OK**.
9. Click **Create**.
10. Select the type of web server that you are configuring, and click **Next**.
11. Select the architecture of your installed target web server (64 bit or 32 bit) and click **Next** if you are asked.
12. Click **Browse** to select the configuration file or files for your web server, verify that the web server port is correct, and then click **Next** when you are finished.

    Select the file and not just the directory of the file. Some web servers have two configuration files and require you to browse for each file.

    The following list shows configuration files for supported web servers:

    Apache HTTP Server
    > *apache_root*/config/httpd.conf

    Domino Web Server
    > names.nsf and Notes.jar

    > The wizard prompts for the notes.jar file. The actual name is Notes.jar.

    > The Web Server Plug-ins Configuration Tool verifies that the files exist but the tool does not validate either file.

    IBM HTTP Server
    > *IHS_root*/conf/httpd.conf

Microsoft Internet Information Services (IIS)
> The Web Server Plug-ins Configuration Tool can determine the correct files to edit.

Sun Java System Web Server (formerly Sun ONE Web Server and iPlanet Web Server) Version 6.0 and later
> `obj.conf` and `magnus.conf`

13. If you are configuring an IBM HTTP web server plug-in, perform the following actions.

    a. Optionally, set up the administration server configuration to administer the web server.

        1) Select **Setup IBM HTTP Server Administration Server**.

        2) Specify a port number on which the IBM HTTP administration server will communicate.

        3) Optionally, select **Create a user ID for IBM Server Administration Server authentication** and enter a user ID and password to authenticate to the IBM HTTP Server administrative server from the administrative console.

    b. Click **Next**.

    c. ▮AIX▮ ▮HP-UX▮ ▮Linux▮ ▮Solaris▮ Specify the system user ID and group to have write permission to IBM HTTP Server, the IBM HTTP Server administrative server, and the web server plug-in configuration files.

        Select **Create a new unique system user ID and group using the credentials** if necessary.

    d. ▮Windows▮ Optionally, set up the IBM HTTP Server Administration Server to run as a Window service.

        1) Select **Run IBM HTTP Server Administration Server as a Windows Service**.

        2) Perform one of the following actions:

            • Select **Log on as a local system account**.

            • Select **Log on as a specified user account**, and enter the user ID and password for that account.

              The user ID requires the following advanced user rights:

              – Act as part of the operating system

              – Log on as a service

        3) Choose whether your startup type will be automatic or manual.

    e. Click **Next**.

14. Specify a unique name for the web server definition, and click **Next**.

15. Select the configuration scenario.

    a. Choose the remote scenario.

    b. Identify the host name or IP address of Machine A, which is the application server machine.

    c. Click **Next**.

16. Select the profile to configure with the current web server plug-in, and click **Next**.

    This panel does not display if you selected the remote scenario in the previous step.

17. Examine the summary panel, and click **Configure** to begin configuring.

    The panel notifies you that you have manual steps to perform to complete the installation and configuration.

    The Web Server Plug-ins Configuration Tool creates the `configure`*web_server_name* script in the *plugins_root*/`bin`/ directory on Machine B (the machine with the web server).

    The Web Server Plug-ins Configuration Tool also creates the `plugin-cfg.xml` file in the *plugins_root*/`config`/*web_server_name* directory.

    The web server reads the `plugin-cfg.xml` file to determine the applications that the application server on Machine A can serve to the web server on Machine B. Whenever the configuration changes, the application server regenerates the file. When regeneration occurs, propagate, or copy the actual `plugin-cfg.xml` file from the application server machine to the web server machine. You can automatically propagate the file to the IBM HTTP Server product.

18. Verify the success of the installation on the summary panel, and click **Finish**.

    If a problem occurs and the installation is unsuccessful, examine the logs in the *plugins_root*/`logs` directory. Correct any problems and re-configure.

19. Copy the `configure`*web_server_name* script from Machine B (the machine with the web server) to the *app_server_root* /`bin` directory on Machine A (the application server machine).

    web_server_name is the nickname of the web server that you specified. *web_server_name* is not a vendor name, such as IIS or Apache.

    On an operating system such as AIX or Linux, the file is `configure`*web_server_name*`.sh`. On a Windows system, the file is `configure`*web_server_name*`.bat`. For example, on a Linux system with an IBM HTTP Server named web_server_1 in the default location, copy *plugins_root*/`bin`/`configure`*web_server_1*`.sh` from Machine B (the machine with the web server) to the *app_server_root*/`bin` directory on Machine A (the application server machine).

    If one platform is a system such as AIX or Linux and the other is a Windows platform, copy the script from the `crossPlatformScripts` directory. For example:

    - ◼ **AIX** ◼ **HP-UX** ◼ **Linux** ◼ **Solaris** *plugins_root*/`bin/configure`*web_server_name*`.sh`
    - ◼ **Windows** *plugins_root*/`bin/crossPlatformScripts/windows/configure`*web_server_name*`.bat`

20. Compensate for file encoding differences to prevent script failure.

    The content of the `configure`*web_server_name*`.bat` script or the `configure`*web_server_name*`.sh` script can be corrupt if the default file encoding of the two machines differs. This scenario is possible when one machine is set up for a double-byte character set (DBCS) locale and the other machine is not.

    Determine the file encoding and use one of the following procedures to circumvent the failure. To determine the default file encoding, run the appropriate command.

    - Run the locale charmap command on a system such as AIX or Linux.

    - Run the CHCP command on a Windows machine.

    Use the result of the command on each machine as the value for the *web_server_machine_encoding* variable and the *application_server_machine_encoding* variable in one of the following procedures.

    Procedures for compensating for encoding differences

    - **Web server running on a system such as AIX or Linux**

      Suppose that the web server is running on a Linux machine and the application server is running on a Windows machine. Before you FTP the web server definition configuration script to the Windows machine in binary mode, run the following command on the system to encode the file:

```
iconv -f web_server_machine_encoding \
   -t application_server_machine_encoding \
   configureweb_server_name.bat
```

      Important: The name of the web server (nick name) is used in the name of the script file. The name cannot contain characters from a double-byte character set (DBCS) if you intend to set up IBM HTTP Server for automatic propagation.

    - **Web server running on a Windows system**

      Suppose that the web server is running on a Windows machine and the application server is running on a machine with a system such as AIX or Linux. You must first download the iconv utility from a third party because the command is not included by default on Windows systems. Before you FTP the web server definition configuration script in binary mode to a system such as AIX or Linux, run the following command on the machine to encode the file:

```
iconv -f web_server_machine_encoding \
   -t application_server_machine_encoding \
   configureweb_server_name.sh
```

      For example, if the target machine is z/OS, you might use this command to convert the file from ASCII to EBCDIC, handling the end-of-line characters correctly:

```
iconv -f ISO8859-1 -t IBM-1047 configureweb_server_name.sh > new_script_name.sh
```

      Omit the continuation characters (\) if you enter the command on one line.

If the conversion mapping is not supported by the iconv command on your system, copy the contents of the web server configuration script to a clip board and paste it onto the machine where the application server is running.

Note: If you copy over a `.sh` file onto a UNIX-based operating system after remote configuration on a Windows operating system, you must perform chmod 755.

21. Start the application server on Machine A.

    Use the startServer command, for example:

    - `AIX` `HP-UX` `Linux` `Solaris` *profile_root*/bin/startServer.sh server1
    - `Windows` *profile_root*\bin\startServer server1

22. Open a command window and change to the profile directory where the web server should be assigned. Run the script that you copied to Machine A (the application server machine). You need the following parameters:

    - Profile Name
    - (Optional) Admin user ID
    - (Optional) Admin user password

    For example, you could enter the following:

```
configurewebserver1.sh AppSrv01 my_user_ID my_Password
```

    The web server will be configured via wsadmin.

    The contents of the configurewebserver1.sh script will be similar to this:

```
wsadmin.bat -profileName AppSrv01 -user my_user_ID -password my_Password
  -f "%WAS_HOME%\bin\configureWebserverDefinition.jacl" webserver1 IHS..
```

23. **Domino Web Server only:** Set the WAS_PLUGIN_CONFIG_FILE environment variable.

    On platforms such as AIX or Linux, sourcing a script to the parent shell allows child processes to inherit the exported variables. On Windows systems, run the script as you would run any other command. Sourcing is automatic on Windows systems.

    a. Open a command window.
    b. Change directories to the plug-ins installation root directory.
    c. Issue the appropriate command for the *plugins_root*/bin/setupPluginCfg.sh script:

       - `AIX` `HP-UX` `Solaris` . *plugins_root*/bin/setupPluginCfg.sh (Notice the space between the period and the installation root directory.)
       - `Linux` source *plugins_root*/bin/setupPluginCfg.sh

    The script is also in the *lotus_root*/notesdata directory on operating systems such as AIX or Linux.

    Issue the appropriate command for the script before starting the Domino Web Server.

24. Regenerate the `plugin-cfg.xml` file on Machine A (the application server machine) using the administrative console. Click **Servers > Server Types > Web servers**. Select the web server, then click **Generate Plug-in**.

    During the installation of the plug-ins, the default `plugin-cfg.xml` file is installed on Machine B (the machine with the web server) in the *plugins_root*/config/*web_server_name* directory. The web server plug-in configuration service regenerates the `plugin-cfg.xml` file automatically. To use the current `plugin-cfg.xml` file from the application server, propagate the `plugin-cfg.xml` file as described in the next step.

    This step shows you how to regenerate the `plugin-cfg.xml` file. WebSphere Application Server products are configured to automatically regenerate the file each time a significant event occurs. Such events include installing applications on the application server and the web server, for example. Creating a new virtual host is another such event.

25. Propagate the `plugin-cfg.xml` file from the application server to the web server using the administrative console. Click **Servers > Web server**. Select the web server, then click **Propagate Plug-in**. Web servers other than IBM HTTP Server require manual propagation.

The web server plug-in configuration service propagates the `plugin-cfg.xml` file automatically for IBM HTTP Server 8.0 only. For all other web servers, propagate the plug-in configuration file by manually copying the `plugin-cfg.xml` file from the *profile_root*/`config/cells/`*cell_name*/`nodes/`*node_name*/ `servers/`*web_server_name* directory on Machine A (the application server machine) to the *plugins_root*/`config/`*web_server_name* directory on Machine B (the machine with the web server).

26. Start the Snoop servlet to verify the ability of the web server to retrieve an application from the application server.

Test your environment by starting your application server, your web server, and using the Snoop servlet with an IP address.

a. Start the application server.

Change directories to the *profile_root*/`bin` directory and run the startServer command:
- **AIX** **HP-UX** **Linux** **Solaris** `./startServer.sh server1`
- **Windows** `startServer server1`

b. Start the IBM HTTP Server or the web server that you are using.

Use a command window to change the directory to the IBM HTTP Server installed image, or to the installed image of your web server. Issue the appropriate command to start the web server, such as these commands for IBM HTTP Server:

To start the IBM HTTP Server from the command line:

Access the apache and apachectl commands in the `IBMHttpServer/bin` directory.
- **AIX** **HP-UX** **Linux** **Solaris** `./apachectl start`
- **Windows** `apache`

c. Point your browser to `http://localhost:9080/snoop` to test the internal HTTP transport provided by the application server. Point your browser to `http://`*Host_name_of_Web_server_machine*`/snoop` to test the web server plug-in.

