WebSphere® Application Server for IBM i, Version 7.0

Setting up intermediary services

Note

Before using this information, be sure to read the general information under “Notices” on page 163.

Compilation date: September 11, 2008

© Copyright International Business Machines Corporation 2008.
US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

© Copyright IBM Corp. 2008

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.

© Copyright IBM Corp. 2008

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.

© Copyright IBM Corp. 2008

Chapter 1. Overview and new features

Introduction: Web servers

An application server works with a Web server to handle requests for dynamic content, such as servlets, from Web applications. A Web server uses a Web server plug-ins to establish and maintain persistent HTTP and HTTPS connections with an application server.

The Supported Hardware and Software Web page provides the most current information about supported Web servers.

Chapter 2, “Communicating with Web servers,” on page 3 describes how to set up your Web server and Web server plug-in environment and how to create a Web server definition. The Web server definition associates a Web server with an application server. After you create a Web server definition, you can use the administrative console to perform the following functions for that Web server:

- Check the status of the Web server
- Generate a plug-in configuration file for that Web server.

You can not propagate a plug-in configuration file for a non-IBM HTTP Server. You must manually install an updated plug-in configuration file on that Web server.

After you set up your Web server and Web server plug-in, whenever you deploy a Web application, you must specify a Web server as the deployment target that serves as a router for requests to the Web application. The configuration settings in the plug-in configuration file (plugin-cfg.xml) for each Web server are based on the applications that are routed through that Web server. If the Web server plug-in configuration service is enabled, a Web server plug-in’s configuration file is automatically regenerated whenever a new application is associated with that Web server.

Note: Before starting the Web server, make sure you are authorized to run any Application Response Measurement (ARM) agent associated with that Web server.

Refer to your Web server documentation for information on how to administer that Web server. For tips on tuning your Web server plug-in, see “Web server plug-in tuning tips” on page 56.

© Copyright IBM Corp. 2008

Chapter 2. Communicating with Web servers

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 in 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

- Install your Web server if it is not already installed.

  Note: The Web server that is provided with i5/OS®, is already installed under product 5722-DG1 for i5/OS V5R4 or 561-DG1 for i5/OS V6R1. The i5/OS Web server is referred to as the IBM® HTTP Server for i5/OS. This Web server is different from the IBM HTTP Server that is provided with WebSphere Application Server, which does not run on i5/OS.

  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 perform the operations that are required by Web applications, such as GET and POST. Typically, configuring your Web server to perform these operations involves setting a directive in the Web server configuration file. Refer to the Web server documentation for instructions.

About this task

The appropriate plug-in file is installed. In addition, an http profile is created (/QIBM/UserData/WebSphere/Plugins/V61/webserver/profiles/http). The http profile can be used to facilitate the creation of Web server definitions. Refer to “Selecting a Web server topology diagram and roadmap” on page 6 for instructions on how to configure IBM HTTP Server for i5/OS to communicate with an application server.

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

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 or not to have the configuration automatically generated in response to a configuration change. When the Web server plug-in configuration service is enabled and any of the following conditions occur, the plug-in configuration file is automatically generated:
   - When the Web server is created or saved.
   - When an application is installed.
   - When an application is uninstalled.
   - When the virtual host definition is updated

   You can either use the administrative console, or issue the GenPluginCfg command to regenerate your plugin-cfg.xml file. To use 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:
      - Caching
      - Request and response
      - Request routing
      - Custom Properties

      Web server plug-in configuration properties maps each property to one of these topics.

Note: It is recommended that you do not manually update the plugin-cfg.xml file. Any manual updates you make for a given Web server are overridden whenever the plugin-cfg.xml file for that Web server is regenerated.

c. Click **OK**.

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

Another method to propagate the plug-in configuration is to run the GenPluginCfg command.

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

Note: If you use the FTP function to perform the copy, and the configuration reload fails, check the file authorities on the plugin-cfg.xml file and make sure that users QTMHHTTP, QNOTES and QEJBSVR have RWX authority. If the authorities are not correct, the Web server will not be able to access the new version of the file, which causes the configuration reload to fail. To check the authorities, run the following i5/OS command:

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

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

If the authorities are incorrect, issue the following i5/OS command to change the file authorities to the appropriate settings:

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

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

e. You might have to stop the application server and then start the application server again to enable the Web server to locate the plugin-cfg.xml file.

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

Results

The configuration is complete. To activate the configuration, stop and restart the Web server. If you encounter problems restarting your Web server, check the http_plugin.log file for information on 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, you should consider enabling this service.

If you are making a series of simultaneous changes, like 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 elect **Servers > Server Types > WebSphere application servers >** *server_name* **> administration services > Web server plug-in configuration service**, and then unselect the **Enable automated Web server configuration processing** option.

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

Installing IBM HTTP Server

This topic describes how to install IBM HTTP Server.

Before you begin

IBM HTTP Server for iSeries®. The IBM HTTP Server product is included in the media package when you order any of the WebSphere® Application Server for i5/OS products. It is not a required product. To install the product on a non-iSeries server, use the following instructions.

To use a Web server other than IBM HTTP Server, install and configure the Web server before or after installing the WebSphere Application Server product, but before installing the Web server plug-ins for 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 the appropriate Web server plug-in for a supported, installed Web server. No further configuration is required for most Web servers.

The Plug-ins installation wizard configures supported Web servers. You can also manually configure supported Web servers for WebSphere Application Server as described in “Editing Web server configuration files” on page 27.

Related tasks

Chapter 2, “Communicating with Web servers,” on page 3
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 in an application server. The Web server plug-in uses the XML configuration file to determine whether a request is for an application server.

Installing Web server plug-ins on i5/OS

You can install a Web server plug-in on your i5/OS system to route requests to a WebSphere Application Server installation.

Before you begin

If you are running the application server on the same i5/OS server or logical partition as your Web server, you do not need to install a Web server plug-in. The Web server plug-in code and an HTTP profile template are included in the WebSphere Application Server installation.

If you are running your application server on a different i5/OS system or logical partition as the Web server, complete the steps in this topic to install a Web server plug-in for your i5/OS Web server.

If you are running your Web server on a platform other than i5/OS, read the documentation on installing Web server plug-ins for the platform that you are using.

About this task

This topic describes installing Web server plug-ins for i5/OS. WebSphere Application Server products supply a unique binary plug-in module for each supported Web server. The plug-in configuration file, which the 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, and to locate specific applications on the application server.

On platforms other than i5/OS, the plug-ins installation wizard installs the plug-in module, configures the Web server for communicating with the application server, and creates a Web server configuration definition in the application server if possible.

On i5/OS, the Web server plug-ins installation does not configure the Web server or create a Web server configuration definition. After installing the Web server plug-ins on i5/OS, use the IBM Web Administration for i5/OS wizards to configure your Web server instance.

You can install a Web server plug-in on your i5/OS system using the installation wizard in remote graphical interface or silent mode.

- Run the installation wizard graphical interface from a Windows® workstation.

  See "Installing Web server plug-ins using a Windows workstation graphical interface" on page 11.
- Install a Web server plug-in silently.

  You can run the silent installation on a local i5/OS system or remotely. See "Installing Web server plug-ins silently on i5/OS" on page 13

Results

You install a Web server plug-in on your i5/OS system. A default profile is created to act as the location for putting key, certificate, or log files. The profile name is http and the profile path is *plugins_user_data_root*/ profiles/http.

What to do next

After installing a Web server plug-in on i5/OS, use the IBM Web Administration for i5/OS wizards to configure your Web server instance. Read the ″Configuring an HTTP server instance″ topic for more information. The IBM Web Administration for i5/OS wizards perform the same the function as the Web server plug-ins installation on other platforms.

You can create additional profiles to use with additional Web server instances.

Selecting a Web server topology diagram and roadmap

Install and configure Web server plug-ins for WebSphere Application Server 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 Plug-ins installation wizard has four main tasks:

- Installs the binary plug-in module on the Web server machine.
- 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 wizard processes the creation of the Web server definition differently depending on the scenario:
  - Recommended remote stand-alone application server installation:

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

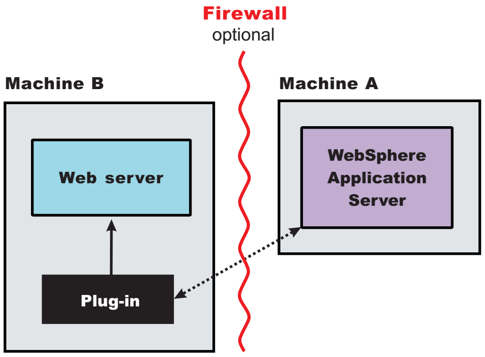
    Detects the default profile on a local application server machine and creates the Web server definition for it directly. Install the Web server and its plug-in on the same machine with the application server. This configuration is for development and test environments.

Select a link to go to the appropriate steps in the following procedure.

- **Set up a remote Web server installation for a Standalone node.**

  The remote Web server configuration is recommended for production environments.

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



Remote installation scenario

Table 1. Installation and configuration

Step	Machine	Task
1	A	Install your WebSphere Application Server product. Read the ″Installing the product and additional software″ topic.
2	A	Create an application server profile.
3	B	If you plan to run IBM HTTP Server on iSeries, it is already installed as product 5722-DG1 or 5761-DG1. You can also use a Domino® Web server on iSeries. Refer to the Domino documentation for installation instructions. For either scenario, you must install the Web server Plug-ins component of the WebSphere Application Server product. See “Installing Web server plug-ins on i5/OS” on page 5.
4	B	Run the **manageprofiles** Qshell command to create an http profile. For example, run this command from Qshell: `app_server_root/bin/manageprofiles -create -profileName myHttpProfile -templatePath http` The *myHttpProfile* variable is the name of the profile.

Table 1. Installation and configuration (continued)

Step	Machine	Task

5	B	Configure the IBM HTTP Server with your http profile myHttpProfile.
		As a result of the above, an i5/OS qshell script called *configureIHS_MyWebServer* will be created in the myHttpProfile_profile_root/config/IHS_myWebServer directory on machine B. For the default WebSphere Application Server install, the*myHttpProfile_profile_root* of the profile myHttpProfile is /QIBM/UserData/WebSphere/AppServer/V61/Base/profiles/myHttpProfile.
		Note: In the remainder of this example, webServerName refers to IHS_myWebServer. If you choose to configure a DOMINO Web server as listed below, then webServerName refers to DOMSRV01.
		The following steps apply to **DOMINO Web servers only**:
		1. Run the *configureOs400WebServerDefinition* script on the http profile myHttpProfile. For example:
		`configureOs400WebServerDefinition -profileName myHttpProfile`
		`   -webserver.name DOMSRV01 -webserver.type DOMINO -webserver.port 80`
		2. Using the WRKDOMSVR command to update the notes.ini file of your Domino server, insert the following directive: WebSphereInit=myHttpProfile_profile_root/config/DOMSRV01/plugin-cfg.xml
		3. From the Lotus Notes® client connected to the Domino server, click the internet protocols tab and then click the HTTP tab. Under DSAPI filter names add the following: /QSYS.LIB/ *product_lib*.LIB/LIBDOMINO.SRVPGM
		4. Save your changes

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

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

9	A	If you use IBM HTTP Server on iSeries, verify that the application server is running. Open the administrative console (ISC) and do the following: 1. Expand Servers > Web servers 2. Select your Web server, in this case it is *IHS_MyWebServer*, then click Remote web server management 3. Enter the user ID and password used to authenticate to Machine B. The authorities required by this profile are the same as that required to access the HTTP administration GUI. For details, see User profiles and required authorities for HTTP Server in the iSeries Information Center. 4. Save your configuration.
10	A	Configure a virtual host alias for the Web server machine(B) and Web server port of MyWebServer.

Table 1. Installation and configuration (continued)

Step	Machine	Task
11	A	Stop and restart your application server.
12	A	In the administrative console (ISC) do the following: 1. Select *webServerName* and click Generate Plug-in to generate the plugin-cfg.xml file. 2. Select *webServerName* and click Propagate Plug-in to propagate the plugin-cfg.xml file to machine B.
13	B	If you use IBM HTTP Server on iSeries, start the Web server. Open the administrative console (ISC) and do the following: 1. Expand Servers > Web servers. 2. Select your Web server, in this case it is *IHS_MyWebServer*, then click Start. If you use Domino HTTP Server on iSeries, start the Web server from a CL command line: 1. Run the Work with Domino Servers (WRKDOMSVR) command. 2. Specify option 1 next to your Domino server. 3. Press Enter.
14	B	Run the snoop servlet. Access the following URL in your browser: `http://host_name_of_machine_B:http_transport_port/snoop` If you get an error, retrace your steps. Add a virtual host to Machine A before restarting the application server on Machine A.

Regeneration of the plugin-cfg.xml file

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

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

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

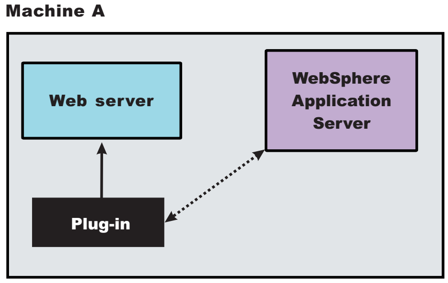
Propagation of the plugin-cfg.xml file

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

- **Set up a local Web server configuration for a Standalone node.**

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

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



Local installation scenario

Table 2. Installation and configuration

Step	Machine	Task
1	A	Install your WebSphere Application Server product.
2	A	Create an application server profile.
3	A	IBM HTTP Server on iSeries is already installed as product 5722-DG1 or 5761-DG1. Alternatively, you can also run Domino Web Server on iSeries. Refer to the Domino documentation for installation instructions.

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

5	A	Stop and restart the application server.
6	A	If you use IBM HTTP Server on iSeries, open the administrative console (ISC) do the following: 1. Expand Servers > Web servers 2. Select your Web server, in this case it is *IHS_MyWebServer*, then click Remote web server management. 3. Enter the user ID and password used to authenticate to Machine A. The authorities required by this profile are the same as that required to access the HTTP administration GUI. For details, see User profiles and required authorities for HTTP Server in the iSeries Information Center. 4. Save your changes.
7	A	In the administrative console (ISC) do the following: 1. Expand Servers > Web servers. 2. If you use IBM HTTP Server on iSeries, Select *IHS_MyWebServer* and click Generate Plug-in to generate the plugin-cfg.xml file. 3. If you use Domino HTTP Server on iSeries, Select DOMSRV01 and click Generate Plug-in to generate the plugin-cfg.xml file.

Table 2. Installation and configuration (continued)

Step	Machine	Task
8	A	If you use IBM HTTP Server on iSeries, start the Web server. Open the administrative console (ISC) and do the following: 1. Expand Servers > Web servers. 2. Select your Web server, in this case it is *IHS_MyWebServer*, then click Start. If you use Domino HTTP Server on iSeries, start the Web server from a CL command line: 1. Run the Work with Domino Servers (WRKDOMSVR) command. 2. Specify option 1 next to your Domino server. 3. Press Enter.
9	A	Run the snoop servlet. Access the following URL in your browser: `http://host_name_of_machine_A:http_transport_port/snoop` If you get an error, retrace your steps. Add a *virtual host* to Machine A before restarting the application server on Machine A.

Regeneration of the plugin-cfg.xml file

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

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

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

Propagation of the plugin-cfg.xml file

The local file does not require propagation.

Results

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

What to do next

See “Editing Web server configuration files” on page 27 for information about how the Plug-ins installation wizard configures supported Web servers.

See “Installing Web server plug-ins on i5/OS” on page 5 for information about other installation scenarios for installing Web server plug-ins.

Installing Web server plug-ins using a Windows workstation graphical interface

You can install a Web server plug-in on your i5/OS system using the installation wizard graphical interface run on a Windows workstation.

Before you begin

If you are running the application server on the same i5/OS server or logical partition as your Web server, you do not need to install a Web server plug-in. The Web server plug-in code and an HTTP profile template are included in the WebSphere Application Server installation.

If you are running your application server on a different i5/OS system or logical partition as the Web server, complete the steps in this topic to install a Web server plug-in for your i5/OS Web server.

About this task

You run the installation wizard on a Windows workstation for remote installation of the Web server plug-in to an i5/OS system. This topic describes how to use the installation wizard graphical interface to install the plug-in.

1. If TCP/IP is not started or if you don't know if TCP/IP is started, enter the Start TCP/IP (STRTCP) command on the Control Language (CL) command line.
2. Verify that the host server jobs are started on your iSeries server. The host server jobs allow the installation code to run on iSeries.

   On a CL command line, enter:

   ```
   STRHOSTSVR SERVER(*ALL)
   ```
3. Verify that your user profile has the *ALLOBJ and *SECADM special authorities.
4. Place the WebSphere Application Server **Supplements** disc in the disc drive of your workstation.
5. Run the launchpad and select **Launch the installation wizard for Web server plug-ins**.

   Read the ″Using the launchpad to start the installation″ topic for more information.
6. Enter your user profile, password, and the host name or IP address for the system on which you are installing the plug-in, and then click **OK**.
7. In the installation wizard, do the following:
   a. On the Welcome panel, deselect **Installation roadmap: overview and installation scenarios** and click **Next**.

   b. On the Software License Agreement panel, select **I accept the terms in the license agreement** message if you agree to the license agreement, and click **Next** to continue.

   c. On the System prerequisites check panel:
      - If the operating system prerequisite check shows **Passed**, click **Next**.
      - If the check shows **Failed**, click **Cancel**. Install operating system service packs and patches needed for the plug-in, and then try installing the Web server plug-in again.

      Web server plug-ins might function correctly even if some prerequisites are not installed on the system. However it is recommended that you update your system to the required level.

   d. If more than one Java™ SE Development Kit 6 (JDK 6) option is installed on the machine, then select one of the following.
      - IBM Technology for JDK 6 - 32 bit (5722-JV1 or 5761JV1 option 11)
      - IBM Technology for JDK 6 - 64 bit (5722-JV1 or 5761JV1 option 12)
      - IBM Development Kit for Java 1.6 (5722-JV1 or 5761JV1 option 10)

   e. When prompted to select the Web server to configure, select **None** and click **Next**.

   f. On the Installation directory panel, specify the directories to which to install the plug-in and default profile, and then click **Next**.

      Specify the target directories for the plug-in and default profile if you do not want to use the default directories.

      Note: The installation directory cannot contain any unsupported characters. See Object names: What the name string cannot contain for more information.

      Deleting the default target location and leaving an installation directory field empty prevents you from continuing the installation process.

   g. On the Installation Summary panel, click **Next**.

      The installation wizard copies Web server plug-in files to your system and creates a profile to act as the location for putting key, certificate, or log files.

   h. On the Installation Results panel, click **Finish**.
8. Verify the success of the installer program by examining installation wizard messages and the *plugins_root*/logs/install/log.txt file for installation status.

   The installation program records the following indicators of success in the logs:

   - INSTCONFSUCCESS indicates that the installation is successful and that no further log analysis is required.
   - INSTCONFFAILED indicates an installation failure that you cannot retry or recover from without reinstalling.

Results

You installed a Web server plug-in and profile on your i5/OS system.

What to do next

Use the IBM Web Administration for i5/OS wizards to configure your Web server instance. Read the ″Configuring an HTTP server instance″ topic for more information. The IBM Web Administration for i5/OS wizards perform the same the function as the Web server plug-ins installation on other platforms.

Create a profile for each additional Web server instance.

Installing Web server plug-ins silently on i5/OS

You can install a Web server plug-in on your i5/OS system silently using the installation wizard run locally on the i5/OS system or remotely from a Windows workstation.

Before you begin

If you are running the application server on the same i5/OS server or logical partition as your Web server, you do not need to install a Web server plug-in. The Web server plug-in code and an HTTP profile template are included in the WebSphere Application Server installation.

If you are running your application server on a different i5/OS system or logical partition as the Web server, complete the steps in this topic to install a Web server plug-in for your i5/OS Web server.

About this task

A silent installation uses the installation wizard to install the product in silent mode, without the graphical user interface. Instead of displaying a wizard interface, the silent installation causes the installation program to read all of your responses from a file that you provide. To specify non-default options during a silent installation, you must use the response file. To install silently, you must accept the license agreement in the agreement option.

You can run the installation wizard on the i5/OS system that has the Web server or remotely from a Windows workstation.

1. If you are installing remotely, start TCP/IP.

   Enter the Start TCP/IP (STRTCP) command on the Control Language (CL) command line.
2. Verify that the host server jobs are started on your iSeries server. The host server jobs allow the installation code to run on iSeries.

   Enter this command on a CL command line:

   ```
   STRHOSTSVR SERVER(*ALL)
   ```
3. Verify that your user profile has the *ALLOBJ and *SECADM special authorities.
4. Place the WebSphere Application Server **Supplements** disc in the disc drive of your iSeries server if you are installing locally or your workstation if you are installing remotely.

5. Customize the options response file.
   a. Locate the sample options response file.

      The file name is responsefile.txt in the operating system platform directory on the product disc.

   b. Use the Copy (CPY) command to copy the responsefile.txt file to a directory of your choosing.

      For example, to copy the file from a CD-ROM on i5/OS:

      ```
      CPY OBJ('/QOPT/WEBSPHERE/plugin/responsefile.txt') TODIR('/MYDIR')
      ```

      For DVD media, specify an additional directory prefix before the product directory that reflects the operating system and architecture of the installation. For example:

      ```
      CPY OBJ('/QOPT/WEBSPHERE/os400_ppc64/plugin/responsefile.txt') TODIR('/MYDIR')
      ```

      where:
      - `QOPT` is the disk mount point.
      - `WEBSPHERE` is the disk volume label.
      - `/plugin` is the product directory on the disc.

   c. Edit the copy of the file using an editor such as EDTF from the i5/OS command line or another editor from a mapped drive.

      Accept the license agreement by changing the value for `–OPT silentInstallLicenseAcceptance` from `false` to `true`. This is the only required change in the response file.

      Read the information in the response file to determine if you want to change any other options. You can change the `installLocation` and `defaultProfileLocation` values.

      Specify a Java Software Development Kit (SDK) to invoke the installer and configure the product and profiles. You can force the use of a specific Java SDK through the `i5osjdklocation` option in the response file. Read the “Web server plug-in response file” on page 22 topic for more information.

      Note: The installation directory cannot contain any unsupported characters. See Object names: What the name string cannot contain for more information.

   d. Save the file.

6. Invoke the installation wizard for Web server plug-ins.

   You can run the installation wizard on the i5/OS system that has the Web server or remotely from a Windows workstation.

   - Run the install command locally from Qshell:
     a. From the i5/OS command line, start a Qshell session: `STRQSH`
     b. Change directory to the plugin subdirectory.

        For a CD-ROM drive, enter: `cd /QOPT/WEBSPHERE/plugin`

        For a DVD-ROM drive, enter: `cd /QOPT/WebSphere/os400_ppc64/plugin`

     c. Issue the install command to start the installation wizard.

        ```
         install –options responsefile
        ```

        where *responsefile* is the fully qualified path name of the response file that you edited.

        For systems with little or no interactive feature, use SBMJOB:

        ```
        SBMJOB CMD(STRQSH CMD('/QOPT/WEBSPHERE/plugin/install -options responsefile')) JOB(INSTWAS) JOBQ(QSYSNOMAX) ALWML
        ```

        Note: Do not exit the Qshell session (PF3) until the installation has completed. Doing so might cause the installation to stop prematurely.

   - Run the install command remotely from a Windows workstation:
     a. Change directory to the plugin subdirectory of the CD-ROM or DVD-ROM drive.
     b. Issue the install command to start the installation wizard.

        ```
        install.exe hostname_or_IP userid password –options responsefile
        ```

        where
        – *hostname_or_IP* is the host or IP address of the i5/OS system.

- *userid* is an i5/OS user profile with *ALLOBJ and *SECADM authority
- *password* is the password for the user ID.
- *responsefile* is the fully qualified path name of the response file that you edited.

7. Verify the success of the installer program by examining installation wizard messages and the *plugins_root*/logs/install/log.txt file for installation status.

   The installation program records the following indicators of success in the logs:

   - INSTCONFSUCCESS indicates that the installation is successful and that no further log analysis is required.
   - INSTCONFFAILED indicates an installation failure that you cannot retry or recover from without reinstalling.

Results

You installed a Web server plug-in and profile silently by using the response file.

What to do next

Use the IBM Web Administration for i5/OS wizards to configure your Web server instance. Read the ″Configuring an HTTP Server instance″ topic for more information. The IBM Web Administration for i5/OS wizards perform the same the function as the Web server plug-ins installation on other platforms.

Create a profile for each additional Web server instance.

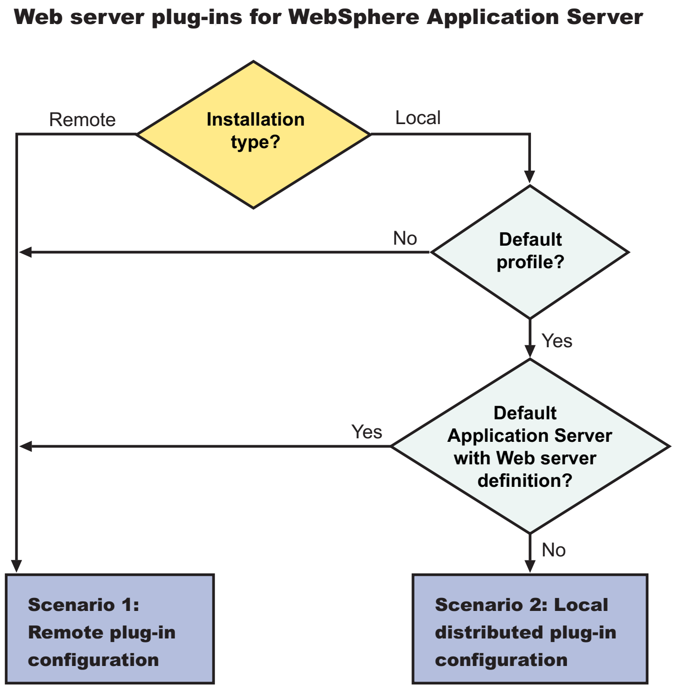
Plug-ins configuration

The Plug-ins installation wizard installs a binary plug-in module and a plug-in configuration file for the Web server. The wizard then configures the supported Web server for the Application Server and creates a Web server definition in the configuration of the application server. This overview shows the different processing paths that the wizard uses.

This topic describes the two ways that the Plug-ins installation wizard configures a Web server to locate the plugin-cfg.xml file, which is the plug-in configuration file.

Configuration flows

The Plug-ins installation wizard resolves all configurations of the Web server and WebSphere Application Server to two scenarios: remote plug-in configuration or 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
Remote
Installation type?
Local
No
Default profile?
Yes
Yes
Default Application Server with Web server definition?
No
Scenario 1: Remote plug-in configuration
Scenario 2: Local distributed plug-in configuration

Legend:

Default application server with Web server definition?
If the default profile has an existing Web server definition, then the installation is considered a remote plug-in configuration. You cannot have more than one Web server definition in a stand-alone application server.

Use the same name for the Web server to configure a new Web server to use the existing Web server definition.

Default profile?
If the product is installed but the default profile is accidentally deleted or otherwise missing, the scenario is considered to be a remote installation. Create a profile before running the script. When multiple profiles exist, the plug-ins installer configures only the default profile.

Installation type?
The installation type is either remote or local.

Scenario A. Remote plug-in configuration

The Plug-ins installation wizard does not automatically create a Web server definition within the default distributed profile on a remote machine. The wizard creates the configure*web_server_name* script instead.

The Plug-ins installation wizard 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.

Three configurations 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 Plug-ins installation wizard	By script	N/A
No default profile detected	By script	N/A
Default stand-alone 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:

   - ./*IHS_root*/bin/apachectl start
2. Start the application server on the remote machine.

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

   - `./`*`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:

      - /configure*web_server_name*

   The configure*web_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. If it is blank, the script will use the default profile.
   - *Admin_Console_Username* indicates the username of the admin console. The profile with the admin console deployed must have admin 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 can not be used if both *profile_name* and *Admin_Console_Username* are blank.

Note: The *webserverNodeName* value in the script is a concatenation of the nick name you have chosen for the web server and the suffix *-node*. It is automatically created during plug-in installation and cannot be changed. For example, if you named your web server *myserver* during plug-in installation, 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.

Scenario B. Local stand-alone plug-in configuration

In this scenario, the Plug-ins installation wizard creates a Web server definition within the application server profile directly, without the use of a script.

The Plug-ins installation wizard 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 installing the binary plug-in for the local Web server, you can start the application server and the Web server immediately upon completion of the installation.

Only one configuration 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 A

A default application server profile that has an existing Web server definition is processed as a remote plug-in configuration.

An existing Web server definition on an application server profile causes the Plug-ins installation wizard to follow the remote installation path. An application server can have just one Web server definition. Specify the same nick name for the Web server to configure a new Web server to use the existing Web server definition.

You can use the plugin-cfg.xml file that is within the Web server definition in the configuration of the application server. Simply click **Browse** on the appropriate panel in the Plug-ins installation wizard to select the file. This file must exist. Otherwise, the Plug-ins installation wizard displays a warning and prevents you from proceeding until you select an existing file. The Web server is configured to use this existing plugin-cfg.xml file.

See Scenario A 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 after installing the binary plug-in module:

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

   For example, start the IBM HTTP Server from a command line:

   - ./*IHS_root*/bin/apachectl start
2. Start the application server.

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

   - `./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.

Summary

Two scenarios exist for Web server plug-ins for WebSphere Application Server. 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 such objects as applications and virtual hosts for serving applications, for example.

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 A** 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.