The HTTP Transport port is 9080 by default and must be unique for every profile. The port is associated with a virtual host named default_host, which is configured to host the installed DefaultApplication. The Snoop servlet is part of the DefaultApplication. Change the port to match your actual HTTP Transport port.

d. Verify that Snoop is running.

Either Web address should display the Snoop Servlet - Request/Client Information page.

e. **Remote IBM HTTP Server only:**

Verify that the automatic propagation function can work on a remote IBM HTTP Server by using the following steps. This procedure is not necessary for local web servers.

1) Create a user=adminUser, password=adminPassword in the *IHS_root* /`conf/admin.passwd` file. For example: `c:\ws\ihs80\bin\htpasswd -cb c:\ws\ihs80\conf\admin.passwd adminUser adminPassword`

2) Use the administrative console of the application server to enter the User ID and password information that you created for the administrative user of IBM HTTP Server. Go to **Servers > Web server >** *web_server_definition* **> Remote Web server administration**. Set the following values: admin Port=8008, User Id=adminUser, Password=adminPassword.

3) Set the correct read/write permissions for the `httpd.conf` file and the plugin-cfg.xml file. See the *IHS_root* /`logs/admin_ERROR. LOG` file for more information.

Automatic propagation of the plug-in configuration file requires the IBM HTTP administrative server to be up and running. If you are managing an IBM HTTP Server using the WebSphere Application Server administrative console, the following error might display:

`"Could not connect to IHS Administration server error"`

Perform the following procedure to correct the error:

1) Verify that the IBM HTTP Server administration server is running.

2) Verify that the web server host name and the port that is defined in the WebSphere Application Server administrative console matches the IBM HTTP Server administration host name and port.

3) Verify that the fire wall is not preventing you from accessing the IBM HTTP Server administration server from the WebSphere Application Server administrative console.

4) Verify that the user ID and password that is specified in the WebSphere Application Server administrative console under remote managed, is created in the `admin.passwd` file, using the htpasswd command.

5) If you are trying to connect securely, verify that you export the IBM HTTP Server administration server keydb personal certificate into the WebSphere Application Server key database as a signer certificate. This key database is specified by the com.ibm.ssl.trustStore directive in the `sas.client.props` file in the profile where your administrative console is running. This consideration is primarily for self-signed certificates.

6) If you still have problems, check the IBM HTTP Server `admin_ERROR. LOG` file and the WebSphere Application Server logs (`trace.log` file) to determine the cause of the problem.

Results

This procedure results in the installation of the Web Server Plug-ins for WebSphere Application Server on a web server machine. The Web Server Plug-ins Configuration Tool also configures the web server to support an application server on a separate machine.

The installation of the Web Server Plug-ins results in the creation of the `Plugins` directory and several subdirectories. The following directories are among those created on a Linux system, for example:
- *plugins_root*/`bin` contains the binary plug-ins for all supported web servers
- *plugins_root*/`logs` contains log files
- *plugins_root*/`properties` contains version information

What to do next

See "Selecting a web server topology diagram and roadmap" on page 52 for an overview of the installation procedure.

See "Web server configuration" on page 34 for more information about the files involved in configuring a web server.

See "Plug-ins configuration" on page 78 for information about the location of the plug-in configuration file.

See "Editing web server configuration files" on page 10 for information about how the Web Server Plug-ins Configuration Tool configures supported web servers.

Configuring multiple web servers and remote standalone application servers

This topic describes the procedure of installing and configuring multiple web servers and application servers on separate machines.

Before you begin

When multiple profiles exist, you can select the profile that the Web Server Plug-ins Configuration Tool configures. See "Plug-ins configuration" on page 78 for a description of the flow of logic that determines how to select the profile to configure.

If the WebSphere Application Server product family supports a particular brand of web server, such as IBM HTTP Server or Microsoft Internet Information Services (IIS), your WebSphere Application Server product provides a binary plug-in for the web server that you must install.

If the WebSphere Application Server product family does not provide a binary plug-in for a particular brand of web server, then the web server is not supported. The purpose of the binary plug-in is to provide the communication protocol between the web server and the application server.

Suppose that you create a new profile and you also want to use a web server. You must install a new web server for the new profile, install the Web Server Plug-ins, and use the Web Server Plug-ins Configuration Tool to configure both the web server and the application server.

If the web server is not already installed, you can still install the Web Server Plug-ins for future use.

About this task

Installing the Web Server Plug-ins installs the plug-in module. The Web Server Plug-ins Configuration Tool configures the web server for communicating with the application server and creates a web server configuration definition in the application server, if possible.

This article describes how to create the following topology:
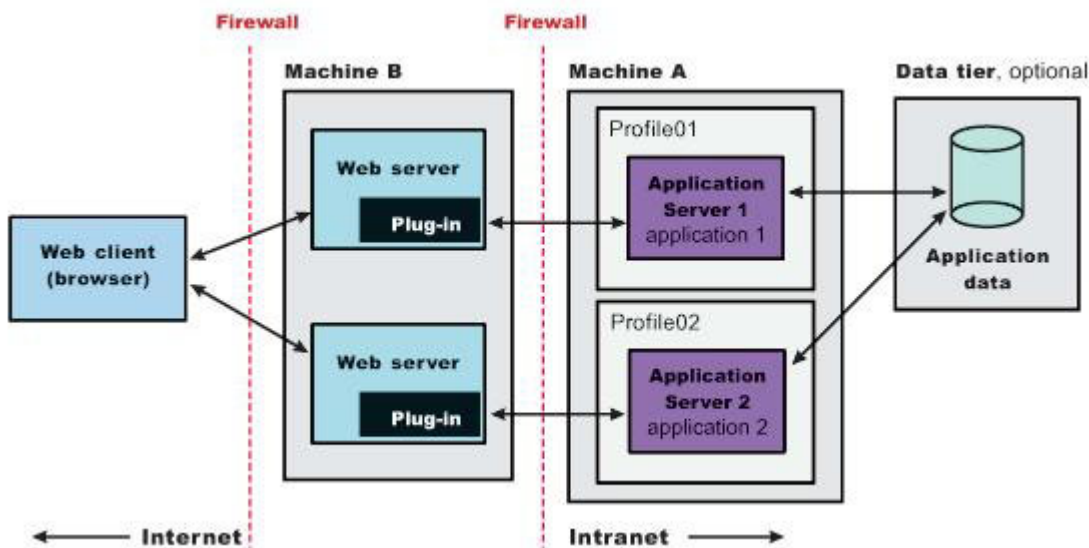
Perform the following procedure to install the plug-ins and configure both web servers and both application servers.

This topology lets each profile have unique applications, configuration settings, data, and log files, while sharing the same set of system files. Creating multiple profiles creates multiple application server environments that you can then dedicate to different purposes.

For example, each application server on a website can serve a different application. In another example, each application server can be a separate test environment that you assign to a programmer or a development team.

Tip: As an alternative to using the Web Server Plug-ins Configuration Tool, you can use the pct command-line tool with a response file to configure a web server. Read "Configuring a web server plug-in using the WCT command-line utility" on page 109 for more information.

Procedure

1. Install Installation Manager on Machine A and Machine B.
2. Use Installation Manager to install WebSphere Application Server on Machine A.
3. Create the first application server profile using the Profile Management Tool on Machine A.
4. Use Installation Manager to install the following on Machine B.
   - Web Server Plug-ins for WebSphere Application Server
   - Websphere Customization Toolbox
5. Use Installation Manager to install the IBM HTTP Server on Machine B, or install another supported web server on Machine B.
6. Open the WebSphere Customization Toolbox, and launch the Web Server Plug-ins Configuration Tool on the machine with the web server.
7. Select a web server plug-in runtime location.

   If the location of a previously installed web server that you want to use is not in the list, perform the following actions to add the location to your working set:

   a. Click **Add**.
   b. Enter a name for the web server plug-in location.
   c. Perform one of the following actions:
      - Enter the location.
      - Click **Browse**, find the location, and click **OK**.
8. Click **Create**.
9. Select the type of web server that you are configuring, and click **Next**.
10. Select the architecture of your installed target web server (64 bit or 32 bit) and click **Next** if you are asked.
11. Click **Browse** to select the configuration file or files for your web server, verify that the web server port is correct, and then click **Next** when you are finished.

    Select the file and not just the directory of the file. Some web servers have two configuration files and require you to browse for each file.

    The following list shows configuration files for supported web servers:

    Apache HTTP Server
    > *apache_root*/`config/httpd.conf`

    Domino Web Server
    > `names.nsf` and `Notes.jar`

    > The wizard prompts for the `notes.jar` file. The actual name is `Notes.jar`.

    > The Web Server Plug-ins Configuration Tool verifies that the files exist but the tool does not validate either file.

    IBM HTTP Server
    > *IHS_root*/`conf/httpd.conf`

    Microsoft Internet Information Services (IIS)
    > The Web Server Plug-ins Configuration Tool can determine the correct files to edit.

    Sun Java System Web Server (formerly Sun ONE Web Server and iPlanet Web Server) Version 6.0 and later
    > `obj.conf` and `magnus.conf`
12. If you are configuring an IBM HTTP web server plug-in, perform the following actions.
    a. Optionally, set up the administration server configuration to administer the web server.
       1) Select **Setup IBM HTTP Server Administration Server**.
       2) Specify a port number on which the IBM HTTP administration server will communicate.

3) Optionally, select **Create a user ID for IBM Server Administration Server authentication** and enter a user ID and password to authenticate to the IBM HTTP Server administrative server from the administrative console.

b. Click **Next**.

c.   **AIX**   **HP-UX**   **Linux**   **Solaris**   Specify the system user ID and group to have write permission to IBM HTTP Server, the IBM HTTP Server administrative server, and the web server plug-in configuration files.

Select **Create a new unique system user ID and group using the credentials** if necessary.

d.   **Windows**   Optionally, set up the IBM HTTP Server Administration Server to run as a Window service.

1) Select **Run IBM HTTP Server Administration Server as a Windows Service**.

2) Perform one of the following actions:

- Select **Log on as a local system account**.
- Select **Log on as a specified user account**, and enter the user ID and password for that account.

  The user ID requires the following advanced user rights:

  – Act as part of the operating system

  – Log on as a service

3) Choose whether your startup type will be automatic or manual.

e. Click **Next**.

13. Specify a unique name for the web server definition, and click **Next**.

14. Select the configuration scenario.

a. Choose the remote scenario.

b. Identify the host name or IP address of Machine A, which is the application server machine.

c. Click **Next**.

15. Select the profile to configure with the current web server plug-in, and click **Next**.

16. Examine the summary panel, and click **Configure** to begin configuring.

The panel notifies you that you have manual steps to perform to complete the installation and configuration.

The Web Server Plug-ins Configuration Tool creates the `configureweb_server_name` script in the *plugins_root*/`bin`/ directory on Machine B (the machine with the web server).

The Web Server Plug-ins Configuration Tool also creates the `plugin-cfg.xml` file in the *plugins_root*/`config`/*web_server_name* directory.

The web server reads the `plugin-cfg.xml` file to determine the applications that the application server on Machine A can serve to the web server on Machine B. Whenever the configuration changes, the application server regenerates the file. When regeneration occurs, propagate, or copy the actual `plugin-cfg.xml` file from the application server machine to the web server machine. You can automatically propagate the file to the IBM HTTP Server product.

17. Verify the success of the installation on the summary panel, and click **Finish**.

If a problem occurs and the installation is unsuccessful, examine the logs in the *plugins_root*/`logs` directory. Correct any problems and re-configure.

18. Copy the `configureweb_server_name` script from Machine B (the machine with the web server) to the *app_server_root* /`bin` directory on Machine A (the application server machine).

web_server_name is the nickname of the web server that you specified. *web_server_name* is not a vendor name, such as IIS or Apache.

On an operating system such as AIX or Linux, the file is `configureweb_server_name.`sh. On a Windows system, the file is `configureweb_server_name.`bat. For example, on a Linux system with an IBM HTTP Server named web_server_1 in the default location, copy *plugins_root*/`bin`/

`configureweb_server_1.sh` from Machine B (the machine with the web server) to the *app_server_root*/bin directory on Machine A (the application server machine).

If one platform is a system such as AIX or Linux and the other is a Windows platform, copy the script from the `crossPlatformScripts` directory. For example:

- **AIX** **HP-UX** **Linux** **Solaris** *plugins_root*/bin/configure*web_server_name*.sh
- **Windows** *plugins_root*/bin/crossPlatformScripts/windows/configure*web_server_name*.bat

19. Compensate for file encoding differences to prevent script failure.

    The content of the `configureweb_server_name`.bat script or the `configureweb_server_name`.sh script can be corrupt if the default file encoding of the two machines differs. This scenario is possible when one machine is set up for a double-byte character set (DBCS) locale and the other machine is not.

    Determine the file encoding and use one of the following procedures to circumvent the failure. To determine the default file encoding, run the appropriate command.

    - Run the locale charmap command on a system such as AIX or Linux.
    - Run the CHCP command on a Windows machine.

    Use the result of the command on each machine as the value for the *web_server_machine_encoding* variable and the *application_server_machine_encoding* variable in one of the following procedures.

    Procedures for compensating for encoding differences

    - **Web server running on a system such as AIX or Linux**

      Suppose that the web server is running on a Linux machine and the application server is running on a Windows machine. Before you FTP the web server definition configuration script to the Windows machine in binary mode, run the following command on the system to encode the file:

```
iconv -f web_server_machine_encoding \
  -t application_server_machine_encoding \
  configureweb_server_name.bat
```

      Important: The name of the web server (nick name) is used in the name of the script file. The name cannot contain characters from a double-byte character set (DBCS) if you intend to set up IBM HTTP Server for automatic propagation.

    - **Web server running on a Windows system**

      Suppose that the web server is running on a Windows machine and the application server is running on a machine with a system such as AIX or Linux. You must first download the iconv utility from a third party because the command is not included by default on Windows systems. Before you FTP the web server definition configuration script in binary mode to a system such as AIX or Linux, run the following command on the machine to encode the file:

```
iconv -f web_server_machine_encoding \
  -t application_server_machine_encoding \
  configureweb_server_name.sh
```

      For example, if the target machine is z/OS, you might use this command to convert the file from ASCII to EBCDIC, handling the end-of-line characters correctly:

```
iconv -f ISO8859-1 -t IBM-1047 configureweb_server_name.sh > new_script_name.sh
```

      Omit the continuation characters (\) if you enter the command on one line.

      If the conversion mapping is not supported by the iconv command on your system, copy the contents of the web server configuration script to a clip board and paste it onto the machine where the application server is running.

      Note: If you copy over a .sh file onto a UNIX-based operating system after remote configuration on a Windows operating system, you must perform chmod 755.

20. Start the application server on Machine A.

    Use the startServer command, for example:

    - **AIX** **HP-UX** **Linux** **Solaris** *profile_root*/bin/startServer.sh server1
    - **Windows** *profile_root*\bin\startServer server1

21. Open a command window and change to the profile directory where the web server should be assigned. Run the script that you copied to Machine A (the application server machine). You need the following parameters:

    - Profile Name
    - (Optional) Admin user ID
    - (Optional) Admin user password

    For example, you could enter the following:

```
configurewebserver1.sh AppSrv01 my_user_ID my_Password
```

    The web server will be configured via wsadmin.

    The contents of the configurewebserver1.sh script will be similar to this:

```
wsadmin.bat -profileName AppSrv01 -user my_user_ID -password my_Password
   -f "%WAS_HOME%\bin\configureWebserverDefinition.jacl" webserver1 IHS..
```

22. **Domino Web Server only:** Set the WAS_PLUGIN_CONFIG_FILE environment variable.

    On platforms such as AIX or Linux, sourcing a script to the parent shell allows child processes to inherit the exported variables. On Windows systems, run the script as you would run any other command. Sourcing is automatic on Windows systems.

    a. Open a command window.
    b. Change directories to the plug-ins installation root directory.
    c. Issue the appropriate command for the *plugins_root*/bin/setupPluginCfg.sh script:

        - **AIX** **HP-UX** **Solaris** `. plugins_root`/bin/setupPluginCfg.sh (Notice the space between the period and the installation root directory.)
        - **Linux** source *plugins_root*/bin/setupPluginCfg.sh

    The script is also in the *lotus_root*/notesdata directory on operating systems such as AIX or Linux.

    Issue the appropriate command for the script before starting the Domino Web Server.

23. Regenerate the `plugin-cfg.xml` file on Machine A (the application server machine) using the administrative console. Click **Servers > Server Types > Web servers**. Select the web server, then click **Generate Plug-in**.

    During the installation of the plug-ins, the default `plugin-cfg.xml` file is installed on Machine B (the machine with the web server) in the *plugins_root*/config/*web_server_name* directory. The web server plug-in configuration service regenerates the `plugin-cfg.xml` file automatically. To use the current `plugin-cfg.xml` file from the application server, propagate the `plugin-cfg.xml` file as described in the next step.

    This step shows you how to regenerate the `plugin-cfg.xml` file. WebSphere Application Server products are configured to automatically regenerate the file each time a significant event occurs. Such events include installing applications on the application server and the web server, for example. Creating a new virtual host is another such event.

24. Propagate the `plugin-cfg.xml` file from the application server to the web server using the administrative console. Click **Servers > Web server**. Select the web server, then click **Propagate Plug-in**. Web servers other than IBM HTTP Server require manual propagation.

    The web server plug-in configuration service propagates the `plugin-cfg.xml` file automatically for IBM HTTP Server 8.0 only. For all other web servers, propagate the plug-in configuration file by manually copying the `plugin-cfg.xml` file from the *profile_root*/config/cells/*cell_name*/nodes/*node_name*/servers/*web_server_name* directory on Machine A (the application server machine) to the *plugins_root*/config/*web_server_name* directory on Machine B (the machine with the web server).

25. Start the Snoop servlet to verify the ability of the web server to retrieve an application from the application server.

    Test your environment by starting your application server, your web server, and using the Snoop servlet with an IP address.

    a. Start the application server.

Change directories to the *profile_root*/`bin` directory and run the startServer command:

- `AIX` `HP-UX` `Linux` `Solaris` `./startServer.sh server1`
- `Windows` `startServer server1`

b. Start the IBM HTTP Server or the web server that you are using.

Use a command window to change the directory to the IBM HTTP Server installed image, or to the installed image of your web server. Issue the appropriate command to start the web server, such as these commands for IBM HTTP Server:

To start the IBM HTTP Server from the command line:

Access the apache and apachectl commands in the `IBMHttpServer/bin` directory.

- `AIX` `HP-UX` `Linux` `Solaris` `./apachectl start`
- `Windows` `apache`

c. Point your browser to `http://localhost:9080/snoop` to test the internal HTTP transport provided by the application server. Point your browser to `http://`*Host_name_of_Web_server_machine*`/snoop` to test the web server plug-in.

The HTTP Transport port is 9080 by default and must be unique for every profile. The port is associated with a virtual host named default_host, which is configured to host the installed DefaultApplication. The Snoop servlet is part of the DefaultApplication. Change the port to match your actual HTTP Transport port.

d. Verify that Snoop is running.

Either Web address should display the Snoop Servlet - Request/Client Information page.

e. **Remote IBM HTTP Server only:**

Verify that the automatic propagation function can work on a remote IBM HTTP Server by using the following steps. This procedure is not necessary for local web servers.

1) Create a user=adminUser, password=adminPassword in the *IHS_root* /conf/`admin.passwd` file. For example: `c:\ws\ihs80\bin\htpasswd -cb c:\ws\ihs80\conf\admin.passwd adminUser adminPassword`

2) Use the administrative console of the application server to enter the User ID and password information that you created for the administrative user of IBM HTTP Server. Go to **Servers > Web server >** *web_server_definition* **> Remote Web server administration**. Set the following values: admin Port=8008, User Id=adminUser, Password=adminPassword.

3) Set the correct read/write permissions for the `httpd.conf` file and the plugin-cfg.xml file. See the *IHS_root* /logs/`admin_ERROR. LOG` file for more information.

Automatic propagation of the plug-in configuration file requires the IBM HTTP administrative server to be up and running. If you are managing an IBM HTTP Server using the WebSphere Application Server administrative console, the following error might display:

`"Could not connect to IHS Administration server error"`

Perform the following procedure to correct the error:

1) Verify that the IBM HTTP Server administration server is running.

2) Verify that the web server host name and the port that is defined in the WebSphere Application Server administrative console matches the IBM HTTP Server administration host name and port.

3) Verify that the fire wall is not preventing you from accessing the IBM HTTP Server administration server from the WebSphere Application Server administrative console.

4) Verify that the user ID and password that is specified in the WebSphere Application Server administrative console under remote managed, is created in the `admin.passwd` file, using the htpasswd command.

5) If you are trying to connect securely, verify that you export the IBM HTTP Server administration server keydb personal certificate into the WebSphere Application Server key database as a signer certificate. This key database is specified by the com.ibm.ssl.trustStore

directive in the `sas.client.props` file in the profile where your administrative console is running. This consideration is primarily for self-signed certificates.

   6) If you still have problems, check the IBM HTTP Server `admin_ERROR. LOG` file and the WebSphere Application Server logs (`trace.log` file) to determine the cause of the problem.

26. Create the second application server profile using the Profile Management Tool on Machine A. Make the profile the default profile during the profile creation by selecting the check box on the appropriate panel.

27. Install a second IBM HTTP Server or another supported web server on Machine B.

28. On Machine B, configure the second web server using the Web Server Plug-ins Configuration Tool. Both web servers share a single installation of the plug-in binaries but must be configured individually.

29. The Installation Manager creates a script named `configureweb_server_name` for the second web server. The script is in the *plugins_root*/`bin` directory on Machine B. Copy the script to the *app_server_root*/`bin` directory on Machine A.

30. Start the second application server.

31. Run the `configureweb_server_name` script on Machine A to create a web server definition in the administrative console. You can then use the administrative console to manage the web server.

32. Propagate the `plugin-cfg.xml` file from the second application server to the web server using the administrative console. Click **Servers > Web server > Propagate Plug-in**. Web servers other than IBM HTTP Server require manual propagation.

33. Run the Snoop servlet on the second web server to verify that it is operational.

Results

This procedure results in installing two or more application servers on one machine and installing dedicated web servers on another machine. This procedure installs the Web Server Plug-ins for both web servers and configures both web servers and both application servers.

What to do next

See "Selecting a web server topology diagram and roadmap" on page 52 for an overview of the installation procedure.

See "Editing web server configuration files" on page 10 for information about how the Web Server Plug-ins Configuration Tool configures supported web servers.