The configuration scheme for Version 6 of WebSphere Application Server puts the plug-in configuration file in a Web server definition that is within a Web server node. All **Scenario B** 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 stand-alone Application Server. The inability of a stand-alone application server to create a Web server definition is the basis for the configuration scripts created by the Web server plug-ins for WebSphere Application Server. Without the scripts you could not easily create a Web server definition on a stand-alone 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 3. Plug-in configuration file locations

Scenario	**Profile type**	**Location of the plugin-cfg.xml file**	
		Plug-ins_ install_ root	**profiles_ root: within the Web server node**
A	Any profile anywhere if you select a remote installation type in the Plug-ins installation wizard	X	
	No default profile detected	X	
	Default application server profile with an existing Web server definition	X	
B	Default application server profile		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
```

Web server configuration

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

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

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

See the following descriptions of each file.

Web server configuration file

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

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

- The binary plug-in file
- The plugin-cfg.xml configuration file

The binary Web server plug-in file

See “Web server plug-ins” on page 53 for a description of the binary plug-in module.

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

Another example of a binary plug-in module is the QSVTAP20 service program on the iSeries 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. See "Web server plug-in configuration service property" on page 46 for examples of when the file gets regenerated and when it does not.

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

The plug-in configuration file, plugin-cfg.xml

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

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

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

On i5/OS systems, the plug-in is not automatically generated. You must regenerate and propagate the file manually.

See "Web server plug-in configuration service property" on page 46 for more information.

The configuration script for the Web server definition

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

Copy the script from the Web server machine to the *app_server_root*/bin directory in the i5/OS partition. Run the script to create a Web server definition in the configuration of the application server.

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

- Host name
- Administrative port
- User ID
- Password

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

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

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

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

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

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

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

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

Web server plug-in response file

This topic describes the response file for performing a silent installation of the Web server plug-ins for WebSphere Application Server.

Install the product silently using an options response file.

The responsefile.txt file has directives that set installation options. Comments in the file describe how to set the string value for each directive.

Use the options file to run the Plug-ins installation wizard in silent mode, which is referred to as installing silently. The wizard reads the options file to determine responses and does not display the graphical user interface.

Location of the response file

The sample options response file is named responsefile.txt. The file is in the plugin directory on the product disc or in the downloaded installation image.

Mode of use

The Plug-ins installation wizard can read an existing options response file and run silently without displaying the graphical user interface.

Installing silently

The options file supplies the values to the plug-ins installation wizard when installing silently. The wizard reads the options file to determine responses and does not display the graphical user interface. Use the following command to use a copy of the options file named myresponsefile.txt for a silent installation:

```
install -options myresponsefile.txt
```

Creating an operational environment

The installation of the plug-ins is a three-step process:

1. Installing the binary plug-in modules for supported Web servers
2. Configuring the Web servers to use the binary module to communicate with the application server
3. Creating a Web server definition in the application server

   As you install an application, you can install it on the Web server definition in addition to the application server. All applications on the Web server definition are listed in its plug-in configuration file. After propagation, the real Web server can access the applications.

The sample options response file, responsefile.txt, controls installing the binary plug-ins, configuring the Web server, and creates a script for creating the Web server definition on a remote application server machine. The script is customized according to values supplied in the responsefile.txt file. The script is generated to run on the application server machine to create the Web server definition.

If the Web server is on the same machine as a stand-alone application server, the responsefile.txt file can create the Web server definition directly without creating a script.

To edit and use the response file for installing the plug-ins and configuring the Web server and application server, perform the following procedure:

1. Copy the responsefile.txt file from the plugins directory on the product disc to a place that you can easily identify on your machine.
2. Edit the file to customize the values for your installation.
3. Save the file.
4. Start the installation. For example:

   ```
   install -options path/myresponsefile.txt
   ```
5. After the installation, examine the logs for success.

Logging

If no installation logs exist, refer to temporary log file, log.txt in your <userhome>/plglogs directory. You can also cause ISMP to record status about a problem that is preventing the installation from occurring, as described in the following section.

For example, if you start the silent installation without accepting the license in the `-OPT silentInstallLicenseAcceptance="false"` directive, the installation does not occur. The fact that the license entry was not accepted is recorded in log.txt in the <userhome>/plglogs directory.

If any response file validation results in a failure, the failure is listed in the plglogs file, then the installation fails.

If all validations pass, the installation occurs. Then, the Plug-ins installation wizard records installation events in the following log files. The log files are in the *plugins_root*/logs/install directory:

log.txt Records all of the ISMP events that occur during the installation. The log also describes whether the installation was local or remote. Messages at the end of the file indicate whether manual configuration steps are required to complete the installation.

Key elements to look for in the installation record are:

Manual steps warning

When the wizard requires you to run a script to create the Web server definition, the wizard refers to the fact that manual steps are required.

If manual steps are required, the name and location of the script that you must run are written in the log file at the end of the installation record.

Web server type

The log has a record of the Web server type, such as IHS for the IBM HTTP Server, for example.

Location of the plug-in configuration file

The log has a record of the plugin-cfg.xml file location currently in the Web server configuration.

installconfig.log

Lists all of the configuration events that occur during the installation.

installGSKit.log

Lists events that occur during the installation of the GSKit code.

The command line for the installation is listed when the installation occurs. The GSKit 7 installation record is written after the GSKIT 7 : entry in the log.

install*Web_server_type***Plugin.log**

Records events that occur during the installation of a Web server plug-in. The name of the file varies to reflect the Web server:

- installAPACHEPlugin.log
- installIHSPlugin.log
- installIISPlugin.log
- installSunOnePlugin.log
- installDomino5Plugin.log
- installDomino6Plugin.log
- installDomino7Plugin.log

Each log lists the following critical information:

- The plug-in binary module that is currently installed
- The current location of the plug-in configuration file that is configured for the Web server

configure_*Web_server_type***_webserver.log**

Lists events that occur during the configuration of a Web server plug-in. The name of the file varies to reflect the Web server:

- configure_APACHE_webserver.log
- configure_IHS_webserver.log
- configure_IIS_webserver.log
- configure_SUNJAVASYSTEM_webserver.log
- configure_DOMINO_webserver.log

The configure*Web_server_type_*webserver.log file reports the actions that the Plug-ins installation wizard performs as it updates the Web server configuration file.

In a remote scenario, this log is not present because you must run the script to create the Web server definition manually.

Selecting an SDK for installation and configuration

You can force the use of a specific Java Software Development Kit (SDK) with the `i5osjdklocation` option in the response file. Uncomment one of the following options to select the SDK to use to configure the product.

- IBM Technology for JDK 6 - 32 bit (5722-JV1 or 5761JV1 option 11)

```
-OPT i5osjdklocation="/QOpenSys/QIBM/ProdData/JavaVM/jdk60/32bit"
```

- IBM Technology for JDK 6 - 64 bit (5722-JV1 or 5761JV1 option 12)

```
-OPT i5osjdklocation="/QOpenSys/QIBM/ProdData/JavaVM/jdk60/64bit"
```

- IBM Developer Kit for Java 1.6 (5722-JV1 or 5761JV1 option 10)

```
-OPT i5osjdklocation="/QIBM/ProdData/Java400/jdk6"
```

If no SDK is specified, IBM Technology for JDK 6 - 32 bit will be selected by default, if it is installed. If IBM Technology for JDK 6 - 32 bit is not installed, IBM Technology for JDK 6 - 64 bit will be selected. If IBM Technology for JDK 6 - 32 bit and IBM Technology for JDK 6 - 32 bit are not installed, IBM Developer Kit for Java 1.6 will be selected. If no valid SDKs are installed, the installation will fail.

Response file user entry validation

Validation of the response file has been coded into the installation. If response file validation does not pass, the failure is recorded in the temporaryPluginInstallLog.txt file.

Accepted license agreement

Default directive setting
-OPT silentInstallLicenseAcceptance=″false″

Valid setting
You must set this directive to true to accept the license and install the plug-ins.

Error identifier in temporaryPluginInstallLog.txt
INSTCONFFAILED : LICENSE _ NOT _ ACCEPTED.

Valid install type

Default directive setting
-OPT installType=″local″

Valid setting
You must set this directive to remote or local. Any other value fails validation.

Error identifier in temporaryPluginInstallLog.txt
INSTCONFFAILED : INVALID _ PLUGIN _ INSTALL _ TYPE _ SCENARIO _ SELECTED.

Valid application server installation location

Default directive setting
-OPT wasExistingLocation=″C:\Program Files\IBM\WebSphere\AppServer″

The setting varies per operating system.

Valid setting
If you set the -OPT installType directive to local, this validation checks that the path is a valid WebSphere Application Server Version 7 directory. Any other path fails validation for a local installation type.

Error identifier in temporaryPluginInstallLog.txt
INSTCONFFAILED : NON _ WAS7 _ DIRECTORY

Valid Web server configuration file 1

Default directive setting
-OPT webServerConfigFile1=″C:\Program Files\IBM\HTTPServer\conf\httpd.conf″

The setting varies per operating system. Valid file names for each Web server are:

- IBM HTTP Server: httpd.conf
- Apache: httpd.conf
- Domino: Notes.jar
- Sun One Web Server: obj.conf

Valid setting
Validation checks that the file exists. A known problem in ISMP validates a directory specification. However, you must identify the file to establish a working configuration.

Error identifier in temporaryPluginInstallLog.txt

INSTCONFFAILED : NON _ EXISTENT _ WS _ CONFIG _ FILE.

Valid Web server configuration file 2

Default directive setting

-OPT webServerConfigFile2=″″

Valid file names for affected Web servers are:

- Domino 7: names.nsf
- Domino 8: names.nsf
- Sun One Web Server: magnus.conf

Valid setting

Validation checks that the file exists. A known problem in ISMP validates a directory specification. However, you must identify the file to establish a working configuration.

Error identifier in temporaryPluginInstallLog.txt

INSTCONFFAILED : NON _ EXISTENT _ WS _ CONFIG _ FILE2.

Valid Web server definition name

Default directive setting

-OPT webServerDefinition=″webserver1″

Valid setting

Validation verifies that spaces do not exist in the Web server definition name. If spaces exist, validation fails.

Error identifier in temporaryPluginInstallLog.txt

INSTCONFFAILED : INVALID _ WEB _ SERVER _ DEFINITION _ NAME.

Verify that application mapping is true or false

Default directive setting

-OPT mapWebserverToApplications=″true″

Valid setting

If you set the directive to a value other than true or false, no error is generated or recorded in the temporaryPluginInstallLog.txt file. The Plug-ins installation wizard defaults the value to true and continues the installation.

Error identifier in temporaryPluginInstallLog.txt

None.

Valid Web server

Default directive setting

-OPT webServerSelected=″none″

Valid setting

You must specify the correct Web server for your operating system. Valid values include:

Error identifier in temporaryPluginInstallLog.txt

INSTCONFFAILED : INVALID _ WEBSERVER _ SELECTED.

Usage notes

- The file is not a read-only file.
- Edit this file directly with your flat file editor of choice.
- The file is updated when you specify the -options parameter when using the plug-ins installation wizard.
- The file must exist to perform a silent installation. The installation program reads this file to determine installation option values when you install silently. Provide the fully qualified file path.
- Save the file in a location that you can identify when you specify the fully qualified path as part of the installation command.

Editing Web server configuration files

Edit Web server configuration files to configure a Web server.

Before you begin

Use the IBM Web Administration for iSeries to configure i5/OS Web servers to communicate with a WebSphere Application Server. Doing so automatically updates the Web server configuration files as needed. See the *Installing your application serving environment* PDF for more information.

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

About this task

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

- See the *Installing your application serving environment* PDF for information about configuring your Web server.
- Configure IBM HTTP Server powered by Apache 2.x. See “Configuring IBM HTTP Server powered by Apache 2.x” on page 32.
- Configure IBM HTTP Server Version 6.x. See “Configuring IBM HTTP Server Version 6.x” on page 34.
- Configure IBM HTTP Server Version 7.0. See “Configuring IBM HTTP Server Version 7.0” on page 35.

Results

You can use the IBM Web Administration for iSeries to configure i5/OS 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

Note: If you are running IBM HTTP Server (powered by Apache) on i5/OS, you can use the manual configuration steps outlined below. However, it is recommended that you use the IBM Web Administration for i5/OS GUI.

Apache HTTP Server v2.0 is different from IBM HTTP Server (powered by Apache). Apache HTTP Server is not supported on i5/OS. For details on configuring IBM HTTP Server (Powered by Apache), see “Configuring IBM HTTP Server powered by Apache 2.x” on page 32.

When you install the Web server plug-ins for WebSphere Application Server on a non-iSeries system, the Plug-ins installation wizard configures the Web server.

This topic describes how to configure the Apache HTTP Server Version 2.0 Web server. Other procedures in “Editing Web server configuration files” describe configuring other supported Web servers.

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 Plug-ins installation wizard 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 stand-alone Application Server

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

Configure entries in thehttpd.conf file. It is recommended that you use the IBM Web Administration for i5/OS GUI to configure 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 /QSYS.LIB/QWAS6.LIB/QSVTAP20.SRVPGM
```

Local distributed example (Network Deployment only - the Web server is configured in a managed node:

```
WebSpherePluginConfig /QIBM/UserData/WebSphere/AppServer/V6/
   ND/profiles/profile1/config/cells/my_cell/nodes/
   my_managednode/servers/webserver1/plugin-cfg.xml
```

Local stand-alone example:

```
WebSpherePluginConfig /QIBM/UserData/WebSphere/AppServer/V6/
   ND/profiles/profile1/config/cells/my_cell/nodes/
   webserver1_node/servers/webserver1/plugin-cfg.xml
```

Remote example:

```
WebSpherePluginConfig /QIBM/UserData/WebSphere/AppServer/V6/
   ND/profiles/httpprofile1/config/webserver1/plugin-cfg.xml
```

Results

This procedure results in reconfiguring the Apache 2.0 Web server.

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 2.0 httpd.conf file, the IBM HTTP Server cannot start. You must comment or delete the directive to start the Version 2 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.

The Plug-ins installation wizard 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.

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 for WebSphere Application Server 7.0 using the UpdateInstaller.

Note: If you are running IBM HTTP Server (powered by Apache) on i5/OS, you can use the manual configuration steps outlined below. However, it is recommended that you use the IBM Web Administration for i5/OS GUI.

Apache HTTP Server v2.2 is different from IBM HTTP Server (powered by Apache). Apache HTTP Server is not supported on i5/OS.

When you install the Web server plug-ins for WebSphere Application Server on a non-iSeries system, the Plug-ins installation wizard configures the Web server.

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 27 describe configuring other supported Web servers.

About this task

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

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

Local file path means a file path to the plugin-cfg.xml file on an Application Server that is on the same machine as the Web server. Remote file path means the file path to the plugin-cfg.xml file when the Application Server is on a remote machine. The Plug-ins installation wizard 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 stand-alone Application Server

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

Configure entries in the httpd.conf file. It is recommended that you use the IBM Web Administration for i5/OS GUI to configure 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 /QSYS.LIB/QWAS6.LIB/QSVTAP20.SRVPGM
```

Local distributed example (Network Deployment only - the Web server is configured in a managed node:

```
WebSpherePluginConfig /QIBM/UserData/WebSphere/AppServer/V6/
   ND/profiles/profile1/config/cells/my_cell/nodes/
   my_managednode/servers/webserver1/plugin-cfg.xml
```

Local stand-alone example:

```
WebSpherePluginConfig /QIBM/UserData/WebSphere/AppServer/V6/
   ND/profiles/profile1/config/cells/my_cell/nodes/
   webserver1_node/servers/webserver1/plugin-cfg.xml
```

Remote example:

```
WebSpherePluginConfig /QIBM/UserData/WebSphere/AppServer/V6/
   ND/profiles/httpprofile1/config/webserver1/plugin-cfg.xml
```

Results

The Apache 2.2 Web server is reconfigured.

What to do next

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

The Plug-ins installation wizard 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.

Configuring Lotus Domino

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

Before you begin

When you install the Web server plug-ins for WebSphere Application Server on a non-iSeries system, the Plug-ins installation wizard configures the Web server.

If you are running Domino Web Server on i5/OS, use the IBM Web Administration for iSeries GUI.

Other procedures in “Editing Web server configuration files” on page 27 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 plugin as root. Domino can only be installed as root, and the configuration files belong to root.

Note: If you install the plugin as local and the WebSphere Application Server as non-root, 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.

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

    The DSAPI filter location on i5/OS is /QSYS.LIB/QWAS7.LIB/LIBDOMINOH.SRVPGM. The plug-in is installed in the bin directory of the Web server plug-ins for WebSphere Application Server.

If specifications for filter files for the Domino Server Application Programming Interface (DSAPI) already exist, use a space to delimit the Web server plug-in for WebSphere Application Server.

11. Click **Save and Close** in the upper-left of the center window.
12. Define the location of the plugin-cfg.xml configuration file.

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

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

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

    Setting the path to the plug-in configuration file

    Edit your notes.ini file.

    a. From a CL command prompt, enter the Work with Domino Servers (WRKDOMSVR) command.
    b. Type **13** next to the applicable Domino server, then press **Enter**.
    c. Add or edit the WebSphereInit property. (See the following table.)

       To add a new line, type ″**i**″ next to the desired insertion line, then press **Enter**.
    d. Press **F3** to save and exit.

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

    During the installation process, the Plug-ins installation wizard creates the setupPluginCfg.sh file 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 reconfiguring 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 wizard configured originally. If you are reconfiguring the Web server to change the original file path, do not use this script.
13. Restart the Domino server. When the server starts, information similar to 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 on configuring Lotus Domino to work with WebSphere Application Server, search the Lotus Support Services Web site at http://www-3.ibm.com/software/lotus/support/. Enter the search term WebSphere in the keyword search field.

Lotus Domino file locations and troubleshooting tips

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

Lotus Domino Server file locations

Lotus Domino server file locations are:

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

  Perform the following steps to determine the directory name:

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

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

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

Sample error message:

```
01/05/2006 13:28:56 HTTP Server:
Failed to load DSAPI module
/QSYS.LIB/QWAS7.LIB/LIBDOMINOH.SRVPGM
```

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

   ```
   WRKSPLF QNOTES
   ```

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

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

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

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

Configuring IBM HTTP Server powered by Apache 2.x

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

Before you begin

If you are running IBM HTTP Server (powered by Apache) on i5/OS, you can use the manual configuration steps outlined below with the IBM Web Administration for i5/OS.

When you install the Web server plug-ins for the product on a non-iSeries system, the Plug-ins installation wizard configures the Web server.

This topic describes how to configure the IBM HTTP Server. Other procedures in “Editing Web server configuration files” on page 27 describe configuring other supported Web servers.

About this task

Perform the step that configures IBM HTTP Server version 2.x for your operating system. IBM HTTP Server powered by Apache 2.0 is supported on i5/OS V5R4. IBM HTTP Server powered by Apache 2.2 is supported on i5/OS V6R1. On i5/OS V6R1, the Web server plug-ins are included with the 5761-DG1 product.

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 Plug-ins installation wizard 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 Web server definition name in the following examples is webserver1.

Configure entries in the httpd.conf file.
Use the IBM Web Administration for i5/OS GUI to configure the httpd.conf file.
Use the following examples of the LoadModule and the WebSpherePluginConfig directives as models for configuring your file for i5/OS V5R4:

```
LoadModule was_ap20_module
   /QSYS.LIB/QWAS7.LIB/QSVTAP20.SRVPGM
```

Use the following examples of the LoadModule and the WebSpherePluginConfig directives as models for configuring your file for i5/OS V6R1:

```
LoadModule was_ap20_module
   /QSYS.LIB/QHTTPSVR.LIB/QSVT2070.SRVPGM
```

Local distributed example (Web server is configured in a managed node):

```
WebSpherePluginConfig
   profile_root/config/cells/my_cell/
   nodes/my_managednode/servers/webserver1/plugin-cfg.xml
```

Example:

```
WebSpherePluginConfig
   profile_root/
   config/webserver1/plugin-cfg.xml
```

Results

This procedure results in editing and reconfiguring 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.

The Plug-ins installation wizard 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.

Configuring IBM HTTP Server Version 6.x

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

Before you begin

IBM HTTP Server Version 6.x is not supported on i5/OS. See the *Installing your application serving environment* PDF for information on how to configure IBM HTTP Server powered by Apache 2.0.

When you install the Web server plug-ins for WebSphere Application Server on a non-iSeries system, the Plug-ins installation wizard configures the Web server.

See “Installing Web server plug-ins on i5/OS” on page 5.

This topic describes how to configure IBM HTTP Server, Version 6.x. Other procedures in “Editing Web server configuration files” on page 27 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 Plug-ins installation wizard 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 stand-alone Application Server

Results

This procedure results in editing and reconfiguring 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.

The Plug-ins installation wizard 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.

Configuring IBM HTTP Server Version 7.0

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

Before you begin

IBM HTTP Server Version 7.0 is not supported on i5/OS. See the *Installing your application serving environment* PDF for information on how to configure IBM HTTP Server powered by Apache 2.0, or IBM HTTP Server powered by Apache 2.0.

When you install the Web server plug-ins for WebSphere Application Server on a non-iSeries system, the Plug-ins installation wizard configures the Web server.

See "Installing Web server plug-ins on i5/OS" on page 5.

This topic describes how to configure IBM HTTP Server, Version 7.0. Other procedures in "Editing Web server configuration files" on page 27 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 Plug-ins installation wizard 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 stand-alone Application Server.

Results

This procedure results in editing and reconfiguring 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.

The Plug-ins installation wizard 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.

Uninstalling the Web server plug-ins for WebSphere Application Server

You can uninstall the Web server plug-ins for WebSphere Application Server using the uninstalller program.

Before you begin

On i5/OS, you cannot uninstall the Web server plug-ins without installing the WebSphere Application Server product. Use the **removeOs400WebServerDefinition** command to delete a Web server definition. For example, to remove the definition for the Web server WEBSERVER1, which is associated with a profile called myHttpProfile, run this command from Qshell:

```
app_server_root/bin/removeOs400WebserverDefinition
   -webserver.name WEBSERVER1
   -profileName myHttpProfile
```

If you used the IBM Key Management wizard to create SSL key files in the Web server Plug-ins home directory, back up the files to a directory outside of the Web server Plugins directory. After the uninstall procedure is complete, you can delete the SSL key files if they are no longer required.

About this task

To uninstall the Web server plug-ins, run the uninstaller program. The program deletes the installation root directory for the Web server plug-ins for WebSphere Application Server, which removes the binary plug-ins.

1. Stop the Web server to allow the uninstaller program to change the Web server configuration.
2. Open a command window.
3. Change directories to the *plugins_root*/uninstall directory and issue the uninstall -silent command. **OR:** Change directories to the *plugins_root*/bin directory and issue the uninstall command.
4. Wait until the uninstall completes.

   Uninstalling the Web server plug-ins for WebSphere Application Server removes many files in the installation root directory, but leaves the following logs in the *plugins_root*/log/uninstall directory:

   - ISMP uninstall log: log.txt
   - Configuration uninstall log: masterConfigurationLog.txt
   - Web server deconfiguration log: uninstall*web_server_name*Plugin.log, such as the uninstallApachePlugin.log.
5. Delete files from the system temporary directory.

   Delete the following files:

   - Install temporary log : temporaryPluginInstallLog.txt
   - Uninstall temporary log : temporaryPluginUnInstallLog.txt
6. Delete the Web server definition in a standalone application server.

   The uninstaller program for the Web server plug-ins for WebSphere Application Server does not delete Web server definitions. However, you can delete a Web server definition from admin console OR by using the following wsadmin commands:

   ```
   $AdminTask deleteServer { -serverName webserver1 -nodeName WebserverHostName-node_node }
   $AdminTask removeUnmanagedNode { -nodeName WebserverHostName-node_node }
   $AdminConfig save
   ```

Results

After you exit from the *plugins_root*/uninstPlugin directory, the directory is removed.

The only remaining directory is the *plugins_root*/logs directory. The logs directory contains the install process log, the uninstall process log, and the Web server creation logs.

Note: The only other files that might exist are the SSL key files that you can create using the IBM Key Management Wizard. You can move these files to a safe location before using the manual uninstalling procedure, if necessary.

What to do next

To reinstall the Web server plug-ins for WebSphere Application Server, launch the installation procedure again.

To reinstall the Web server plug-ins for WebSphere Application Server into the original directory, delete the existing installation root directory for the plug-ins before reinstalling.

The default location is shown in “Directory conventions,” on page 161.

Use any of the installation procedures that are described in “Installing Web server plug-ins on i5/OS” on page 5.

See “Manually uninstalling Web server plug-ins for WebSphere Application Server” for information about manually uninstalling Web server plug-ins.

Manually uninstalling Web server plug-ins for WebSphere Application Server

You can uninstall Web server plug-ins for WebSphere Application Server manually, instead of using the uninstaller program.

Before you begin

Try uninstalling Web server plug-ins for WebSphere Application Server using the uninstaller program. Use this manual procedure if the uninstaller is not available for some reason.

Note: If you are running WebSphere Application Server on a i5/OS platform, see “Uninstalling the Web server plug-ins for WebSphere Application Server” on page 36.

About this task

To manually uninstall the Web server plug-ins, delete the installation root directory for the Web server plug-ins for WebSphere Application Server. Deleting the directory removes the binary plug-ins.

1. Delete the installation root directory for Web server plug-ins for WebSphere Application Server.
2. Delete the Web server definition in the application server configuration.

   You can delete a Web server definition from the admin console OR by using the following wsadmin commands:

   ```
   $AdminTask deleteServer { -serverName   webserver1 -nodeName WebserverHostName-node_node }
   $AdminTask removeUnmanagedNode { -nodeName WebserverHostName-node_node }
   $AdminConfig save
   ```
3. Reconfigure any Web servers that were configured to use the binary plug-ins that you deleted.

   See “Editing Web server configuration files” on page 27.

Results

Deleting the installation root directory for the Web server plug-ins for WebSphere Application Server removes the binary plug-ins. Any Web servers that are configured to use the deleted binary modules do not work.

What to do next

After uninstalling the Web server plug-ins for WebSphere Application Server, you can reinstall the plug-ins. Reinstalling the Web server plug-ins for WebSphere Application Server and reconfiguring the Web servers restores their functionality.

Use any of the installation procedures that are described in “Installing Web server plug-ins on i5/OS” on page 5.

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 Version 6 WebSphere Application Server product, a Web server, and the Web server plug-ins for WebSphere Application Server.

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

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

About this task

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

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

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

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

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

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

   Start a Qshell session and run the following command:

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

   Then issue the following command to stop the application server:

```
stopServer -profileName myProfile server1
```

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

```
Server server1 stop completed.
```

To start the application server, issue the following command:

```
startServer server1
```

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

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

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

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

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

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

The *port* variable is your HTTP server port.

Results

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

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.

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 set 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. Any server for which the host name can not be resolved is marked **unavailable** for the life of the configuration. No attempts to resolve the host name are made later on 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. When **false** is specified, DNS failures cause the Web server not to start.

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.

You can change the name of the plug-in configuration file. However, if you do change the file name, you must also change the Web server configuration to point to the new plug-in configuration file.

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

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

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

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

Data type	String
Default	plugin-cfg.xml

Automatically generate plug-in configuration file

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

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

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

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

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

Automatically propagate plug-in configuration file

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

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

By default, this field is checked.

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

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

Plug-in key store file name

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

- Click **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.

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.

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.

Note: 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 20 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 IHSDisableNagle element in the plugin-cfg.xml file. It only appears 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 appears 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, 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.

By default, this field is not checked. Select this field to enable users to include content in POST, PUT, GET, and HEAD requests when a Content-Length or Transfer-encoding header is contained in the request header.

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 Properties**.

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 not 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 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 LoadBalanceWeight 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 ServerWaitforContinue 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 serverprocess the request.

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

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

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

Remove special headers

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

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

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

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

Clone separator change

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

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

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

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

Web server plug-in configuration service property

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

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

Enable automated Web server configuration processing

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

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

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

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

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

Application Server property settings for a Web server plug-in

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

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

Server Role

Specifies the role this application server is assigned.

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

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

Default setting	Primary

Connection timeout

Specifies the connection timeout settings.

This setting specifies whether or not there is a limited amount of time the application server will maintain a connection with the Web server. Check the **Use connection timeout** checkbox to set a connection timeout. If no timeout is selected, the plug-in performs nonblocking connections with the application server.

If the checkbox is checked, you must specify a value in the seconds field. Specify a value of 0, if you want the plug-in to perform a blocking connection. Specify a value greater than 0, if you want the plug-in to wait the specified number for seconds to perform a successful connection. If a connection does not occur after that time interval, the plug-in marks the server unavailable and sends the request to another application server defined in the cluster.

This property 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 whether or not the port is available. If no value is specified for this property, the plug-in performs a blocking connect in which the plug-in sits until an operating system times out (which could be 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 another application server defined for the requested application.

Data type	Integer
Default	5

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.

Check the **Use read/write timeout** checkbox to set a read/write timeout. If the checkbox is checked, 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.

This field is ignored for a plug-in running on a Solaris platform.

Data type	Integer
Default	60 seconds

Maximum number of connections that can be handled by the Application Server

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.

Check the **Use maximum number of connections** checkbox to set a maximum number of connections. If the checkbox is checked 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 attribute 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 will use an extended handshake to check whether or not the Application Server is running.

This field corresponds to the ServerExtendedHandshake 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 other application servers.

If the plug-in performs some handshaking with the application server to ensure that it is started before it sends a request it can failover to another application server if it detects that the application server with which it is attempting to perform a handshake is down.

By default, this field is not checked. 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

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

When selected, the Web server plug-in will send the header ″100 Continue″ to the application server before it sends the request content.

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

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.

Table 4. Web server plug-in configuration properties

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

Table 4. Web server plug-in configuration properties (continued)

In the administrative console, click **Servers** > **Server Types** > **Web servers** > *web_server_name* > **Plug-in properties**	KeyringLocation	Keyring
In the administrative console, click **Servers** > **Server Types** > **Web servers** > *web_server_name* > **Plug-in properties**	StashfileLocation	Stashfile
In the administrative console, click **Servers** > **Server Types** > **Web servers** > *web_server_name* > **Plug-in properties** > **Custom properties** > **New**	FIPSEnable	FIPSEnable
In the administrative console, click **Servers** > **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
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

Table 4. Web server plug-in configuration properties (continued)

In the administrative console, click **Servers > Server Types > Web servers >** *web_server_name* **> plug-in properties > Custom properties > New**	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
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

Table 4. Web server plug-in configuration properties (continued)

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

Web server plug-in connections

The WebSphere Application Server 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.

Note: 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 third-party 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 application's call to getRemoteUser() 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 application's call to getRemoteUser() returns a null value, or if the correct remote user information is not being added to the Web server plug-in's data structure, make sure the remote user parameter within the WebAgent is still set to **YES**. (Sometimes this parameter gets set to **NO** when service is applied.)

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 Web site 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.

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.

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

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

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

Results

The version is shown in the ″Server version″ field and will look something like the following:

```
Server version: Apache/2.0.47 Server built:   Nov 26 2005 15:57:01
```

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 %was_profile_home%/config/cells directory.

About this task

You must update the global Web server plug-in configuration file whenever you:

- 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:

1. Click **Environment > Update global Web server plug-in configuration**.
2. Click **OK** to update the plugin-cfg.xml file.
3.
4. 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.

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.

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. Use the following table to assist you in selecting the correct GSKIT installation image file.

Operating system	GSKit 7 Installation image file
Microsoft Windows	No image name
AIX	gskta.rte
HP-UX	gsk7bas
Solaris Operating Environment	gsk7bas
Linux	gsk7bas_7.0.3.1.i386.rpm
Linux390	gsk7bas-7.0.3.1.s390.rpm
LinuxPPC	gsk7bas-7.0.3.1.ppc.rpm

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

- IBM HTTP Server documentation at http://www-3.ibm.com/software/webservers/httpservers/library.html
- 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

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.

Balancing workloads

When this maximum number of connections is reached, the plug-in, when establishing connections, automatically skips that application server, and tries the next available application server. If no application servers are available, an HTTP 503 response code will be returned to the client. This code indicates that the server is currently unable to handle the request because it is experiencing a temporary overloading or because maintenance is being performed.

Limiting the number of connections that can be established with an application server works best for Web servers that follow use a single, multithreaded process for serving requests.