See "Web server configuration" on page 34 for more information about the files involved in configuring a web server.

For IHS web servers, you can stop and start the web server and propagate the `plugin-cfg.xml` file from the WebSphere Application Server machine to the web server machine. For all other web servers, you cannot start/stop or propagate the `plugin-cfg.xml` file in the admin console. You will need to propagate the `plugin-cfg.xml` file manually. The following three steps describes how to perform manual propagation:

1. After completion of configuration with web servers other than IHS 6.x, verify that the `plugin-cfg.xml` file exists at `<WAS_HOME>/profiles/<PROFILE_HOME>/config/cells/<CELL_NAME>/nodes/<SERVER_NAME>/servers/<WEBSERVER_DEFINITION>`

2. Transfer the above `plugin-cfg.xml` to replace `<PLUGIN_HOME>/config/<WEBSERVER_DEFINITION>/plugin-xfg.xml`

3. Restart the web server and corresponding profile.

Configuring a web server and a custom profile on the same machine

This procedure describes installing a web server and its plug-in on a machine where the default profile is a custom profile.

Before you begin

When multiple profiles exist, you can select the profile that the Web Server Plug-ins Configuration Tool configures. See "Plug-ins configuration" on page 78 for a description of the flow of logic that determines how to select the profile to configure.
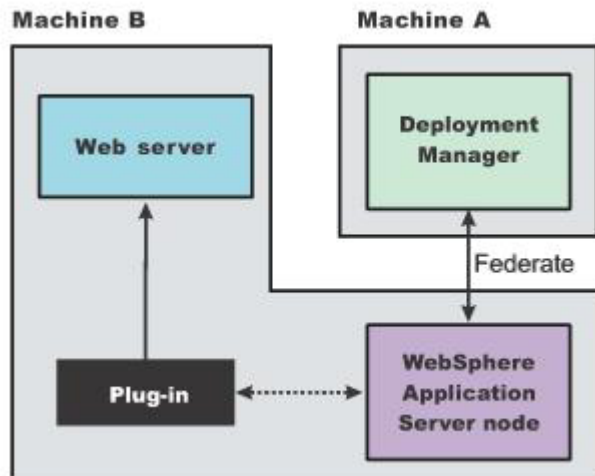
This procedure configures the custom profile on Machine B. This procedure assumes that you already have installed a deployment manager on Machine A.

The WebSphere Application Server node on Machine B is the custom node that you create in this procedure. This procedure starts the deployment manager and federates the custom node before installing the Web Server Plug-ins.

Start the deployment manager. The deployment manager must be running to successfully federate and configure the custom node.

About this task

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.

Use this procedure to install the web server plug-in, configure the web server, and create a web server definition in the custom profile (custom node).

Tip: As an alternative to using the Web Server Plug-ins Configuration Tool, you can use the pct command-line tool with a response file to configure a web server. Read "Configuring a web server plug-in using the WCT command-line utility" on page 109 for more information.

Procedure

1. Log on to the operating system.

   If you are installing as a nonroot or non-administrative user, then there are certain limitations.

AIX **HP-UX** **Linux** **Solaris** In addition, select a umask that allows the owner to read/write to the files, and allows others to access them according to the prevailing system policy. For root, a umask of 022 is recommended. For nonroot users, a umask of 002 or 022 could be used, depending on whether or not the users share the group. To verify the umask setting, issue the following command:

```
umask
```

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

```
umask 022
```

Windows When installing as an administrative user on a Windows operating system, a Windows service is automatically created to autostart the application server. The installer user account must have the following advanced user rights:

- Act as part of the operating system
- Log on as a service

For example, on some Windows operating systems, click **Control Panel > Administrative Tools > Local Security Policy > Local Policies > User Rights Assignments** to set the advanced options. See your Windows operating system documentation for more information.

Windows If you plan to run the application server as a Windows service, do not install from a user ID that contains spaces. A user ID with spaces cannot be validated. Such a user ID is not allowed to continue the installation. To work around this restriction, install with a user ID that does not contain spaces.

2. Use Installation Manager to install the following on Machine B.
   - WebSphere Application Server Network Deployment
   - Web Server Plug-ins for WebSphere Application Server
   - Websphere Customization Toolbox

3. Use Installation Manager to install IBM HTTP Server on Machine B, or install another supported web server on Machine B.

4. Create a custom profile as the first profile Machine B, and federate the node as you create it.

5. Optional: Use the administrative console of the deployment manager to create an application server on the custom node.

   Click **Servers > Applications servers > New** and follow the instructions to create a server. A server is not required for installing the plug-ins but it lets you verify the functionality of the web server.

6. Optional: Install the DefaultApplication on the new server while you are in the administrative console of the deployment manager.

   The DefaultApplication includes the Snoop servlet. The verification step uses the Snoop servlet.

7. Open the WebSphere Customization Toolbox and launch the Web Server Plug-ins Configuration Tool on Machine B.

8. Select a web server plug-in runtime location.

   If the location of a previously installed web server plug-in that you want to use is not in the list, perform the following actions to add the location to your working set:

   a. Click **Add**.

   b. Enter a name for the web server plug-in location.

   c. Perform one of the following actions:
      - Enter the location.
      - Click **Browse**, find the location, and click **OK**.

9. Click **Create**.

10. Select the type of web server that you are configuring, and click **Next**.

11. Select the architecture of your installed target web server (64 bit or 32 bit) and click **Next** if you are asked.

12. Click **Browse** to select the configuration file or files for your web server, verify that the web server port is correct, and then click **Next** when you are finished.

Select the file and not just the directory of the file. Some web servers have two configuration files and require you to browse for each file.

The following list shows configuration files for supported web servers:

Apache HTTP Server
> *apache_root*/config/httpd.conf

Domino Web Server
> names.nsf and Notes.jar
>
> The wizard prompts for the notes.jar file. The actual name is Notes.jar.
>
> The Web Server Plug-ins Configuration Tool verifies that the files exist but the tool does not validate either file.

IBM HTTP Server
> *IHS_root*/conf/httpd.conf

Microsoft Internet Information Services (IIS)
> The Web Server Plug-ins Configuration Tool can determine the correct files to edit.

Sun Java System Web Server (formerly Sun ONE Web Server and iPlanet Web Server) Version 6.0 and later
> obj.conf and magnus.conf

13. If you are configuring an IBM HTTP web server plug-in, perform the following actions.

    a. Optionally, set up the administration server configuration to administer the web server.

        1) Select **Setup IBM HTTP Server Administration Server**.

        2) Specify a port number on which the IBM HTTP administration server will communicate.

        3) Optionally, select **Create a user ID for IBM Server Administration Server authentication** and enter a user ID and password to authenticate to the IBM HTTP Server administrative server from the administrative console.

    b. Click **Next**.

    c. `AIX` `HP-UX` `Linux` `Solaris` Specify the system user ID and group to have write permission to IBM HTTP Server, the IBM HTTP Server administrative server, and the web server plug-in configuration files.

    Select **Create a new unique system user ID and group using the credentials** if necessary.

    d. `Windows` Optionally, set up the IBM HTTP Server Administration Server to run as a Window service.

        1) Select **Run IBM HTTP Server Administration Server as a Windows Service**.

        2) Perform one of the following actions:

            • Select **Log on as a local system account**.

            • Select **Log on as a specified user account**, and enter the user ID and password for that account.

            The user ID requires the following advanced user rights:

            – Act as part of the operating system

            – Log on as a service

        3) Choose whether your startup type will be automatic or manual.

    e. Click **Next**.

14. Specify a unique name for the web server definition, and click **Next**.

15. Select the configuration scenario.

    a. Choose the local scenario.

b. Perform one of the following actions:
- Enter the installation location of WebSphere Application Server (*app_server_root*).
- Click **Browse**, find the installation location of WebSphere Application Server (*app_server_root*), and click **OK**.

c. Click **Next**.

16. Select the profile to configure with the current web server plug-in, and click **Next**.

17. Review the summary information, and click **Configure** to begin configuring the web server, web server plug-in, and profile.

18. Verify the success of the installation on the summary panel, and click **Finish**.

If a problem occurs and the installation is unsuccessful, examine the logs in the *plugins_root*/logs directory. Correct any problems and re-configure.

19. Create the web server definition on Machine A.

You can use the administrative console of the deployment manager to create the web server definition on a federated node; or you can run the configuration script that the Web Server Plug-ins Configuration Tool created.

The script already contains all of the information that you must gather when using the administrative console option.

- **Using the administrative console**

  Click **Servers > Web servers > New** and use the Create new web server entry wizard to create the web server definition.

- **Running the configuration script**

  a. Paste the `configureweb_server_name` script from Machine B to the `app_server_root`/bin directory on Machine A.

  b. Issue the appropriate command from a command window:
  - **AIX** **HP-UX** **Linux** **Solaris** `./`*plugins_root*`/bin/`
    `configureweb_server_name`.sh
  - **Windows** `plugins_root\bin\configureweb_server_name`.bat

  If you have enabled security or changed the default JMX connector type, edit the script and include the appropriate parameters on the wsadmin command.

20. **Domino Web Server only:** Set the WAS_PLUGIN_CONFIG_FILE environment variable.

On platforms such as AIX or Linux, sourcing a script to the parent shell allows child processes to inherit the exported variables. On Windows systems, run the script as you would run any other command. Sourcing is automatic on Windows systems.

a. Open a command window.

b. Change directories to the plug-ins installation root directory.

c. Issue the appropriate command for the *plugins_root*/bin/setupPluginCfg.sh script:
- **AIX** **HP-UX** **Solaris** `.` *plugins_root*`/bin/setupPluginCfg.sh` (Notice the space between the period and the installation root directory.)
- **Linux** `source` *plugins_root*`/bin/setupPluginCfg.sh`

The script is also in the *lotus_root*/notesdata directory on operating systems such as AIX or Linux.

Issue the appropriate command for the script before starting the Domino Web Server.

21. Start the Snoop servlet to verify the ability of the web server to retrieve an application from the application server.

Test your environment by starting your application server, your web server, and using the Snoop servlet with an IP address.

a. Start the application server.

Change directories to the *profile_root*/bin directory and run the startServer command:
- **AIX** **HP-UX** **Linux** **Solaris** `./startServer.sh server1`

- **Windows** `startServer server1`

b. Start the IBM HTTP Server or the web server that you are using.

Use a command window to change the directory to the IBM HTTP Server installed image, or to the installed image of your web server. Issue the appropriate command to start the web server, such as these commands for IBM HTTP Server:

To start the IBM HTTP Server from the command line:

Access the apache and apachectl commands in the `IBMHttpServer/bin` directory.

- **AIX** **HP-UX** **Linux** **Solaris** `./apachectl start`
- **Windows** `apache`

c. Point your browser to `http://localhost:9080/snoop` to test the internal HTTP transport provided by the application server. Point your browser to `http://Host_name_of_Web_server_machine/snoop` to test the web server plug-in.

The HTTP Transport port is 9080 by default and must be unique for every profile. The port is associated with a virtual host named default_host, which is configured to host the installed DefaultApplication. The Snoop servlet is part of the DefaultApplication. Change the port to match your actual HTTP Transport port.

d. Verify that Snoop is running.

Either Web address should display the Snoop Servlet - Request/Client Information page.

e. **Remote IBM HTTP Server only:**

Verify that the automatic propagation function can work on a remote IBM HTTP Server by using the following steps. This procedure is not necessary for local web servers.