```
ServerLimit          1
ThreadLimit          1024
StartServers         1
MaxClients           1024
MinSpareThreads      1
MaxSpareThreads      1024
ThreadsPerChild      1024
MaxRequestsPerChild  0
```

Improving performance in a high stress environment

To tune the TcpTimedWaitDelay setting, change the value of the tcp_time_wait_interval parameter from the default value of 240 seconds, to 30 seconds:

1. Locate in the Windows Registry:

   ```
   HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\tcpip\Parameters\TcpTimedWaitDelay
   ```

   If this entry does not exist in your Windows Registry, create it by editing this entry as a new DWORD item.
2. Specify, in seconds, a value between 30 and 300 inclusive for this entry. (It is recommended that you specify a value of 30.)

To tune the MaxUserPort setting:

1. Locate in the Windows Registry:

   ```
   HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\tcpip\Parameters\MaxUserPort
   ```

   If this entry does not exist in your Windows Registry, create it by editing this entry as a new DWORD item.
2. Set the maximum number of ports to a value between 5000 and 65534 ports, inclusive. (It is recommended that you specify a value of 65534,)

See the Microsoft Web site for more information about these settings.

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.

plugin-cfg.xml file

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

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

Config (required)

This element starts the WebSphere HTTP plug-in configuration file. It can include one or more of the following elements and attributes.

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 ServerCluster is able to resolve the host name. Any server for which the host name can not be resolved is marked *unavailable* for the life of the configuration. No attempts to resolve the host name are made later on 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

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 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.

ASDisableNagle

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

The value can be true or false.

IISDisableNagle

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

The value can be true or false.

AppServerPortPreference

This attribute is used to specify which port number the Application Server should use to build URI's for a sendRedirect. 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 is hostHeader.

ResponseChunkSize

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.

The ResponseChunkSize attribute lets you specify the maximum chunk size to use when reading the response body. For example, Config ResponseChunkSize=″N″>, where N equals the chunk size in kilobytes.

If the content length of the response body is unknown, a buffer size of N kilobytes is allocated and the body is read in N kilobyte 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 64k.

AcceptAllContent

Specifies whether or not 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 to be expected and read for all requests
- False if content only is only to be expected and read for POST and PUT requests.

False is the default.

ChunkedResponse

Specifies whether the plug-in should chunk the response to the client when a Transfer-Encoding : Chunked response header is present in the response.

This attribute only applies to the IIS, IPlanet, and Domino Web servers. 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.

false is the default.

IISPluginPriority

Specifies the priority in which the IIS Web server loads the WebSphere Web server plug-in. You can specify one of the following values for this attribute:

- High
- Medium
- Low

The default value is High.

NOTES:

- The IIS Web server uses this value during startup. Therefore, the Web server must be restarted before this change will take effect.
- 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/extensions. If problems occur while using a priority of Medium or Low, you will have to rearrange the order or change the priority of the interfering filter/extension.

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.

For example, you might specify the following:

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

Name (exactly one attribute for each Log)

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

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

LogLevel (zero or one attribute for each Log)

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 LogLevel is not specified for the Log element, 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 to fill up very quickly. A Trace setting should never be used in a normally functioning environment as it adversely affects performance.

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

Used to enable or disable the Edge Side Include (ESI) processor. If the ESI processor is disabled, the other ESI elements in this file are ignored.

Value can be set to true or false. By default, the ESI processor is enabled (set to true).

Property Name=″esiMaxCacheSize″ Value=″*interger*″

An integer specifying, 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.

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

Used to indicate whether or not the ESI processor should receive invalidations from the Application Server.

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

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

Used to indicate whether or not the Federal Information Processing Standard (FIPS) is enabled for making secure (SSL) connections to the Application Server. This property should be set to true, if FIPS is enabled on the Application Server..

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

ServerCluster (one or more elements for each Config)

A group of servers that are generally configured to service the same types of requests.

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 will load balance across the defined servers using either a Round Robin or a Random algorithm. The default is Round Robin.

Following 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>
```

Name (exactly one attribute for each ServerCluster)

The logical or administrative name to be used for this group of servers.

LoadBalance (zero or one attribute for each ServerCluster)

The default load balancing type is Round Robin.

The Round Robin implementation has a random starting point. The first server will be picked randomly. Round Robin will be used to pick 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.

IgnoreAffinityRequests (zero or one attribute for each ServerCluster)

Specifies whether the plug-in ignores the number of affinity requests made to a server when selecting servers based on the Round Robin algorithm. The value can be 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 (zero or one attribute for each ServerCluster)

An integer specifying the length of time that should elapse from the time that a server is marked down to the time that the plug-in will retry a connection. The default is 60 seconds.

RemoveSpecialHeaders (zero or one attribute for each ServerCluster)

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. By default the plug-in will remove these headers from incoming requests before adding the headers it is supposed to add.

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 (zero or one attribute for each ServerCluster)

Some pervasive devices cannot handle the colon character (:) used to separate clone IDs in conjunction with session affinity. This attribute for the server group tells the plug-in to expect the plus character (+) as the clone separator. You must change application server configurations so that an application server separates clone IDs with the plus character as well.

The value can be true or false.

PostSizeLimit (zero or one attribute for each ServerCluster)

The maximum number of bytes of request content allowed in order 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 bytes, which indicates that there is no limit for the post size.

Server (one or more elements for each ServerCluster)

A WebSphere Application Server instance that is configured to handle requests routed to it given the routing rules of the plug-in configuration. The Server should correspond to an application server running on either the local machine or a remote machine.

Name (exactly one attribute for each Server)

The administrative or logical name for the server.

WaitForContinue (zero or one attribute for each Server)

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.

This property will be ignored for POST requests in order to prevent a failure from occurring if the Application server closes a connection because of a keep alive time-out.

Enable this function (set to true) when configuring the plug-in to work with certain types of proxy firewalls.

LoadBalanceWeight (zero or one attribute for each Server)

Specifies the weight associated with this server when the plug-in does weighted Round Robin load balancing. The starting value for a server can be any integer between 0 and 20. However, zero should be specified only for a server that is shut down.

The LoadBalanceWeight for each server is decremented for each request 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 starts over.

When a server is shut down, it is recommended that you 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 (zero or one attribute for each Server)

The ConnectTimeout attribute of a Server element 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.

If no ConnectTimeout value is specified, the plug-in performs a blocking connect in which the plug-in sits until an operating system times out (as long as two 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 (zero or one attribute for each Server)

The ExtendedHandshake attribute 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.

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 other application servers.

The plug-in performs some handshaking with the application server to ensure that it is started before sending the request. This enables the plug-in to failover in the event the application server is down.

The value can be true or false.

MaxConnections (one element for each Server)

The MaxConnections attribute is used to specify the maximum number of pending connections to an Application Server that can be flowing through a Web server process at any point in time.

For example, assuming that:

- The application server is fronted by 5 nodes that are running an IBM HTTP Server.
- Each node starts 2 processes.
- The MaxConnections attribute is set to 50.

In this example, the application server could potentially get up to 500 connections. (You take the number of nodes, 5, multiply it 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 (one or more elements for each Server)

The transport for reading and writing requests to a particular WebSphere application server instance. The transport provides the information needed to determine the location of the application server to which the request will be sent. If the Server has multiple transports defined to use the same protocol, the first one will be used.

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 will be performed to determine the appropriate transport to use to send the request to the application server.

Hostname (exactly one attribute for each Transport)

The host name or IP address of the machine on which the WebSphere application server instance is running.

Port (exactly one attribute for each Transport)

The port on which the WebSphere application server instance is listening.

Protocol (exactly one attribute for each Transport)

The protocol to use when communicating over this transport -- either HTTP or HTTPS.

Property (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:

```
<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"/>
```

Note: The default password for viewing the plugin-key.kdb file using iKeyMan is WebAS.

Name (exactly one attribute for each Property)
The name of the Property being defined. Supported names recognized by the transport are keyring, stashfile, and password.

Note: password is the only name that can be specified for the WebSphere HTTP Plug-in for z/OS. keyring, and stashfile, if specified, will be ignored.

Value (exactly one attribute for each Property)
The value of the Property being defined.

ServerIOTimeout
The ServerIOTimeout attribute of a server element enables the plug-in to set a time out value, in seconds, for sending requests to and reading responses from the application server. 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, if you specify:

```
<Server Name="server1" ServerIOTimeout=300>
```

In this case, 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 time out 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 a couple of minutes for an application server to process a request. Setting the value of the ServerIOTimeout attribute too low could cause the plug-in to send a false server error response to the client. The default value is `60` seconds.

ClusterAddress (zero or one element for each ServerCluster)
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 only define 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.

PrimaryServers (zero or one element for each server cluster)

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.

BackupServers (zero or one element for each server cluster)

Specifies a list of servers to which requests should be sent to if all servers 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 received from an application server.

VirtualHostGroup

A group of virtual host names that will be specified in the HTTP Host header. Enables you to group virtual host definitions together that are configured to handle similar types of requests.

Following is an example of a VirtualHost Group 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 (exactly one attribute for each VirtualHostGroup)

The logical or administrative name to be used for this group of virtual hosts.

VirtualHost (one or more elements for each VirtualHostGroup)

The name used for a virtual or real machine used to determine if incoming requests should be handled by WebSphere Application Server or not. Use this element to specify host names that will be in the HTTP Host header which should be seen for requests that need to be handled by the application server. You can specify specific host names and ports that incoming requests will have or specify an asterisk (*) for either the host name, port, or both.

Name (exactly one attribute for each VirtualHost)

The actual name that should be specified in the HTTP Host header in order to match successfully with this 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 what those combinations are.

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 will match this rule. If no port is specified in the definition the default HTTP port of 80 is assumed.

UriGroup

A group of URIs that will be specified on the HTTP request line. The same application server must be able to handle the URIs. The route will compare the incoming URI with the URIs in the group to determine if the application server will handle the request.

Following is an example of a UriGroup element and associated elements and attributes:

```
<UriGroup Name="Uris">
<Uri Name="/servlet/snoop"/>
<Uri Name="/webapp/*"/>
<Uri Name="*.jsp"/>
</UriGroup>
```

Name (exactly one attribute for each UriGroup)

The logical or administrative name for this group of URIs.

Uri (one or more elements for each UriGroup)

The virtual path to the resource that will be serviced by WebSphere Application Server. Each URI specifies the incoming URLs that need to be handled by the application server. You can use a wildcard in these definitions.

Name (exactly one attribute for each Uri)

The actual string that should be specified in the HTTP request line in order 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 a URI having <Uri Name=″*Web_application_URI*/servlet/*″> is generated.

AffinityCookie (zero or one attribute for each Uri)

The name of the cookie the plug-in should use when trying to determine if the inbound request has session affinity. The default value is **JSESSIONID**.

AffinityURLIdentifier (zero or one attribute for each Uri)

The name of the identifier the plug-in should use when trying to determine if the inbound request has affinity specified in the URL to a particular clone. The default value is **jsessionid**.

Route A request routing rule by which the plug-in will determine if an incoming request should be handled by a WebSphere application server.

The route definition is the central element of the plug-in configuration. It specifies how the plug-in will handle 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 should be sent on to the ServerCluster defined in this route.

Following is an example of this element:

```
<Route VirtualHostGroup="Hosts" UriGroup="Uris" ServerCluster="servers/>
```

VirtualHostGroup (zero or one attribute for each Route)

The group of virtual hosts that should be used in route determination. The incoming host header and server port are matched to determine if this request should be handled by the application server.

It is possible to omit this from the route definition. If it is not present then every request will match during the virtual host match portion of route determination.

UriGroup (zero or one attribute for each Route)

The group of URIs to use for determining the route. The incoming URI for the request is matched to the defined URIs in this group to determine if this request should be handled by the application server.

It is possible to omit this from the route definition. If it is not present than every request will match during the URI match portion of route determination.

ServerCluster (exactly one attribute for each Route)

The cluster to which to send request that successfully match the route.

The cluster that should be 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 defined within this cluster.

RequestMetrics

This element is used to determine if request metrics is enabled, and how to filter the requests based on the Internet protocol (IP) and Uniform Resource Identifiers (URI) filters when request metrics is enabled.

Following is an example of this element:

```
<RequestMetrics armEnabled="false"  loggingEnabled="true"
   rmEnabled="false" traceLevel="PERF_DEBUG">
```

armEnabled (zero or one attribute for RequestMetrics)
This attribute indicates whether the ARM 4 agent is enabled in the plug-in. When it is set to true, the ARM 4 agent will be 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 will never be called.

loggingEnabled (exactly one attribute for RequestMetrics)
This attribute indicates 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.

rmEnabled (exactly one attribute for RequestMetrics)
This attribute indicates whether or not the request metrics is enabled in the plug-in. When it is set to true, the plug-in request metrics will look at the filters and log 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 will be ignored..

traceLevel (exactly one attribute for RequestMetrics)
When rmEnabled is true, this attribute indicates how much information is logged. 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.

filters (zero, one, or two attributes for RequestMetrics)
When rmEnabled is true, the filters control which requests are traced.

enable (exactly one attribute for each filter)
When enable is true, the type of filter is on and requests must pass the filter.

type (exactly one attribute for each filter)
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.

filterValues (one or multiple attribute for each filter)
The filterValues show the detailed filter information.

value (exactly one attribute for each filterValue)
Specifies the filter value for the corresponding filter type. This could be either a client IP address or a URI.

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 spawned 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`.

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.

Note: 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*.

1. Install your WebSphere Application Server product.
2. Install IBM HTTP Server or another supported Web server.
3. Install the binary plug-in module using the Plug-ins installation wizard.
4. 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 entry** wizard. You will create the new Web server definition using this wizard. The wizard 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.

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.**

5. **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 Setting permissions manually for instructions.

6. **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.

7. Start IBM HTTP Server. Refer to Starting the IBM HTTP 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 configuration file.
- Viewing the Web server logs.
- Propagation of the Web server `plugin-cfg.xml` file.

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 an base or Express WebSphere Application Server 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 a WebSphere

Application Server – Express, base, 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`

Special consideration for IBM HTTP Server for z/OS, powered by Apache: To support remote administration and configuration of IBM HTTP Server for z/OS, a Web server type IHSZOS must be defined in the WebSphere Application Server repository.

Note: During profile creation, using the z/OS Profile Management Tool, if you select **Advanced profile creation** and **Create a Web server definition**, you will not be permitted to select the combination of Web server type `IBM HTTP Server` and Web server operating system of `z/OS`. However, you can create the Web server type for IBM HTTP Server on z/OS through the administrative console wizard, `createWebServerDefintion.jacl`, or the wsadmin command after the profile create, using the z/OS Profile Management Tool.

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.

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 collection

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

Web servers

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

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

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

Name

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

Web server type

Indicates the type of Web Server you are using.

Node

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

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

Version

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

Status

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

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

➡	**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.

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.

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 - Express configuration, click **Servers > Server Types > Web Servers > New** to create a Web server.

Web servers

Port

Indicates the port to access the administration server (default is 8008).

The HTTP administration server port is 2001 on i5/OS.

Use SSL

Specifies if the port is secure.

User ID

Specifies a user ID in the `<install_dir>/conf/admin.passwd` file. Create this with the `htpasswd` script file, located in the `<install_dir>/bin` directory.

Password

Specifies a password in the `<install_dir>/conf/admin.passwd` file. Create this with the `htpasswd` script file, located in the `<install_dir>/bin` directory.

On i5/OS, there is no htpasswd script in the `<install_dir>/bin` directory. The HTTP administrator is typically created as an i5/OS SECOFR profile.

Web server configuration file

Use this page to view or modify the contents of the Web server configuration file in your Web browser.

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.

Chapter 3. Using Session Initiation Protocol to provide multimedia and interactive services

Follow these procedures for creating SIP applications and configuring the SIP container.

About this task

Session Initiation Protocol (SIP) is used to establish, modify, and terminate multimedia IP sessions including IP telephony, presence, and instant messaging. A *SIP application* in WebSphere is a Java program that uses at least one Session Initiation Protocol (SIP) servlet. A SIP servlet is a Java-based application component that is managed by a SIP servlet container.

The servlet container is a part of an application server that provides the network services over which requests and responses are received and sent. The servlet container decides which applications to invoke and in what order. A servlet container also contains and manages a servlet through its lifecycle.

This topic is divided into the following subsections:

- Configure the SIP container: Information and instructions for configuring SIP container properties and timers.
- Developing SIP applications: Reference information for developers.
- Deploying SIP applications: Information for installing, starting, and stopping, your applications.
- Securing SIP applications: Instructions for enabling security providers and setting up a trust association interceptor (TAI).
- Tracing a SIP container: Troubleshoot SIP applications through traces on the SIP container.

SIP in WebSphere Application Server

WebSphere Application Server delivers rich SIP functionality throughout its infrastructure.

Session Initiation Protocol (SIP) has grown considerably since it first became an IETF standard in 1999. SIP was originally intended purely for video and audio but has now grown to become the control protocol for many interactive services, particularly in the peer-to-peer realm. SIP, and the standards surrounding SIP, provide the mechanisms to look up, negotiate, and manage connections to peers on any network over any other protocol.

The SIP Servlet 1.0 specification allows enterprise applications to use SIP and to support SIP-predominant applications in the Java EE environment.

WebSphere Application Server also provides tooling for the development environment and high performing Edge Components to handle distributed application environments.

In the application server, the Web container and SIP container are converged and are able to share session management, security and other attributes. In this model, an application that includes SIP servlets, HTTP servlets, and portlets can seamlessly interact, regardless of the protocol.

High availability, offered by the Network Deployment (ND) version of the product, of the converged applications is made possible because of the tight integration of HTTP and SIP in the base application server.

In front of a clustered application sits the proxy server, managing the traffic and workload of the SIP and HTTP traffic to the container. This proxy server is a stateless SIP proxy and a HTTP reverse proxy together, which uses the unified clustering framework and high availability (HA) manager services of the

© Copyright IBM Corp. 2008

ND package to seamlessly monitor the health of the servers. The proxy server also can act as a stand-alone stateless SIP proxy in front of the SIP container in the application server when no HTTP traffic is present.

The proxy server uses the unified clustering framework and HA manager services of the ND package to perform failover work, when necessary. With the converged proxy and converged container, session failover is done with affinity to the application, allowing the HTTP and SIP sessions to be tied together automatically. Having the SIP and HTTP sessions automatically tied together from the container to the Proxy is another way the application server solution excels in converged environments.

It's important to note that the SIP function in the proxy server is stateless. The SIP RFC defines two types of proxy servers, one stateful and one stateless. Normally, a SIP proxy is a stateful instance and stateless proxies are specified as such. A stateful proxy participates in the call flows and is implemented using SIP servlets.

The stateless SIP proxy functionality in the proxy server allows the proxy to handle the workload, routing, and session affinity needs of the SIP container with less complexity. Being stateless, the proxy server can be fronted by a simple IP sprayer such as the load balancer component included in the ND package. If a proxy server fails, the affinity is to the container and not to the proxy itself so there is one less potential failure along the message flow.

SIP Infrastructure

The SIP infrastructure is a multi-tiered architecture made up of SIP containers, SIP proxies and an IP sprayer. The SIP container is a general purpose SIP application server. The SIP infrastructure consists of:

- SIP container – Web container extension that implements JSR 116 plus a SIP protocol stack that implements all pertinent RFCs.
- SIP proxy – Stateless edge device that handles I/O concentration, load balancing, and other functions, in a similar manner to the reverse HTTP proxy. This is not the same as the SIP proxy defined by RFC 3261.
- Load balancer – SIP enabled to interoperate with SIP proxies and SIP containers. The extendable SIP proxy handles session affinity, load balancing, and failover. The load balancer functions as a highly available IP sprayer to dispatch messages to the proxies.

SIP is a key element for many new applications, especially when converged with HTTP, including:

- Click-To-Call
- Voice over IP
- Third Party Call Control and Call Monitoring
- Presence and Instance Messaging

SIP applications

A *SIP application* is a Java program that uses at least one Session Initiation Protocol (SIP) servlet.

A SIP servlet is a Java-based application component that is managed by a SIP servlet container and that performs SIP signaling. Like other Java-based components, servlets are platform-independent Java classes that are compiled to platform-neutral bytecode that can be loaded dynamically into and run by a Java-enabled SIP application server. Containers, sometimes called servlet engines, are server extensions that handle servlet interactions. SIP servlets interact with clients by exchanging request and response messages through the servlet container.

SIP is used to establish, modify, and terminate multimedia IP sessions including IP telephony, presence, and instant messaging. ″Presence″ in this context refers to user status such as ″Active,″ ″Away,″ or ″Do not disturb.″ The standard that defines a programming model for writing SIP-based servlet applications is JSR 116.

SIP container

This product complies with the following SIP standards:

IETF

JCP

For a complete list of the supported Internet Engineering Task Force (IETF) and Java Community Process (JCP) industry standards, see the ″Compliance with industry SIP standards″ topic linked below.

SIP industry standards compliance

The product implementation of Session Initiation Protocol (SIP) complies with industry standards for both a SIP container and SIP applications.

SIP container

This product complies with the following SIP standards:

IETF

JCP

This product also complies with the Internet Engineering Task Force (IETF) and Java Community Process (JCP) industry standards for SIP. The following table contains a list of the IETF and JCP standards.

Table 5. WebSphere Application Server complies with these SIP standards

Standard	Description
RFC 2543	SIP: Session Initiation Protocol
RFC 3261	SIP: Session Initiation Protocol
RFC 3262	Reliability of provisional responses in SIP
RFC 3265	SIP-specific event notification
RFC 3326	The Reason Header field for the SIP
RFC 3515	The SIP Refer method
RFC 3824	Using E.164 numbers with the SIP
RFC 3903	SIP Extension for event state publication
RFC 3263	Locating SIP servers **Note:** SIP does not support use of DNS procedures for a server to send a response to a back-up client if the primary client fails.

SIP applications

This product complies with standards for SIP applications.

Table 6. Compliance with standards for SIP applications

Standard	Description
RFC 2848	The PINT Service Protocol: Extensions to SIP and Session Description Protocol (SDP) for internet protocol (IP) access to telephone call services
RFC 2976	The SIP INFO method
RFC 3050	Common gateway interface for SIP
RFC 3087	Control of service context using SIP request-URI
RFC 3264	An offer and answer model with SDP
RFC 3266	Support for IPv6 in SDP

Table 6. Compliance with standards for SIP applications (continued)

Standard	Description
RFC 3312	Integration of resource management and SIP
RFC 3313	Private SIP extensions for media authorization
RFC 3319	Dynamic Host Configuration Protocol (DHCPv6) options for SIP servers
RFC 3327	SIP Extension Header field for registering non-adjacent contacts
RFC 3372	SIP for telephones (SIP-T): context and architectures
RFC 3398	Integrated Services Digital Network (ISDN) User Part (ISUP) to SIP mapping
RFC 3428	SIP extension for instant messaging
RFC 3455	Private Header (P-Header) extensions to the SIP for the 3rd-Generation Partnership Project (3GPP)
RFC 3578	Mapping of Integrated Services Digital Network (ISDN) User Part (ISUP) overlap signaling to the SIP
RFC 3603	Private SIP proxy-to-proxy extensions for supporting the PacketCable distributed call signaling architecture
RFC 3608	SIP Extension Header field for service route discovery during registration
RFC 3665	SIP basic call flow examples
RFC 3666	SIP Public Switched Telephone Network (PSTN) call flows
RFC 3680	A SIP event package for registrations
RFC 3725	Best current practices for third-party call control (3pcc) in the SIP
RFC 3840	Indicating user agent capabilities in the SIP
RFC 3842	A message summary and message waiting indication event package for the SIP
RFC 3856	A presence event package for the SIP
RFC 3857	A watcher information event template package for the SIP
RFC 3959	The early session disposition type for the SIP
RFC 3960	Early media and ringing tone generation in the SIP
RFC 3976	Interworking SIP and intelligent network (IN) applications
RFC 4032	Update to the SIP preconditions framework
RFC 4092	Usage of the SDP Alternative Network Address Types (ANAT) semantics in the SIP
RFC 4117	Transcoding services invocation in the SIP using third-party call control (3pcc)
RFC 4235	An invite-initiated dialog event package for the SIP
RFC 4240	Basic network media services with SIP
RFC 4353	A framework for conferencing with the SIP
RFC 4354	A SIP event package and data format for various settings in support for the push-to-talk over cellular (PoC) service
RFC 4411	Extending the SIP Reason Header for preemption events
RFC 4457	The SIP P-user-database Private-Header (P-Header)
RFC 4458	SIP URIs for applications such as voicemail and interactive voice response (IVR)
RFC 4483	A mechanism for content indirection in SIP messages
RFC 4497	Interworking between the SIP and QSIG
RFC 4508	Conveying feature tags with the SIP REFER method

Runtime considerations for SIP application developers

You should consider certain product runtime behaviors when you are writing Session Initiation Protocol (SIP) applications.

Container may accept non-SIP URI schemes

The SIP container will not reject a message if it doesn't recognize the scheme in the request URI because the container cannot know which URI schemes are supported by the applications. SIP elements may support a request URI with a scheme other than `sip` or `sips`, for example, the `pres:` scheme has a particular meaning for presence servers, but the container does not recognize it. It is up to the application to determine whether to accept or to reject a specific scheme. SIP elements may translate non-SIP URIs using any mechanism available, resulting in SIP URIs, SIPS URIs, or other schemes, like the `tel` URI scheme of RFC 2806 [9].

Invoking session listener events

SipSessionListener and SipApplicationSessionListener events are invoked only if an application requests the corresponding session object. You do this by using in your application the method shown in Table 7.

Table 7. Methods that invoke session listener events

Event	Method
SipSessionListener	getSession()
SipApplicationSessionListener	getApplicationSession()

Session activation and passivation

During normal operation, this product never migrates a session from one server to another. Session migration occurs only as a result of a server failure. Therefore the SipSessionActivationListener method's passivation callback is never invoked. However, the activation callback is invoked when a failure forces session failover to a different server.

External resources

If a SIP application performs intensive I/O or accesses an external database, it may be blocked for several milliseconds. If possible, use asynchronous APIs for these resources . Under stress, a blocked SIP application may trigger a Request Timeout or re-transmission.

SIP application attributes

Avoid hanging large objects or BLOBs as SIP Session attributes (via SIPSession.setAttribute API). This may damage the overall performance when combined with high availability (HA). The same recommendation applies for SIPApplicationSession.setAttribute. In most cases, the large object can be replaced by several simple or composed strings.

SIP IBM Rational Application Developer for WebSphere framework

This page provides information about the SIP IBM Rational Application Developer for WebSphere framework.

WebSphere Application Server includes IBM Rational Application Developer for WebSphere to meet all the basic development needs for Java EE applications. Included in IBM Rational Application Developer for WebSphere is support for developing SIP servlet applications. IBM Rational Application Developer for WebSphere provides graphical deployment descriptor editors and basic wizards to get you started writing SIP servlets.

IBM Rational Application Developer for WebSphere also includes many other pieces that integrate well in WebSphere Application Server deployments, such as the Unit Test Environment, which provides the WebSphere Application Server servlet container to run SIP servlets in the development phase of the product, as well as tools for server automation and application packaging.

IBM Rational Application Developer for WebSphere supports:
- SIP servlet development (JSR 116)
- Converged SIP/HTTP applications
- Import/Export SAR packages
- SIP samples (call forward, call block, third party call)

SIP container

A *SIP container* is a Web application server component that invokes the Session Initiation Protocol (SIP) action servlet and that interacts with the action servlet to process SIP requests.

The servlet container provides the network services over which requests and responses are received and sent. It decides which applications to invoke and in what order. The container also contains and manages servlets through their life cycle.

A SIP servlet container manages the network listener points on which it listens for incoming SIP traffic. A listener point is a combination of transport protocol, IP address, and port number. The SIP servlet container supports the transport protocols UDP, TCP, and TLS over TCP.

Configuring the SIP container

Configure the Session Initiation Protocol (SIP) container to adjust message response times or set a custom property.

About this task

Use the administrative console to configure SIP container settings. Complete the following steps to find and configure the SIP container settings.
1. Start WebSphere Application Server.
2. From the administrative console, click **Servers** → **Application servers** → ***serverName***.
3. From **Container Settings**, expand **SIP Container Settings**, and click **SIP Container**. Select the container settings that you want to change.
   - From **General Properties**, you can configure session, message, and response time maximums. See *Session Initiation Protocol container settings* and *SIP container custom properties*.
   - From **Additional Properties**, you can define custom properties, manage transport chains or inbound channel settings, or configure the session manager.
4. After configuring the SIP container, click **Apply** to save the changes.
5. Restart WebSphere Application Server.

Results

Changes to the SIP container settings take effect after you restart WebSphere Application Server.

SIP container custom properties

You can add any of the following custom properties to the configuration settings for a Session Initiation Protocol (SIP) container.

To specify custom properties for a specific SIP container, navigate to the custom properties page, and then specify a value for the custom property.

Note: The custom properties are supported as the primary method of configuration. Therefore, if a custom property is set and then you set the corresponding setting in the administrative console, the custom property value is used.

1. In the administrative console, expand **Servers > Server Types > WebSphere application servers >** *server_name* to open the configuration tab for the server.
2. From **Container settings**, expand **SIP Container settings**, and click **SIP container**.
3. From **Additional properties**, select **Custom Properties** → **New**.
4. On the settings page, type the custom property to configure in the **Name** field, and then type the value of the custom property in the **Value** field.
5. Click **Apply** or **OK**.
6. Click **Save** on the console task bar to save your configuration changes.
7. Restart the server.

The following list of SIP container custom properties is provided with the product. These properties are not shown on the settings page for the container.

enable.system.headers.modify

Specifies whether the application has access to headers that are otherwise restricted.

Data type	String
Default	False

replicate.with.confirmed.dialog.only

Specifies whether to replicate the application session, even when no dialogs are confirmed. If the value is set to `false`, then the application session is replicated immediately after the session is created. Otherwise, the application session is only replicated when an associated dialog is confirmed.

Data type	String
Default	False

com.ibm.sip.sm.lnm.size

Specifies the number of logical names in the application server. Each SIP object that can be replicated, such as a SIP session, is associated with a logical name. All objects with the same logical name are replicated to the same back-up container. The proxy can route messages to the correct container using the logical name found in the message. The value must be greater than 1.

Data type	String
Default	10

auth.int.enable

Specifies the **auth-int** quality of protection (QOP) for digest authentication. Digest authentication defines two types of QOP: **auth** and **auth-int**. By default, **auth** is used. When this custom property is set to True, the highest level of protection is used, which is the **auth-int** QOP.

Data type	String
Default	False

DigestPasswordServerClass

Specifies the Java class name that implements the PasswordServer interface.

Data type	String
Default	LdapPasswordServer

com.ibm.websphere.sip.security.tai.usercachecleanperiod

Specifies the clean security subject cache period in minutes.

Data type	String
Default	15

com.ibm.ws.sip.tai.DisableSIPBasicAuth

Specifies whether to allow basic authentication for SIP.

Data type	String
Default	False

com.ibm.webspehere.sip.security.digest.ldap.cachecleanperiod

Specifies the clean Lightweight Directory Access Protocol (LDAP) cache period in minutes.

Data type	String
Default	120

pws_atr_name

Specifies the LDAP attribute name that stores the user password.

Data type	String
Default	userpassword

javax.sip.transaction.invite.auto100

Specifies whether to automatically reply to invite requests with a `100 Trying` response. Disabling this property might increase the number of invite retransmissions.

Data type	String
Default	True

Note: This custom property is deprecated. Do not use this custom property unless you are running in a mixed cell environment that includes at least one core group that contains a mixture of Version 7.0 and Version 6.x processes.

Note: If you are running in a mixed cell environment, and you have core groups that contain a mixture of Version 7.0 and Version 6.x processes, you must continue to use this custom property.

javax.sip.force.connection.reuse

Specifies whether to force reuse of inbound connections for outbound requests. This custom property is only relevant for stream transports, such as Transmission Control Protocol (TCP) and Transport Layer Security (TLS). Disabling this property causes the container to create a separate connection for outbound requests, even if an existing connection is already established to the same peer address. The connection is automatically reused if the top Via header in the inbound request contains an alias parameter. (http://www.ietf.org/internet-drafts/draft-ietf-sip-connect-reuse-07.txt)

Data type	String
Default	TrueFalse

Note: This custom property is deprecated. Do not use this custom property unless you are running in a mixed cell environment that includes at least one core group that contains a mixture of Version 7.0 and Version 6.x processes.

Note: If you are running in a mixed cell environment, and you have core groups that contain a mixture of Version 7.0 and Version 6.x processes, you must continue to use this custom property.

javax.sip.hide.message.body

Specifies to hide message content in logs. Set the value of this property to `true` to remove the message body text from SIP messages printed in the log files. This property only affects the representation of the messages in log files.