1) Create a user=adminUser, password=adminPassword in the *IHS_root* `/conf/admin.passwd` file. For example: `c:\ws\ihs80\bin\htpasswd -cb c:\ws\ihs80\conf\admin.passwd adminUser adminPassword`

2) Use the administrative console of the application server to enter the User ID and password information that you created for the administrative user of IBM HTTP Server. Go to **Servers > Web server > *web_server_definition* > Remote Web server administration**. Set the following values: admin Port=8008, User Id=adminUser, Password=adminPassword.

3) Set the correct read/write permissions for the `httpd.conf` file and the plugin-cfg.xml file. See the *IHS_root* `/logs/admin_ERROR.LOG` file for more information.

Automatic propagation of the plug-in configuration file requires the IBM HTTP administrative server to be up and running. If you are managing an IBM HTTP Server using the WebSphere Application Server administrative console, the following error might display:

`"Could not connect to IHS Administration server error"`

Perform the following procedure to correct the error:

1) Verify that the IBM HTTP Server administration server is running.

2) Verify that the web server host name and the port that is defined in the WebSphere Application Server administrative console matches the IBM HTTP Server administration host name and port.

3) Verify that the fire wall is not preventing you from accessing the IBM HTTP Server administration server from the WebSphere Application Server administrative console.

4) Verify that the user ID and password that is specified in the WebSphere Application Server administrative console under remote managed, is created in the `admin.passwd` file, using the htpasswd command.

5) If you are trying to connect securely, verify that you export the IBM HTTP Server administration server keydb personal certificate into the WebSphere Application Server key database as a signer certificate. This key database is specified by the com.ibm.ssl.trustStore directive in the `sas.client.props` file in the profile where your administrative console is running. This consideration is primarily for self-signed certificates.

6) If you still have problems, check the IBM HTTP Server `admin_ERROR. LOG` file and the WebSphere Application Server logs (`trace.log` file) to determine the cause of the problem.

22. If the deployment manager does not have the DefaultApplication installed, you can test the functionality of the web server and the custom node using an application of your own.

23. From the administrative console of the deployment manager, click **System administration > Save Changes to Master Repository > Synchronize changes with Nodes > Save**.

24. To create multiple web server definitions for the managed node, use the Web Server Plug-ins Configuration Tool to configure each web server.

    Identify the same managed node each time. Give each web server a different nick name.

Results

This procedure results in the installation of the Web Server Plug-ins for WebSphere Application Server on a web server machine. The Web Server Plug-ins Configuration Tool creates a web server definition within the managed node.

The Web Server Plug-ins Configuration Tool configures the web server to use the `plugin-cfg.xml` file that is within the managed custom node.

The deployment manager regenerates the web server plug-in configuration file, `plugin-cfg.xml` whenever an event occurs that affects the file. Such events include the addition or removal of an application, server, or virtual host.

The creation or removal of clusters and cluster members also causes file regeneration. Automatic propagation through node synchronization copies the file after each regeneration to the following location on the custom node machine:

```
profile_root
    /config/cells/cell_name/nodes/
    node_name_of_custom_profile/servers/
    web_server_name/plugin-cfg.xml
```

The installation of the Web Server Plug-ins results in the creation of the `Plugins` directory and several subdirectories. The following directories are among those created on a Linux system, for example:

- *plugins_root*/`bin` contains the binary plug-ins for all supported web servers
- *plugins_root*/`logs` contains log files
- *plugins_root*/`properties` contains version information

What to do next

See "Plug-ins configuration" on page 78 for information about the location of the plug-in configuration file.

See "Web server configuration" on page 34 for more information about the files involved in configuring a web server.

See "Editing web server configuration files" on page 10 for information about how the Web Server Plug-ins Configuration Tool configures supported web servers.

See "Installing and configuring web server plug-ins" on page 50 for information about other installation scenarios for installing Web server plug-ins.

Configuring a web server plug-in using the WCT command-line utility

The WCT command-line utility invokes the command-line tool that is specified by the -tool parameter. You can use the WCT command-line utility and specify the pct tool to configure a web server to use an application server as a hosting server.

Procedure

Configure a web server to use an application server as a hosting server.

Location of the WCT command-line utility

The product includes the following script that sets up the environment and invokes the WCT command-line utility.

- `Windows` *WCT_install_root*\WCT\wctcmd.bat
- `Linux` *WCT_install_root*/WCT/wctcmd.sh

Syntax of the WCT command-line utility

`Windows`

```
wctcmd.bat
     -tool tool_ID
     -defLocPathname definition_location_pathname
     -createDefinitionLocation definition_location_name
     -importDefinitionLocation definition_location_name
     -removeDefinitionLocation definition_location_name
     -defLocName definition_location_name
     -defLocVersion definition_location_version
     -response response_file
     -listDefinitionLocations
     -deleteDefinition definition_name
     -listDefinitions
```

`Linux`

```
./wctcmd.sh
     -tool tool_ID
     -defLocPathname definition_location_pathname
     -createDefinitionLocation definition_location_name
     -importDefinitionLocation definition_location_name
     -removeDefinitionLocation definition_location_name
     -defLocName definition_location_name
     -defLocVersion definition_location_version
     -response response_file
     -listDefinitionLocations
     -deleteDefinition definition_name
     -listDefinitions
```

Parameters of the WCT command-line utility

-tool *tool_ID*

Specifies the name of the tool to launch as it is registered with the WCT command-line utility

This parameter is required.

-defLocPathname *definition_location_pathname*

Specifies the absolute path name of the definition location to use when the specified tool is launched

This parameter is required.

-createDefinitionLocation *definition_location_name*

Specifies that the WCT command-line utility should create a definition location

This parameter is optional.

-importDefinitionLocation *definition_location_name*

Specifies that the WCT command-line utility should import a definition location

This parameter is optional.

-removeDefinitionLocation *definition_location_name*

Specifies that the WCT command-line utility should remove a definition location

This parameter is optional.

-defLocName *definition_location_name*

Specifies the name of the definition location as it resides in the definition location registry

-defLocVersion *definition_location_version*
> Specifies the version of definition location to create

> This parameter is optional.

-response *response_file*
> Specifies the response file containing tool arguments

> This parameter is optional.

-listDefinitionLocations
> Lists the available definition locations.

-deleteDefinition *definition_name*
> Specifies that the WCT command-line utility should delete a definition

> This parameter is optional.

> The *definition_name* is required. Either one of the following parameters is also required:
> * `-defLocName` *definition_location_name*
> * `-defLocpathname` *definition_location_pathname*

> If both parameter values are supplied, the first one is used. If the first value supplied does not pass the validation check, the command fails with an error message.

-listDefinitions
> Lists the available definitions at a specified definition location or definition location path name

> Either one of the following parameters is required:
> * `-defLocName` *definition_location_name*
> * `-defLocpathname` *definition_location_pathname*

> If both parameter values are supplied, the first one is used. If the first value supplied does not pass the validation check, the command fails with an error message.

Notes:
> * Command-line arguments are case sensitive.
> * If an argument accepts a value containing spaces, the value must be enclosed in double quotes (" ").

Examples

Using the pct tool to configure an IHS Web Server to use an application server as a hosting server:

* **Windows** `wctcmd.bat -tool pct -defLocPathname C:\data\IBM\WebSphere\Plugins -defLocName someDefLocName -response C:\IBM\WebSphere\tools\WCT\ responsefile.txt`
* **Linux** `./wctcmd.sh -tool pct -defLocPathname /data/IBM/WebSphere/Plugins -defLocName someDefLocName -response /var/IBM/WebSphere/tools/WCT/ responsefile.txt`

The following is an example of the content of a response file for an IHS local plug-in configuration.

 AIX   **HP-UX**   **Linux**   **Solaris**

```
configType=local_standalone
enableAdminServerSupport=true
enableUserAndPass=true
enableWinService=false
ihsAdminCreateUserAndGroup=true
ihsAdminPassword=******
```

```
                        ihsAdminPort=8008
                        ihsAdminUnixUserGroup=grp101
                        ihsAdminUnixUserID=user1
                        mapWebServerToApplications=true
                        profileName=AppSrv01
                        wasExistingLocation=opt/IBM/WebSphere/AppServer80
                        webServerConfigFile1=opt/IBM/HTTPServer/conf/httpd.conf
                        webServerDefinition=webserver1
                        webServerHostName=local.ibm.com
                        webServerInstallArch=32
                        webServerOS=windows
                        webServerPortNumber=80
                        webServerSelected=ihs
```

Windows

```
configType=local_standalone
enableAdminServerSupport=true
enableUserAndPass=true
enableWinService=true
ihsAdminPassword=******
ihsAdminPort=8008
ihsAdminUserID=admin1
ihsWindowsPassword=******
ihsWindowsStartupType=Automatic
ihsWindowsUserID=user1
mapWebServerToApplications=true
profileName=AppSrv01
wasExistingLocation=D:\IBM\WebSphere\AppServer80
webServerConfigFile1=D:\IBM\HTTPServer\conf\httpd.conf
webServerDefinition=webserver1
webServerHostName=local.ibm.com
webServerInstallArch=32
webServerOS=windows
webServerPortNumber=80
webServerSelected=ihs
```

The following is an example of the content of a response file for an IHS remote plug-in configuration.

AIX   **HP-UX**   **Linux**   **Solaris**

```
configType=remote
enableAdminServerSupport=true
enableUserAndPass=true
enableWinService=false
ihsAdminCreateUserAndGroup=true
ihsAdminPassword=******
ihsAdminPort=8008
ihsAdminUnixUserGroup=grp101
ihsAdminUnixUserID=user1
mapWebServerToApplications=true
profileName=AppSrv01
wasExistingLocation=opt/IBM/WebSphere/AppServer80
wasMachineHostname=192.168.1.2
webServerConfigFile1=opt/IBM/HTTPServer/conf/httpd.conf
webServerDefinition=webserver1
webServerHostName=remote.ibm.com
webServerInstallArch=32
webServerOS=windows
webServerPortNumber=80
webServerSelected=ihs
```

Windows

```
configType=remote
enableAdminServerSupport=true
enableUserAndPass=true
enableWinService=true
ihsAdminPassword=******
ihsAdminPort=8008
ihsAdminUserID=admin1
ihsWindowsPassword=******
ihsWindowsStartupType=Automatic
ihsWindowsUserID=user1
mapWebServerToApplications=true
profileName=AppSrv01
wasExistingLocation=D:\IBM\WebSphere\AppServer80
wasMachineHostname=192.168.1.2
webServerConfigFile1=D:\IBM\HTTPServer\conf\httpd.conf
webServerDefinition=webserver1
webServerHostName=remote.ibm.com
```

```
webServerInstallArch=32
webServerOS=windows
webServerPortNumber=80
webServerSelected=ihs
```

Importing a definition location for the pct tool:

`Windows`

wctcmd.bat -tool pct -importDefinitionLocation -defLocName someDefLocName -defLocPathname C:\data\IBM\WebSphere\Plugins

`Linux`

./wctcmd.sh -tool pct -importDefinitionLocation -defLocName someDefLocName -defLocPathname /data/IBM/WebSphere/Plugins

Removing a definition location for the pct tool:

`Windows`

wctcmd.bat -tool pct -removeDefinitionLocation -defLocName someDefLocName -defLocPathname C:\data\IBM\WebSphere\Plugins

`Linux`

./wctcmd.sh -tool pct -removeDefinitionLocation -defLocName someDefLocName -defLocPathname /data/IBM/WebSphere/Plugins

Listing the available definition locations for the pct tool:

`Windows`

wctcmd.bat -tool pct -defLocPathname C:\data\IBM\WebSphere\Plugins -listDefinitionLocations

`Linux`

./wctcmd.sh -tool pct -defLocPathname /data/IBM/WebSphere/Plugins -listDefinitionLocations

Removing a definition for the pct tool:

`Windows`

wctcmd.bat -tool pct -deleteDefinition someDefName -defLocName someDefLocName

`Linux`

./wctcmd.sh -tool pct -deleteDefinition someDefName -defLocName someDefLocName

Listing the available definitions for the pct tool:

`Windows`

wctcmd.bat -tool pct -defLocPathname C:\data\IBM\WebSphere\Plugins -listDefinitions

`Linux`

./wctcmd.sh -tool pct -defLocPathname /data/IBM/WebSphere/Plugins -listDefinition

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 (distributed)

The following file paths are default locations. You can install the product and other components or create profiles in any directory where you have write access. Multiple installations of WebSphere Application Server products or components require multiple locations. Default values for installation actions by root and nonroot users are given. If no nonroot values are specified, then the default directory values are applicable to both root and nonroot users.

app_client_root

Table 9. Default installation root directories for the Application Client for IBM WebSphere Application Server.

This table shows the default installation root directories for the Application Client for IBM WebSphere Application Server.

User	Directory
Root	**AIX** `/usr/IBM/WebSphere/AppClient` (Java EE Application client only) **HP-UX** **Linux** **Solaris** `/opt/IBM/WebSphere/AppClient` (Java EE Application client only) **Windows** `C:\Program Files\IBM\WebSphere\AppClient`
Nonroot	**AIX** **HP-UX** **Linux** **Solaris** `user_home/IBM/WebSphere/AppClient` (Java EE Application client only) **Windows** `C:\IBM\WebSphere\AppClient`

app_server_root

Table 10. Default installation directories for WebSphere Application Server.

This table shows the default installation directories for WebSphere Application Server.

User	Directory
Root	**AIX** `/usr/IBM/WebSphere/AppServer` **HP-UX** **Linux** **Solaris** `/opt/IBM/WebSphere/AppServer` **Windows** `C:\Program Files\IBM\WebSphere\AppServer`
Nonroot	**AIX** **HP-UX** **Linux** **Solaris** `user_home/IBM/WebSphere/AppServer` **Windows** `user_home\IBM\WebSphere\AppServer`

component_root

The component installation root directory is any installation root directory described in this topic. Some programs are for use across multiple components—in particular, the Web Server Plug-ins, the Application Client, and the IBM HTTP Server. All of these components are part of the product package.

gskit_root

IBM Global Security Kit (GSKit) can now be installed by any user. GSKit is installed locally inside the installing product's directory structure and is no longer installed in a global location on the target system. The following list shows the default installation root directory for Version 8 of the GSKit, where *product_root* is the root directory of the product that is installing GSKit, for example IBM HTTP Server or the web server plug-in.

AIX **HP-UX** **Linux** **Solaris**

`product_root/gsk8`

Windows

`product_root\gsk8`

profile_root

Table 11. Default profile directories.

This table shows the default directories for a profile named profile_name on each distributed operating system.

User	Directory
Root	**AIX** `/usr/IBM/WebSphere/AppServer/profiles/profile_name` **HP-UX** **Linux** **Solaris** `/opt/IBM/WebSphere/AppServer/profiles/profile_name` **Windows** `C:\Program Files\IBM\WebSphere\AppServer\profiles\profile_name`
Nonroot	**AIX** **HP-UX** **Linux** **Solaris** `user_home/IBM/WebSphere/AppServer/profiles` **Windows** `user_home\IBM\WebSphere\AppServer\profiles`

plugins_root

Table 12. Default installation root directories for the Web Server Plug-ins.

This table shows the default installation root directories for the Web Server Plug-ins for WebSphere Application Server.

User	Directory
Root	**AIX** `/usr/IBM/WebSphere/Plugins` **HP-UX** **Linux** **Solaris** `/opt/IBM/WebSphere/Plugins` **Windows** `C:\Program Files\IBM\WebSphere\Plugins`
Nonroot	**AIX** **HP-UX** **Linux** **Solaris** `user_home/IBM/WebSphere/Plugins` **Windows** `C:\IBM\WebSphere\Plugins`

wct_root

Table 13. Default installation root directories for the WebSphere Customization Toolbox.

This table shows the default installation root directories for the WebSphere Customization Toolbox.

User	Directory
Root	**AIX** `/usr/IBM/WebSphere/Toolbox` **HP-UX** **Linux** **Solaris** `/opt/IBM/WebSphere/Toolbox` **Windows** `C:\Program Files\IBM\WebSphere\Toolbox`
Nonroot	**AIX** **HP-UX** **Linux** **Solaris** *user_home*`/IBM/WebSphere/Toolbox` **Windows** `C:\IBM\WebSphere\Toolbox`

web_server_root

Table 14. Default installation root directories for the IBM HTTP Server.

This table shows the default installation root directories for the IBM HTTP Server.

User	Directory
Root	**AIX** `/usr/IBM/HTTPServer` **HP-UX** **Linux** **Solaris** `/opt/IBM/HTTPServer` **Windows** `C:\Program Files\IBM\HTTPServer`
Nonroot	**AIX** **HP-UX** **Linux** **Solaris** *user_home*`/IBM/HTTPServer` **Windows** `C:\IBM\HTTPServer`

Configuring simple load balancing across multiple application server profiles

Simple load balancing distributes HTTP requests across multiple IBM WebSphere Application Server instances. You can configure simple load balancing to provide failover of an application state that is maintained in an HTTP session.

Before you begin

Note: This offering applies to stand-alone application server profiles for IBM WebSphere Application Server. This offering does not include a centralized management capability such as the deployment manager in WebSphere Application Server, Network Deployment.

About this task

You can configure simple load balancing capability with WebSphere Application Server by combining the plug-in configuration files of multiple stand-alone application server profiles into a single configuration file. The number of configuration files that you can combine are bound by the limits that exist in the WebSphere Application Server license agreement. You can use the following different configurations of the application server to combine the plug-in configuration files of multiple application server profiles into a single output file:

• Using multiple stand-alone base application server profiles

- Using multiple stand-alone base application server profiles with an administrative agent. For more information, see the documentation on configuring simple load balancing across multiple stand-alone base application server profiles with an administrative agent.
- Using multiple stand-alone base application server profiles with an administrative agent using the job manager. The job manager function is a part of WebSphere Application Server, Network Deployment. However, you can use the job manager function with stand-alone, base application server profiles. For more information, see the documentation on configuring simple load balancing across multiple stand-alone, base application server profiles with an administrative agent using the job manager.

Complete the following steps to configure simple load balancing across multiple stand-alone, base application server profiles:

Procedure

1. Install WebSphere Application Server and create application server profiles. For more information, see the documentation on WebSphere Application Server installation and application server profiles.
2. Install the enterprise application or web module. For more information, see the documentation on the methods for installing applications or modules.
3. Determine if you require session affinity.

   Session affinity directs requests from a given client to a specific application server. The application state maintained in the HTTP session is accessed in the HTTP session cache, which is local to the application server. Session affinity provides higher performance than database persistence of the session object, alone. Without session affinity, session requests must be obtained from the database if they are sent to a server that does not have the session object in the local cache.
4. Optional: Configure a unique HTTP session clone ID for *each* application server. You must complete this step if you require session affinity.

   You can configure a unique HTTP session clone ID using wsadmin scripting or the administrative console. To use wsadmin commands for the Jython or Jacl programming language, see the documentation on configuring a unique HTTP session clone ID for each application server using scripting. To configure a unique HTTP session clone ID using the administrative console, complete the following steps:

   a. Expand **Servers** > **Server Types** and click **WebSphere application servers** > *server_name*.
   b. Under **Container Settings**, expand **Web Container Settings**, and click **Web container**.
   c. Under **Additional Properties**, click **Custom properties** > **New**.
   d. In the **Name** field, enter `HttpSessionCloneId`.
   e. In the **Value** field, enter a unique value for the server. The unique value must be 8 - 9 alphanumeric characters; for example, test1234
   f. Click **Apply** or **OK**.
   g. Click **Save** to save the configuration changes to the master configuration.
5. Optional: Configure session persistence, if needed. If you require session failover capability, you must configure session persistence. Persistence of the session object to a database is the only option for session failover with WebSphere Application Server. To configure session persistence using the administrative console, see the documentation on configuring database session persistence. To configure database session persistence using wsadmin commands for the Jython or Jacl programming language, see the documentation on configuring database session persistence using scripting.
6. Restart the server.
7. Generate the `plugin-cfg.xml` file for each application server using the GenPluginCfg script, the administrative console, or wsadmin scripting.

   See the topic *GenPluginCfg command* for a description of how to use the GenPluginCfg script to generate the `plugin-cfg.xml` file.

   See the topic *Implementing a web server plug-in* for a description of how to use the administrative console to generate the `plugin-cfg.xml` file.

See the topic *Regenerating the node plug-in configuration using scripting* for a description of how to use wsadmin scripting to generate the `plugin-cfg.xml` file.

8. Merge the `plugin-cfg.xml` files from multiple application server nodes.

   You can either manually merge the `plugin-cfg.xml` files or use the pluginCfgMerge tool to automatically merge the `plugin-cfg.xml` file from multiple application server profiles into a single output file. The `pluginCfgMerge.bat` and `pluginCfgMerge.sh` files are located in the *install_root*/bin directory.

   To use the pluginCfgMerge tool, complete the following steps:

   a. Rename the `plugin-cfg.xml` files to a unique name across your application server profiles.

   b. Copy the `plugin-cfg.xml` file for all stand-alone application server profiles into a common directory.

   c. Use the pluginCfgMerge tool to combine the `plugin-cfg.xml` files from each of the application server profiles into a single output file. For example: **AIX** **HP-UX** **Linux** **Solaris**

      install_root/bin/pluginCfgMerge.sh *plugin_configuration_file1 plugin_configuration_file2 resulting_plugin_config*

      Windows

      install_root\bin\pluginCfgMerge.bat *plugin_configuration_file1 plugin_configuration_file2 resulting_plugin_confi*

      The *resulting_plugin_configuration_file* variable value is normally `plugin-cfg.xml`

   For more information about manually merging the `plugin-cfg.xml` files, see the technote on merging `plugin-cfg.xml` files from multiple application server profiles.

9. Copy the merged `plugin-cfg.xml` file to the *plugin_installation_root*/config/web_server_name/ directory on the web server host.

10. **AIX** **HP-UX** **Linux** **Solaris** Ensure that you have defined the correct operating system file access permissions for the merged `plugin-cfg.xml` file. These file access permissions allow the HTTP server plug-in process to read the file.

Results

When you complete this process, you have one plug-in configuration file for multiple stand-alone application server profiles.

Configuring simple load balancing across multiple application server profiles with an administrative agent

Simple load balancing distributes HTTP requests across multiple IBM WebSphere Application Server instances. Also, you can configure simple load balancing to provide failover of an application state that is maintained in an HTTP session.