Data type	String
Default	False

Note: This custom property is deprecated. Do not use this custom property unless you are running in a mixed cell environment that includes at least one core group that contains a mixture of Version 7.0 and Version 6.x processes.

Note: If you are running in a mixed cell environment, and you have core groups that contain a mixture of Version 7.0 and Version 6.x processes, you must continue to use this custom property.

javax.sip.PATH_MTU

Specifies the maximum transmission unit, in bytes, for outbound User Datagram Protocol (UDP) requests. The SIP stack measures the size of a request before sending it out on the UDP channel. If the request is larger than the value specified for **PATH_MTU-200** (1300 bytes by default), then the transport is switched from UDP to TCP before transmission. Increase this value to send larger requests over the UDP channel; however, messages might be truncated or dropped. See the RFC 3261-18.1.1 specification for details.

Data type	String
Default	1500

Note: This custom property is deprecated. Do not use this custom property unless you are running in a mixed cell environment that includes at least one core group that contains a mixture of Version 7.0 and Version 6.x processes.

Note: If you are running in a mixed cell environment, and you have core groups that contain a mixture of Version 7.0 and Version 6.x processes, you must continue to use this custom property.

javax.sip.OUTBOUND_PROXY

Specifies the fixed address for routing all outbound SIP messages. The format is `address:port/transport`, such as `1.2.3.4:5065/tcp`.

Note: Do not use this property if the container is fronted by an application server SIP proxy.

Data type	String
Default	null

Note: This custom property is deprecated. Do not use this custom property unless you are running in a mixed cell environment that includes at least one core group that contains a mixture of Version 7.0 and Version 6.x processes.

Note: If you are running in a mixed cell environment, and you have core groups that contain a mixture of Version 7.0 and Version 6.x processes, you must continue to use this custom property.

javax.sip.trace.msg.in

Specifies whether to print incoming messages to a system.out file.

Data type	String
Default	False

javax.sip.trace.msg.out

Specifies whether to print outbound messages to a system.out file.

Data type	String
Default	False

javax.sip.stat.report.interval

Specifies the amount of time, in milliseconds, for reporting dispatch and timer statistics to a system.out file. A value of zero indicates no report.

Data type	String
Default	0

javax.sip.detect.pre.escaped.params

Specifies whether to prevent the container from re-escaping Uniform Resource Identifier (URI) parameters that were pre-escaped by the application.

Enabling this property provides the application with more control over escaping URI parameters, when calling the **javax.servlet.sip.SipFactory.createURI()** and the **javax.servlet.sip.SipURI.setParameter()** parameters.

By default, the container only escapes characters that it must encode according to the RFC 3261 25.1 specification. In some cases, however, escaping additional characters might be required. Due to a limitation in the JSR 116 (5.2.1) specification, the application cannot perform its own escaping. Because of this limitation, attempts by the application to encode URI parameters causes the container to re-encode the percent sign. If the value of this property is set to true, the container cannot re-encode the percent sign.

Setting the value to true is not in compliance with the JSR 116 (5.2.1) specification, but provides the application with greater control over URI parameter escaping. APAR PK37192 describes the problem and the workaround.

Data type	String
Default	False

DigestPasswordServerClass

Specifies types of user registries that are supported, except LDAP. To configure DigestTAI without the LDAP user registry, complete the following steps.

1. Create a class that implements this interface: com.ibm.ws.sip.security.digest.DigestPasswordServer 1
2. Add these properties to the SIpDigestTAI custom property.

   name =

   DigestPasswordServerClass value =
3. Ensure that all users declared by the **impl** class are declared in the user registry configured for the product security.

Data type	String

Default	impl

javax.sip.bind.retries

Specifies the amount of time, in milliseconds, between attempts to start the SIP channel if the SIP port is busy with another process during server startup.

Data type	String
Default	60

javax.sip.bind.retry.delay

Specifies the delay, in milliseconds, between attempts to start the SIP channel if the SIP port is busy with another process during server startup.

Data type	String
Default	5000

javax.sip.transaction.timer.t1

Specifies the amount of time, in milliseconds, for a network round trip delay for timer T1 for the RFC 3261 specification. The value is used as a base for calculating some timers and is relevant for all types of transactions, such as client, server, invite, and non-invite transactions.

Data type	String
Default	500

Note: This custom property is deprecated. Do not use this custom property unless you are running in a mixed cell environment that includes at least one core group that contains a mixture of Version 7.0 and Version 6.x processes.

Note: If you are running in a mixed cell environment, and you have core groups that contain a mixture of Version 7.0 and Version 6.x processes, you must continue to use this custom property.

javax.sip.transaction.timer.t2

Specifies the maximum time in milliseconds before retransmitting non-invite requests and invite responses for timer T2 for the RFC 3261 specification.

Data type	String
Default	4000

Note: This custom property is deprecated. Do not use this custom property unless you are running in a mixed cell environment that includes at least one core group that contains a mixture of Version 7.0 and Version 6.x processes.

Note: If you are running in a mixed cell environment, and you have core groups that contain a mixture of Version 7.0 and Version 6.x processes, you must continue to use this custom property.

javax.sip.transaction.timer.t4

Specifies the maximum amount of time, in milliseconds, for a message to remain in the network. This value is used as a base for calculating other timers for timer T4 for the RFC 3261 specification.

Data type	String
Default	5000

Note: This custom property is deprecated. Do not use this custom property unless you are running in a mixed cell environment that includes at least one core group that contains a mixture of Version 7.0 and Version 6.x processes.

Note: If you are running in a mixed cell environment, and you have core groups that contain a mixture of Version 7.0 and Version 6.x processes, you must continue to use this custom property.

javax.sip.transaction.timer.a

Specifies, for UDP only, the amount of time, in milliseconds, before retransmitting invite requests for timer A for the RFC 3261 specification. This property is relevant for the invite client transaction.

Data type	String
Default	javax.sip.transaction.timer.t1

Note: This custom property is deprecated. Do not use this custom property unless you are running in a mixed cell environment that includes at least one core group that contains a mixture of Version 7.0 and Version 6.x processes.

Note: If you are running in a mixed cell environment, and you have core groups that contain a mixture of Version 7.0 and Version 6.x processes, you must continue to use this custom property.

javax.sip.transaction.timer.b

Specifies the amount of time, in milliseconds, for the invite client transaction timeout timer (timer B) for the RFC 3261 specification.

Data type	String
Default	64*javax.sip.transaction.timer.t1

Note: This custom property is deprecated. Do not use this custom property unless you are running in a mixed cell environment that includes at least one core group that contains a mixture of Version 7.0 and Version 6.x processes.

Note: If you are running in a mixed cell environment, and you have core groups that contain a mixture of Version 7.0 and Version 6.x processes, you must continue to use this custom property.

javax.sip.transaction.timer.d

Specifies the wait time, in milliseconds, before retransmission of the invite response for timer D for the RFC 3261 specification. This property is relevant for the invite client transaction.

Data type	String
Default	32000

Note: This custom property is deprecated. Do not use this custom property unless you are running in a mixed cell environment that includes at least one core group that contains a mixture of Version 7.0 and Version 6.x processes.

Note: If you are running in a mixed cell environment, and you have core groups that contain a mixture of Version 7.0 and Version 6.x processes, you must continue to use this custom property.

javax.sip.transaction.timer.e

Specifies, for UDP only, the amount of time, in milliseconds, before the retransmission of the initial non-invite request for timer E for the RFC 3261 specification. This property is relevant for the non-invite client transaction.

Data type	String
Default	javax.sip.transaction.timer.t1

Note: This custom property is deprecated. Do not use this custom property unless you are running in a mixed cell environment that includes at least one core group that contains a mixture of Version 7.0 and Version 6.x processes.

Note: If you are running in a mixed cell environment, and you have core groups that contain a mixture of Version 7.0 and Version 6.x processes, you must continue to use this custom property.

javax.sip.transaction.timer.f

Specifies the amount of time, in milliseconds, for the non-invite transaction timeout timer (timer F) for the RFC 3261 specification. This property is relevant for the non-invite client transaction.

Data type	String
Default	64*javax.sip.transaction.timer.t1

Note: This custom property is deprecated. Do not use this custom property unless you are running in a mixed cell environment that includes at least one core group that contains a mixture of Version 7.0 and Version 6.x processes.

Note: If you are running in a mixed cell environment, and you have core groups that contain a mixture of Version 7.0 and Version 6.x processes, you must continue to use this custom property.

javax.sip.transaction.timer.g

Specifies the amount of time, in milliseconds, before retransmission of an initial invite response for timer G for the RFC 3261 specification. This property is relevant for the invite server transaction.

Data type	String
Default	javax.sip.transaction.timer.t1

Note: This custom property is deprecated. Do not use this custom property unless you are running in a mixed cell environment that includes at least one core group that contains a mixture of Version 7.0 and Version 6.x processes.

Note: If you are running in a mixed cell environment, and you have core groups that contain a mixture of Version 7.0 and Version 6.x processes, you must continue to use this custom property.

javax.sip.transaction.timer.h

Specifies the amount of time, in milliseconds, to wait for an acknowledgement (ACK) receipt for timer H for the RFC 3261 specification. This property is relevant for the invite server transaction.

Data type	String
Default	64*javax.sip.transaction.timer.t1

Note: This custom property is deprecated. Do not use this custom property unless you are running in a mixed cell environment that includes at least one core group that contains a mixture of Version 7.0 and Version 6.x processes.

Note: If you are running in a mixed cell environment, and you have core groups that contain a mixture of Version 7.0 and Version 6.x processes, you must continue to use this custom property.

javax.sip.transaction.timer.i

Specifies the amount of time in milliseconds to wait for an ACK retransmission for timer I for the RFC 3261 specification. This property is relevant for the invite server transaction.

Data type	String
Default	javax.sip.transaction.timer.t4

Note: This custom property is deprecated. Do not use this custom property unless you are running in a mixed cell environment that includes at least one core group that contains a mixture of Version 7.0 and Version 6.x processes.

Note: If you are running in a mixed cell environment, and you have core groups that contain a mixture of Version 7.0 and Version 6.x processes, you must continue to use this custom property.

javax.sip.transaction.timer.j

Specifies the amount of time in milliseconds to wait for non-invite request retransmission for timer J for the RFC 3261 specification. This property is relevant for the non-invite server transaction.

Data type	String
Default	64*javax.sip.transaction.timer.t1

Note: This custom property is deprecated. Do not use this custom property unless you are running in a mixed cell environment that includes at least one core group that contains a mixture of Version 7.0 and Version 6.x processes.

Note: If you are running in a mixed cell environment, and you have core groups that contain a mixture of Version 7.0 and Version 6.x processes, you must continue to use this custom property.

javax.sip.transaction.timer.k

Specifies the amount of time, in milliseconds, to wait for non-INVITE response retransmissions for timer K for the RFC 3261 specification. This property is relevant for the non-invite client transaction.

Data type	String
Default	javax.sip.transaction.timer.t4

Note: This custom property is deprecated. Do not use this custom property unless you are running in a mixed cell environment that includes at least one core group that contains a mixture of Version 7.0 and Version 6.x processes.

Note: If you are running in a mixed cell environment, and you have core groups that contain a mixture of Version 7.0 and Version 6.x processes, you must continue to use this custom property.

javax.sip.transaction.timer.non.invite.server

Specifies the amount of time, in milliseconds, for an Application Programming Interface (API) timer for the application to respond to a non-invite request. This property is relevant for non-invite server transactions.

This timer is not defined in the RFC specification. This property is needed to stop the transaction if the application does not generate a final response to the request. The timer starts when the request arrives in the stack and stops when a response is generated by the application. If no response is generated before the timer stops, then the transaction completes.

Data type	String
Default	34000

javax.sip.transaction.timer.invite.server

Specifies the amount of time, in milliseconds, for the timer to keep the invite server transaction in the complete state. This timer is not defined in the RFC specification.

To avoid creating a new server transaction when a client retransmits an invite request, keep the completed server transaction for a period of time before removing invite retransmissions. This timer is started when the transaction changes to the terminated state. When the timer completes, the transaction is removed.

Data type	String
Default	32000

javax.sip.transaction.timer.cancel

Specifies the amount of timer, in milliseconds, for the timer to keep the cancelled client transaction in the proceeding state before completing the cancelled transaction for the RFC 3261 9.1 specification. This property is relevant for the invite client transaction.

Data type	String
Default	64*javax.sip.transaction.timer.t1

SIP_RFC3263_nameserver

Specifies whether to allow a SIP URI to be resolved through Domain Name System (DNS) into the IP address, port, and transport protocol of the next hop.

The value of the property is a string containing one or two address and port tuples, where two tuples are separated by a space. The following examples specify a one address and port tuple or a two address and port tuple.

dottedDecimalAddress@.port

hostname.domain@port

IPV6address@port

The following example values represent a single tuple.

- 1.2.3.4@53
- example.com@53
- a:b:c::d@53

The following example values represent two tuples separated by a space.

- 1.2.3.4@53 example.com@53
- a:b:c::d@53 9.32.211.14@53

Data type	String
Default	null

Note: This custom property is deprecated. Do not use this custom property unless you are running in a mixed cell environment that includes at least one core group that contains a mixture of Version 7.0 and Version 6.x processes.

Note: If you are running in a mixed cell environment, and you have core groups that contain a mixture of Version 7.0 and Version 6.x processes, you must continue to use this custom property.

thread.message.queue.max.size

Specifies the maximum number of events allowed in the container threads queue. When this number is exceeded, the proxy server is notified that the container is overloaded and requests for new sessions are not accepted. Instead, the container returns an error message that indicates that the container is temporarily unavailable.

This value represents the total number of messages for all queues and reflects the state of the CPU. When the CPU approaches 100%, the maximum value for this custom property is reached quickly. Configure your system to limit the queue size and prevent the queue from reaching this threshold.

Data type	String
Default	1000

weight.overload.watermark

Specifies the threshold value for the internal weight calculated by the container. When the container calculates the internal weight to be higher than the value specified, an overloaded container becomes available for service again.

This custom property represents a percentage of the maximum internal weight, such as 30 percent when the default value is set. When the high-water mark, or maximum threshold, is exceeded, the container waits until the weight drops below the maximum weight. This value cannot exceed `10`.

Data type	String
Default	3

sip.container.heartbeat.enabled

Specifies whether or not SIP network outage detection is enabled for the SIP container. SIP network outage detection allows the SIP proxy to send keepalive messages to the SIP container if the value of this property is set to `true`.

If the value is set to `false` for the SIP container, then this property has no effect on the SIP proxy. However, if the value is set to `true` for the SIP container, the value should also be set to `true` for the SIP proxy to ensure that keepalive messages are answered at the SIP container and not presented to the application.

Data type	String
Default	true

Using DNS procedures to locate SIP servers

The Session Initiation Protocol (SIP) can use Domain Name Server (DNS) procedures for a client to resolve a SIP Uniform Resource Identifier (URI).

About this task

WebSphere Application Server provides support for the RFC 3263 standard. This allows a SIP URI to be resolved through DNS into the IP address, port, and transport protocol of the next hop to contact.

Note: SIP does not support use of DNS procedures for a server to send a response to a back-up client if the primary client fails.

Complete these steps to configure WebSphere Application Server to support the RFC 3263 standard.

1. Start WebSphere Application Server.
2. From the administrative console, expand **Servers**, and click **Application servers** → ***serverName***.

3. Under **General Properties**, check the **Enable locating SIP servers using DNS NAPTR records** checkbox, then fill in the **Primary DNS server name** and **Secondary DNS server name** fields.
4. Click **Apply** to save your changes.
5. Restart WebSphere Application Server.