Before you begin

Note: This offering applies to stand-alone application server profiles for IBM WebSphere Application Server. This offering does not include a centralized management capability such as the deployment manager in WebSphere Application Server, Network Deployment.

About this task

You can configure simple load balancing capability with WebSphere Application Server by combining the plug-in configuration files of multiple stand-alone application server profiles into a single configuration file. The number of configuration files that you can combine are bound by the limits that exist in the WebSphere Application Server license agreement. You can use the following different configurations of the application server to combine the plug-in configuration files of multiple application server profiles into a single output file:

- Using multiple stand-alone base application server profiles. For more information, see the documentation on configuring simple load balancing across multiple application server profiles.
- Using multiple stand-alone base application server profiles with an administrative agent. This topic explains how to complete this configuration.
- Using multiple stand-alone base application server profiles with an administrative agent using the job manager. The job manager function is a part of WebSphere Application Server, Network Deployment. However, you can use the job manager function with stand-alone, base application server profiles. For more information, see the documentation on configuring simple load balancing across multiple stand-alone, base application server profiles with an administrative agent using the job manager.

Complete the following steps to register stand-alone application server profiles with an administrative agent and combine the plug-in configuration files from these profiles into a single output file.

Procedure

1. Install WebSphere Application Server and create application server profiles. For more information, see the documentation on WebSphere Application Server installation and application server profiles.
2. Configure the administrative agent and register each application server profile with the administrative agent. Complete the following steps:
   a. Set up the administrative agent, which includes creating the administrative agent profile.
   b. Register the stand-alone application server with the administrative agent.
   c. Start and stop the administrative agent.

   After you complete these steps, you can complete all the administrative operations through the administrative agent. When you log in to the administrative console for the administrative agent, you can select which application servers to manage. For more information, see the documentation on administering stand-alone nodes using the administrative agent.
3. Install the enterprise application or web module. For more information, see the documentation on installing enterprise applications or modules.
4. Determine if you require session affinity.

   Session affinity directs requests from a given client to a specific application server. The application state maintained in the HTTP session is accessed in the HTTP session cache, which is local to the application server. Session affinity provides higher performance than database persistence of the session object, alone. Without session affinity, session requests must be obtained from the database if they are sent to a server that does not have the session object in the local cache.
5. Optional: Configure a unique HTTP session clone ID for *each* application server. You must complete this step if you require session affinity.

   You can configure a unique HTTP session clone ID using wsadmin scripting or the administrative console. To use wsadmin commands for the Jython or Jacl programming language, see the documentation on configuring a unique HTTP session clone ID for each application server using scripting. To configure a unique HTTP session clone ID using the administrative console, complete the following steps:
   a. Expand **Servers** > **Server Types** and click **WebSphere application servers** > *server_name*.
   b. Under **Container Settings**, expand **Web Container Settings**, and click **Web container**.
   c. Under **Additional Properties**, click **Custom properties** > **New**.
   d. In the **Name** field, enter `HttpSessionCloneId`.
   e. In the **Value** field, enter a unique value for the server. The unique value must be 8 - 9 alphanumeric characters; for example, test1234
   f. Click **Apply** or **OK**.
   g. Click **Save** to save the configuration changes to the master configuration.
6. Optional: Configure session persistence. If you require session failover capability, you must configure session persistence. Persistence of the session object to a database is the only option for session failover with WebSphere Application Server. To configure session persistence using the administrative

console, see the documentation on configuring database session persistence. To configure database session persistence using wsadmin commands for the Jython or Jacl programming language, see the documentation on configuring database session persistence using scripting.

7. Restart the server.
8. Generate the `plugin-cfg.xml` file for each stand-alone application server using the GenPluginCfg script, the administrative console, or wsadmin scripting.

   To use the GenPluginCfg script, enter the following command on the command line:*profile_root*/`config/cells/GenPluginCfg.sh|bat`

   To use the administrative console, see the documentation on creating or updating a global web server plug-in configuration file.

   The following variables apply to the Jython and Jacl commands:
   - *cell_name* is the name of your cell.
   - *web_server_node* is the name of the node for your web server.
   - *web_server_name* is the name of your web server.

   Jython
   > On the command line, enter each of the following commands on a separate line:
   >
   > ```
   > generator = AdminControl.completeObjectName('type=PluginCfgGenerator,*')
   > AdminControl.invoke(generator, 'generate', "profile_root/config cell_name web_server_node web_ser
   > ```

   Jacl On the command line, enter each of the following commands on a separate line:

   > ```
   > set generator [$AdminControl completeObjectName type=PluginCfgGenerator,*]
   > $AdminControl invoke $generator generate "profile_root/config cell_name web_server_node web_serve
   > ```

9. Merge the `plugin-cfg.xml` files from multiple application server nodes.

   You can either manually merge the `plugin-cfg.xml` files or use the pluginCfgMerge tool to automatically merge the `plugin-cfg.xml` file from multiple application server profiles into a single output file. The `pluginCfgMerge.bat` or `pluginCfgMerge.sh` tool is available after you install this fix pack and is located in the *install_root*/`bin` directory. To use the pluginCfgMerge tool, complete the following steps:

   a. Rename the `plugin-cfg.xml` files to a unique name across your application server profiles.
   b. Copy the `plugin-cfg.xml` file for all stand-alone application server profiles into a common directory.
   c. Use the pluginCfgMerge tool to combine the `plugin-cfg.xml` files from each of the application server profiles into a single output file. For example: `AIX` `HP-UX` `Linux` `Solaris`

      install_root/`bin/pluginCfgMerge.sh` *plugin_configuration_file1 plugin_configuration_file2 resulting_plugin_config*

      `Windows`

      install_root\`bin\pluginCfgMerge.bat` *plugin_configuration_file1 plugin_configuration_file2 resulting_plugin_confi*

      The *resulting_plugin_configuration_file* variable value is normally `plugin-cfg.xml`

   For more information about manually merging the `plugin-cfg.xml` files, see the technote on merging `plugin-cfg.xml` files from multiple application server profiles.

10. Copy the merged `plugin-cfg.xml` file to the *plugin_installation_root*/`config`/*web_server_name*/ directory on the Web server host.
11. `AIX` `HP-UX` `Linux` `Solaris` Ensure that you have defined the correct operating system file access permissions for the merged `plugin-cfg.xml` file. These file access permissions allow the HTTP server plug-in process to read the file.

Results

When you complete this process, you have one plug-in configuration file for multiple stand-alone application server profiles.

Configuring simple load balancing across multiple application server profiles with an administrative agent using a job manager

Simple load balancing distributes HTTP requests across multiple IBM WebSphere Application Server instances. You can configure simple load balancing to provide failover of an application state that is maintained in an HTTP session.

About this task

You can configure simple load balancing capability with WebSphere Application Server by combining the plug-in configuration files of multiple stand-alone application server profiles into a single configuration file. The number of configuration files that you can combine are bound by the limits that exist in the WebSphere Application Server license agreement. You can use the following different configurations of the application server to combine the plug-in configuration files of multiple application server profiles into a single output file:

- Using multiple stand-alone base application server profiles. For more information, see the documentation on configuring simple load balancing across multiple application server profiles.
- Using multiple stand-alone base application server profiles with an administrative agent. For more information, see the documentation on configuring simple load balancing across multiple application server profiles with an administrative agent
- Using multiple stand-alone base application server profiles with an administrative agent using the job manager. The job manager function is a part of WebSphere Application Server, Network Deployment. However, you can use the job manager function with stand-alone, base application server profiles. Use this topic to complete this configuration.

Complete the following steps to register stand-alone application server profiles with an administrative agent using a job manager and combine the plug-in configuration files from these profiles into a single output file.

Procedure

1. Install WebSphere Application Server and create application server profiles. For more information, see the documentation on WebSphere Application Server installation and application server profiles.
2. Configure the administrative agent and register each application server profile with the administrative agent. Complete the following steps:
   a. Set up the administrative agent, which includes creating the administrative agent profile.
   b. Register the stand-alone application server with the administrative agent.
   c. Start and stop the administrative agent.

   After you complete these steps, you can complete all the administrative operations through the administrative agent. When you log in to the administrative console for the administrative agent, you can select which application servers to manage. For more information, see the documentation on administering stand-alone nodes using the administrative agent.
3. Install WebSphere Application Server, Network Deployment for the licensed WebSphere Application Server, Network Deployment instances that will perform centralized management for the stand-alone application server instances. For more information, see the installation documentation for the WebSphere Application Server, Network Deployment product.

   Attention: You must access the Information Center for WebSphere Application Server, Network Deployment to read its installation documentation.
4. Create the job manager profile, configure the job manager, and register stand-alone application servers with the job manager. For more information, see the documentation on setting up a job

manager environment. You can complete administrative options centrally using the job manager. For more information about the job manager, see the conceptual information about the job manager.

5. Install the enterprise application or web module. You can use one of the following methods to install the enterprise application or web module:

   - Install the enterprise application or web module on each application server. For more information, see the documentation on installing enterprise applications or modules.

   - Install the enterprise application or web module using the job manager. For more information, see the documentation on installing applications using the job manager.

6. Determine if you require session affinity.

   Session affinity directs requests from a given client to a specific application server. The application state maintained in the HTTP session is accessed in the HTTP session cache, which is local to the application server. Session affinity provides higher performance than database persistence of the session object, alone. Without session affinity, session requests must be obtained from the database if they are sent to a server that does not have the session object in the local cache.

7. Optional: Configure a unique HTTP session clone ID for *each* application server. You must complete this step if you require session affinity.

   You can configure a unique HTTP session clone ID using wsadmin scripting or the administrative console. To use wsadmin commands for the Jython or Jacl programming language, see the documentation on configuring a unique HTTP session clone ID for each application server using scripting. To configure a unique HTTP session clone ID using the administrative console, complete the following steps:

   a. Expand **Servers** > **Server Types** and click **WebSphere application servers** > *server_name*.
   b. Under **Container Settings**, expand **Web Container Settings** and click **Web container**.
   c. Under **Additional Properties**, click **Custom properties** > **New**.
   d. In the **Name** field, enter `HttpSessionCloneId`.
   e. In the **Value** field, enter a unique value for each server. The unique value must be 8 - 9 alphanumeric characters; for example, test1234
   f. Click **Apply** or **OK**.
   g. Click **Save** to save the configuration changes to the master configuration.

8. Optional: Configure session persistence. If you require session failover capability, you must configure session persistence. Persistence of the session object to a database is the only option for session failover with WebSphere Application Server. To configure session persistence using the administrative console, see the documentation on configuring database session persistence. To configure database session persistence using wsadmin commands for the Jython or Jacl programming language, see the documentation on configuring database session persistence using scripting.

9. Restart the server.

10. Generate the `plugin-cfg.xml` file for each stand-alone application server using the GenPluginCfg script, the administrative console, or wsadmin scripting.

    To use the GenPluginCfg script, enter the following command on the command line:*profile_root*/`config/cells/GenPluginCfg.sh|bat`

    To use the administrative console, see the documentation on creating or updating a global web server plug-in configuration file.

    The following variables apply to the Jython and Jacl commands:

    - *cell_name* is the name of your cell.
    - *web_server_node* is the name of the node for your web server.
    - *web_server_name* is the name of your web server.

    Jython

    　　On the command line, enter each of the following commands on a separate line:

```
        generator = AdminControl.completeObjectName('type=PluginCfgGenerator,*')
        AdminControl.invoke(generator, 'generate', "profile_root/config cell_name web_server_node web_serve
```

Jacl   On the command line, enter each of the following commands on a separate line:

```
        set generator [$AdminControl completeObjectName type=PluginCfgGenerator,*]
        $AdminControl invoke $generator generate "profile_root/config cell_name web_server_node web_server_
```

11. Merge the `plugin-cfg.xml` files from multiple application server nodes.

    You can either manually merge the `plugin-cfg.xml` files or use the tool to automatically merge the `plugin-cfg.xml` file from multiple application server profiles into a single output file. The `.bat` or `.sh` tool is available after you install this fix pack and is located in the *install_root*/bin directory. To use the tool, complete the following steps:

    a. Rename the `plugin-cfg.xml` files to a unique name across your application server profiles.

    b. Copy the `plugin-cfg.xml` file for all stand-alone application server profiles into a common directory.

    c. Use the tool to combine the `plugin-cfg.xml` files from each of the application server profiles into a single output file. For example: **AIX**   **HP-UX**   **Linux**   **Solaris**

       install_root/bin/.sh *plugin_configuration_file1 plugin_configuration_file2 resulting_plugin_configuration_file*

       Windows

       install_root\bin\.bat *plugin_configuration_file1 plugin_configuration_file2 resulting_plugin_configuration_file*

       The *resulting_plugin_configuration_file* variable value is normally `plugin-cfg.xml`

    For more information about manually merging the `plugin-cfg.xml` files, see the technote on merging `plugin-cfg.xml` files from multiple application server profiles.

12. Copy the merged `plugin-cfg.xml` file to the *plugin_installation_root*/config/*web_server_name*/ directory on the Web server host.

13. **AIX**   **HP-UX**   **Linux**   **Solaris**   Ensure that you have defined the correct operating system file access permissions for the merged `plugin-cfg.xml` file. These file access permissions allow the HTTP server plug-in process to read the file.

Results

When you complete this process, you have one plug-in configuration file for multiple stand-alone application server profiles.

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.

Important: When the plug-in configuration file is generated, it does not include admin_host on the list of virtual hosts. The Information Center article "Allowing web servers to access the administrative console" describes how to add admin_host to the list of virtual hosts.

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.

For distributed platforms, you can check the related messages in the deployment manager SystemOut.log file to verify that the automatic propagation successfully completed.

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

For IBM i and distributed platforms running in a Base or Express® environment, the application server's SystemOut log contains the status of the automatic plug-in generation and propagation.

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"
- "RefreshInterval" on page 139
- "ASDisableNagle" on page 139
- **Windows** "IISDisableNagle" on page 139
- "VHostMatchingCompat" on page 139
- "AppServerPortPreference" on page 139
- "ResponseChunkSize" on page 139
- "AcceptAllContent" on page 140
- "ChunkedResponse" on page 140
- **Windows** "IISPluginPriority" on page 140
- "TrustedProxyEnable" on page 140
- "TrustedProxyList" on page 141
- "SSLConsolidate" on page 141
- "SSLPKCSDriver" on page 141
- "SSLPKCSPassword" on page 141
- "Log" on page 141
- "Property Name="esiEnable" Value="true/false"" on page 142
- "Property Name="esiMaxCacheSize" Value="integer"" on page 142
- "Property Name="ESIInvalidationMonitor" Value="true/false" " on page 142
- "Property Name="FIPSEnable" Value="true/false" " on page 142
- "Property Name="PluginInstallRoot" Value="C:\IBM\WebSphere\Plugins" " on page 142
- "ServerCluster" on page 143
- "VirtualHostGroup" on page 148
- "UriGroup" on page 149
- "Route" on page 149
- "RequestMetrics" on page 150

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`.

Windows
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.

SSLPKCSDriver

Specifies the fully qualified name of the loadable module that interfaces with an optional SSL co-processor. The fully qualified name must include the directory path and the module name.

SSLPKCSPassword

Specifies the password for the SSL co-processor with which the module, specified for the SSLPKCSDriver custom property, is interfacing.

If you are using an IBM HTTP Server, you can use the sslstash program to create a file that contains this password. In this situation, you can specify the fully-qualified name of that file, instead of the actual password, as the value for this custom property.

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.
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 154 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

SSLPKCSDriver

Specifies the fully qualified name of the loadable module that interfaces with an optional SSL co-processor. The fully qualified name must include the directory path and the module name.

Data type	String
Default	None

SSLPKCSPassword

Specifies the password for the SSL co-processor with which the module, specified for the SSLPKCSDriver custom property, is interfacing.

If you are using an IBM HTTP Server, you can use the sslstash program to create a file that contains this password. In this situation, you can specify the fully-qualified name of that file, instead of the actual password, as the value for this custom property.

Data type	String
Default	None

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 15. 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
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

Table 15. 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 > 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 > Web Servers >** *Web_server_name*> **Plug-in properties > Custom properties > New**	SSLPKCSDriver	SSLPKCSDriver
In the administrative console, click **Servers > Web Servers >** *Web_server_name*> **Plug-in properties > Custom properties > New**	SSLPKCSPassword	SSLPKCSPassword
In the administrative console, click **Servers > Server Types > Web servers >** *Web_server_name* > **Plug-in properties > Request routing**	Load balancing option	LoadBalance
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

Table 15. 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**	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
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

Table 15. 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 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
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.

Windows IBM HTTP Server uses a single, multithreaded process for serving requests. No configuration changes are required.

AIX **HP-UX** **Solaris** IBM HTTP Server typically uses multiple multithreaded processes for serving requests. Specify the following values for the properties in the web server configuration file (httpd.conf) to prevent the IBM HTTP Server from using more than one 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

Windows If you use the default settings for a Microsoft Windows operating system, you might encounter web server plug-in performance problems if you are running in a high stress environment. To avoid these problems, consider tuning the TCP/IP setting for this operating system. Two of the keys setting to tune are TcpTimedWaitDelay and MaxUserPort.

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 (distributed)

The following file paths are default locations. You can install the product and other components or create profiles in any directory where you have write access. Multiple installations of WebSphere Application Server products or components require multiple locations. Default values for installation actions by root and nonroot users are given. If no nonroot values are specified, then the default directory values are applicable to both root and nonroot users.

app_client_root

Table 16. Default installation root directories for the Application Client for IBM WebSphere Application Server.

This table shows the default installation root directories for the Application Client for IBM WebSphere Application Server.

User	Directory
Root	**AIX** `/usr/IBM/WebSphere/AppClient` (Java EE Application client only) **HP-UX** **Linux** **Solaris** `/opt/IBM/WebSphere/AppClient` (Java EE Application client only) **Windows** `C:\Program Files\IBM\WebSphere\AppClient`
Nonroot	**AIX** **HP-UX** **Linux** **Solaris** *user_home*`/IBM/WebSphere/AppClient` (Java EE Application client only) **Windows** `C:\IBM\WebSphere\AppClient`

app_server_root

Table 17. Default installation directories for WebSphere Application Server.

This table shows the default installation directories for WebSphere Application Server.

User	Directory
Root	**AIX** `/usr/IBM/WebSphere/AppServer` **HP-UX** **Linux** **Solaris** `/opt/IBM/WebSphere/AppServer` **Windows** `C:\Program Files\IBM\WebSphere\AppServer`
Nonroot	**AIX** **HP-UX** **Linux** **Solaris** *user_home*`/IBM/WebSphere/AppServer` **Windows** *user_home*`\IBM\WebSphere\AppServer`

component_root
> The component installation root directory is any installation root directory described in this topic. Some programs are for use across multiple components—in particular, the Web Server Plug-ins, the Application Client, and the IBM HTTP Server. All of these components are part of the product package.

gskit_root
> IBM Global Security Kit (GSKit) can now be installed by any user. GSKit is installed locally inside

the installing product's directory structure and is no longer installed in a global location on the target system. The following list shows the default installation root directory for Version 8 of the GSKit, where *product_root* is the root directory of the product that is installing GSKit, for example IBM HTTP Server or the web server plug-in.

| AIX | HP-UX | Linux | Solaris |

`product_root/gsk8`

| Windows |

`product_root\gsk8`

profile_root

Table 18. Default profile directories.

This table shows the default directories for a profile named profile_name on each distributed operating system.

User	Directory
Root	**AIX** `/usr/IBM/WebSphere/AppServer/profiles/profile_name` **HP-UX** **Linux** **Solaris** `/opt/IBM/WebSphere/AppServer/profiles/profile_name` **Windows** `C:\Program Files\IBM\WebSphere\AppServer\profiles\profile_name`
Nonroot	**AIX** **HP-UX** **Linux** **Solaris** `user_home/IBM/WebSphere/AppServer/profiles` **Windows** `user_home\IBM\WebSphere\AppServer\profiles`

plugins_root

Table 19. Default installation root directories for the Web Server Plug-ins.

This table shows the default installation root directories for the Web Server Plug-ins for WebSphere Application Server.

User	Directory
Root	**AIX** `/usr/IBM/WebSphere/Plugins` **HP-UX** **Linux** **Solaris** `/opt/IBM/WebSphere/Plugins` **Windows** `C:\Program Files\IBM\WebSphere\Plugins`
Nonroot	**AIX** **HP-UX** **Linux** **Solaris** `user_home/IBM/WebSphere/Plugins` **Windows** `C:\IBM\WebSphere\Plugins`

wct_root

Table 20. Default installation root directories for the WebSphere Customization Toolbox.

This table shows the default installation root directories for the WebSphere Customization Toolbox.

User	Directory
Root	**AIX** `/usr/IBM/WebSphere/Toolbox` **HP-UX** **Linux** **Solaris** `/opt/IBM/WebSphere/Toolbox` **Windows** `C:\Program Files\IBM\WebSphere\Toolbox`

Table 20. Default installation root directories for the WebSphere Customization Toolbox (continued).

This table shows the default installation root directories for the WebSphere Customization Toolbox.

User	Directory
Nonroot	**AIX** **HP-UX** **Linux** **Solaris** *user_home*/IBM/WebSphere/Toolbox **Windows** C:\IBM\WebSphere\Toolbox

web_server_root

Table 21. Default installation root directories for the IBM HTTP Server.

This table shows the default installation root directories for the IBM HTTP Server.

User	Directory
Root	**AIX** /usr/IBM/HTTPServer **HP-UX** **Linux** **Solaris** /opt/IBM/HTTPServer **Windows** C:\Program Files\IBM\HTTPServer
Nonroot	**AIX** **HP-UX** **Linux** **Solaris** *user_home*/IBM/HTTPServer **Windows** C:\IBM\HTTPServer

Notices

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM's product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any of IBM's intellectual property rights may be used instead of the IBM product, program, or service. Evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, is the user's responsibility.

APACHE INFORMATION. This information may include all or portions of information which IBM obtained under the terms and conditions of the Apache License Version 2.0, January 2004. The information may also consist of voluntary contributions made by many individuals to the Apache Software Foundation. For more information on the Apache Software Foundation, please see http://www.apache.org. You may obtain a copy of the Apache License at http://www.apache.org/licenses/LICENSE-2.0.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Intellectual Property & Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
USA

163

Trademarks and service marks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol ($^{®}$ or $^{™}$), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. For a current list of IBM trademarks, visit the IBM Copyright and trademark information Web site (www.ibm.com/legal/copytrade.shtml).

Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Other company, product, or service names may be trademarks or service marks of others.

Index

D
directory
   installation