What to do next

You must configure your DNS server in order for RFC 3263 support to work for the SIP container. The following example is a BIND db file for configuring RFC 3263 support on a DNS server.

```
; Copyright (C) 2004  Internet Systems Consortium, Inc. ("ISC")
; Copyright (C) 2001  Internet Software Consortium.
;
; Permission to use, copy, modify, and distribute this software for any
; purpose with or without fee is hereby granted, provided that the above
; copyright notice and this permission notice appear in all copies.
;
; THE SOFTWARE IS PROVIDED "AS IS" AND ISC DISCLAIMS ALL WARRANTIES WITH
; REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY
; AND FITNESS.  IN NO EVENT SHALL ISC BE LIABLE FOR ANY SPECIAL, DIRECT,
; INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM
; LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE
; OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR
; PERFORMANCE OF THIS SOFTWARE.

; $Id: include.db,v 1.2.206.1 2004/03/06 10:22:13 marka Exp $

; Test $INCLUDE current domain name and origin semantics

example.com. 43200 IN SOA ns.example.com. email.example.com. ( 2003032001 10800 3600 604800 86400 )
;
example.com.             43200 IN NS      ns.example.com.
;
ns.example.com.          43200 IN A       10.0.0.20
sipserver1.example.com.  43200 IN A       10.0.0.21
sipserver2.example.com.  43200 IN A       10.0.0.22
sipserver3.example.com.  43200 IN A       10.0.0.23
;
router.example.com.        43200  IN CNAME sipserver3
;
sipserver1.example.com.  43200 IN AAAA    fec0:0:0:0:0:0:0:abcd
sipserver2.example.com.  43200 IN AAAA    fec0:0:0:0:0:0:0:abba
;
_sip._udp.example.com.   43200 IN SRV 2 0 5060  sipserver1.example.com.
_sip._udp.example.com.   43200 IN SRV 2 0 5060  sipserver2.example.com.
_sip._tcp.example.com.   43200 IN SRV 1 4 5060  sipserver1.example.com.
_sip._tcp.example.com.   43200 IN SRV 1 2 5060  sipserver2.example.com.
_sips._tcp.example.com.  43200 IN SRV 0 1 5061  sipserver1.example.com.
_sips._tcp.example.com.  43200 IN SRV 0 0 5061  sipserver2.example.com.
;
example.com. 43200 IN NAPTR 0 0 "s" "SIPS+D2T" "" _sips._tcp.example.com.
example.com. 43200 IN NAPTR 1 0 "s" "SIP+D2T"  "" _sip._tcp.example.com.
example.com. 43200 IN NAPTR 2 0 "s" "SIP+D2U"  "" _sip._udp.example.com.
```

SIP container settings

Use this page to configure the SIP container settings for Session Initiation Protocol (SIP).

To view this administrative console page, click **Application servers** → ***serverName*** → **SIP container settings** → **SIP container**.

Related tasks

../ae/ttrb_confighangdet.dita

Related information

Server collection
Use this topic to learn how to navigate within the administrative console to the pages where you can view information about the application servers, generic servers, Java message service (JMS) servers, and Web servers that are defined for your system.

Use SIP proxy for external domains

Specifies that, when true, the SIP container assumes the existence of the SIP Proxy running on another server process and therefore, delegates routing outbound traffic to that component.

Data type	Boolean
Default	True

Maximum application sessions

Specifies the maximum number of SIP application sessions that the container manages. When the maximum is reached, no new SIP conversations are started. When the maximum is exceeded in a clustered environment, the server does not forward new dialogs until the number of application sessions no longer exceeds the maximum.

Application sessions are typically created by new incoming calls, but can also be created by other events. The application session count does not impact failover, but applies only to new sessions that are created as a result of incoming calls.

When application sessions are transferred from one application server to another due to failover, the active application server inherits the sessions created on the failed server. In addition, the servlet might create a new application session in the SIP container by calling SipFactory.createApplicationSession().

New application sessions created for events other than starting SIP conversations are not controlled by this setting. But all new application sessions are included when computing the maximum number of application sessions allowed. Thus, all active application sessions, including those not related to starting SIP conversations, can cause the maximum to be exceeded.

Data type	Integer
Default	120000 (recommended)
Range	1 <= n <= java.lang.Integer.MAX_VALUE

Maximum messages per averaging period

Specifies the maximum amount of SIP messages processed per averaging period. The averaging period is the period of time during which the average number of messages received by the container is calculated.

This average is used to determine the load for the container and to determine if the number of messages is approaching the maximum. When the maximum is exceeded, the stand-alone server or the proxy server continues to handle all in-dialog messages. Other non-dialog requests are rejected. When a container is in an overloaded state, the proxy server returns a 503 error.

Data type	Integer
Default	5000 (recommended)
Range	1 <= n <= java.lang.Integer.MAX_VALUE

Maximum dispatch queue size

Specifies the size of the internal dispatch queue. When the maximum queue size threshold is reached, the container queue becomes overloaded and begins to drop requests for new sessions. In this case, the container does not report its overloaded state to the proxy server.

Configure your system to limit the queue size to prevent the queue from reaching this threshold. If the internal queue reaches the overloaded state, incoming UDP packets are dropped until the queue is no longer in an overloaded state. Limiting the queue size enables better recovery if the CPU is used by other processes or threads and prevents the container from reaching out-of-memory conditions. When the value is set to `0`, the queue size is unlimited.

Data type	Integer
Default	5000 (recommended)
Range	0 <= n <= java.lang.Integer.MAX_VALUE

Maximum response time

Specifies the maximum response time, in milliseconds, for an application. When the amount of time is exceeded, the container notifies the clustering framework that it is unavailable. You can disable this feature in the administrative console by deselecting the check box and specifying a value of `0`.

Use the maximum SIP response time setting cautiously because the calculated response time does not match the behavior of all applications. For requests, such as INVITE requests, where the responses are generated as a result of a user interaction, the calculated response time is extensive. However, the extensive response time is not caused by a delay in the SIP container. Therefore, you should not calculate the response time as a load factor. The recommended applications for effective calculation of response time are applications that respond immediately without a user interaction. The subscribe and register applications are relevant examples.

Data type	Integer
Default	0
Range	1 <= n <= java.lang.Integer.MAX_VALUE

Averaging period in milliseconds

Specifies the amount of time, in milliseconds, to use for calculating the maximum messages per averaging period. This setting is the sliding window during which the SIP container counts the number of messages sent to the container.

Data type	Integer
Default	1000 (recommended)
Range	1000 <= n <= java.lang.Integer.MAX_VALUE

Statistic update rate

Specifies control over the interval for which the container calculates averages and publishes statistics to Performance Monitoring Infrastructure.

Data type	Integer
Default	1000 (recommended)
Range	1000 <= n <= java.lang.Integer.MAX_VALUE

Locating SIP servers using DNS

Specifies whether to enable locating SIP servers using DNS (Directory Name Service).

A SIP Uniform Resource Identifier (URI) can be resolved through DNS into the Internet Protocol (IP) address, port, and transport protocol of the next hop.

The value for the **Primary DNS server name** or **Secondary DNS server name** fields is a string containing one address and port tuple. The following examples specify an address and port tuple.

Note: The container contacts the primary DNS server first; however, if the primary DNS server is unavailable, then the container contacts the secondary DNS server. If the secondary DNS server remains responsive, the container does not attempt to contact the primary DNS server.

dottedDecimalAddress@.port

hostname.domain@port

IPV6address@port

Data type	Boolean
Default	False

Primary DNS server name

Specifies an IP address and port tuple for the primary DNS server. If this server is not available, then the container sends a response to the secondary DNS server.

Data type	String
Default	empty string.

Secondary DNS server name

Specifies an IP address and port tuple for the secondary DNS server.

Data type	String
Default	empty string.

Thread pool

Specifies the available thread pools that you can select from the drop-down list for the SIP container to use when dispatching work. If you do not select a thread pool from the drop-down list, a default thread pool, which is automatically created by the container, is used.

It is recommended that you create a dedicated WebSphere thread pool for SIP applications. For general usage, this is a minimum of 15 threads and a maximum of 30 threads (with one thread per queue). This becomes convenient when combined with WebSphere hung thread detection. A hung thread can block many SIP messages, so it is important it be detected as soon as possible. However, the default hung thread detection threshold is too long for most SIP scenarios, so it is recommended you change it to 30 seconds. See the ″Configuring the hang detection policy″ topic (linked below) for the exact property names.

Data type	menu list
Default	None

SIP stack settings

Use this page to configure values for the Session Initiation Protocol (SIP) stack settings that are different from those specified in RFC 3261. The default values are the same as those specified for RFC 3261.

To view this administrative console page, click **Servers** → **Application servers** → ***serverName*** → **SIP container settings** → **SIP container** → **SIP stack**.

Automatically reply to INVITE with ″100 Trying″ response

Specifies whether to automatically reply to INVITE requests with a 100 Trying response. Disabling this setting might increase the number of INVITE retransmissions.

Data type	Boolean
Default	True

Hide message body

Specifies whether to hide message content in log files. Set the value to `True` to remove the message body text from SIP messages printed in the log files. This setting only affects the representation of the messages in log files.

Data type	Boolean
Default	False

Outbound connection timeout

Specifies the amount of time, in milliseconds, for creating outbound connections. This setting is relevant only for stream transports, such as Transmission Control Protocol (TCP) and Transport Layer Security (TLS).. The default value of `0` provides an infinite amount of time.

Data type	Integer
Default	0

Maximum transmission unit

Specifies the maximum transmission unit, in bytes, for outbound User Datagram Protocol (UDP) requests. The SIP stack measures the size of a request before sending it out on the UDP channel. If the request is higher than the value specified for PATH_MTU-200, 1300 bytes by default, then the transport is switched from UDP to TCP before transmission.

Increase this value to send larger requests over the UDP channel; however, messages might be truncated or discarded. See the RFC 3261-18.1.1 specification for details.

Data type	Integer
Default	1500

Outbound proxy

Specifies the fixed address for routing all outbound SIP messages. The format is address:port/transport, such as 1.2.3.4:5065/tcp.

Note: Do not use this setting if the container is fronted by an application server SIP proxy.

Data type	String
Default	empty string

SIP timers settings

Use this page to set values for the Session Initiation Protocol (SIP) timers that are different from those specified in RFC 3261. SIP timers provide a mechanism for session expiration.The default values are the same as those specified for RFC 3261.

To view this administrative console page, click **Servers** → **Application servers** → ***serverName*** → **SIP container settings** → **SIP container** → **SIP stack** → **SIP timers**.

The following SIP timers can be configured from the administrative console.

T1

Specifies the amount of time, in milliseconds, for a network round trip delay for timer T1 for the RFC 3261 specification. The value is used as a base for calculating some timers and is relevant for all types of transactions, such as client, server, invite, and non-invite transactions.

Data type	Integer
Default	500

A Specifies, for UDP only, the amount of time, in milliseconds, before retransmitting invite requests for timer A for the RFC 3261 specification. This setting is relevant for the invite client transaction.

Data type	Integer
Default	T1

B Specifies the amount of time, in milliseconds, for the invite client transaction timeout timer (timer B) for the RFC 3261 specification.

Data type	Integer
Default	T1*64

E Specifies, for UDP only, the amount of time, in milliseconds, before the retransmission of the initial non-invite request for timer E for the RFC 3261 specification. This setting is relevant for the non-invite client transaction.

Data type	Integer
Default	800

F Specifies the amount of time, in milliseconds, for the non-invite transaction timeout timer (timer F) for the RFC 3261 specification. This setting is relevant for the non-invite client transaction.

Data type	Integer
Default	24000

G Specifies the amount of time, in milliseconds, before retransmission of an initial invite response for timer G for the RFC 3261 specification. This setting is relevant for the invite server transaction.

Data type	Integer
Default	T1

H Specifies the amount of time, in milliseconds, to wait for an acknowledgement (ACK) receipt for timer H for the RFC 3261 specification. This setting is relevant for the invite server transaction.

Data type	Integer
Default	T1*64

J Specifies the amount of time, in milliseconds, to wait for non-invite request retransmission for timer J for the RFC 3261 specification. This setting is relevant for the non-invite server transaction.

Data type	Integer
Default	T1*64

T2

Specifies the maximum amount of time, in milliseconds, before retransmitting non-invite requests and invite responses for timer T2 for the RFC 3261 specification.

Data type	Integer
Default	4000

T4

Specifies the maximum amount of time, in milliseconds, for a message to remain in the network. This value is used as a base for calculating other timers for timer T4 for the RFC 3261 specification.

Data type	Integer
Default	5000

I Specifies the amount of time, in milliseconds, to wait for an ACK retransmission for timer I for the RFC 3261 specification. This setting is relevant for the invite server transaction.

Data type	Integer
Default	T4

K Specifies the amount of time, in milliseconds, to wait for non-INVITE response retransmissions for timer K for the RFC 3261 specification. This setting is relevant for the non-invite client transaction.

Data type	Integer
Default	T4

D

Specifies the amount of time to wait, in milliseconds, before retransmission of the invite response for timer D for the RFC 3261 specification. This setting is relevant for the invite client transaction.

Data type	Integer
Default	32000

SIP digest authentication settings

Use this page to configure Session Initiation Protocol (SIP) digest authentication settings that are different from those specified in RFC 3261. The default values are the same as those specified for RFC 3261.

To view this administrative console page, click **Security** → **Global Security** → **Authentication** → **Web and SIP Security** → **SIP digest authentication**.

Enable digest authentication integrity

Specifies the authentication integrity (auth-int) quality of protection (QOP) for digest authentication. Digest authentication defines two types of QOP: auth and auth-int. By default, basic authentication (auth) is used. If the value is set to `True`, the auth-int QOP is used, which is the highest level of protection.

Data type	Boolean
Default	False

Enable SIP basic authentication

Specifies the authentication (auth) quality of protection (QOP) for digest authentication. Digest authentication defines two types of QOP: auth and auth-int. By default, basic authentication (auth) is used. If the value is set to `True`, basic authentication will be performed. It will not be processed by the Trust Association Interceptor.

Data type	Boolean
Default	True

Enable multiple use of nonce

Specifies whether to enable multiple uses of the same nonce. If you use the same nonce more than once, then less system resources are required, however, your system is not as secure.

Data type	Boolean
Default	False

Enable nonce maximum age

Specifies the amount of time, in milliseconds, for which a nonce is valid. If the value is set to 1, then the amount of time is considered to be infinite.

Data type	Integer
Default	1

LDAP cache clean intervals

Specifies the amount of time that must expire, in minutes, before the LDAP cache is cleaned.

Data type	Integer
Default	120

LDAP password attribute name

Specifies the LDAP attribute name that stores the user password .

Data type	String
Default	userpassword

User cache clean intervals

Specifies the amount of time that must expire, in minutes, before the security subject cache is cleaned.

Data type	Integer
Default	15

Digest password server class

Specifies the Java class name that implements the PasswordServer interface.

Data type	String
Default	LdapPasswordServer

Hashedcredentials

Specifies the name of the LDAP field that contains the hashed credentials. If a value is specified for this setting, then this setting overrides the pws_atr_name setting.

Data type	String
Default	empty string

Hashedrealm

Specifies the realm for hashed credentials, if the hashed credentials setting is enabled.

Data type	String
Default	empty string

Configuring SIP timers

You can configure SIP timers to set values for the Session Initiation Protocol (SIP) timers that are different from default values specified in RFC 3261. SIP timers provide a mechanism for session expiration. The default values are the same as those specified for RFC 3261.

Before you begin

Before you begin

Ensure that servers are created.

About this task

You can set values for the SIP timers from the administrative console.

Note: If you have already used the custom properties to specify a value for a SIP timer, then the custom property value is the primary value. Therefore, the SIP timer value specified from the administrative console is not used.

1. From the administrative console, click **Servers** → ***server_name*** → **SIP container** → **SIP stack** → **SIP timers**.
2. Specify a value for a timer by clicking the Use Custom Value check box beside the timer.
3. Specify a value for the timer in the Value column.
4. Click **OK**.
5. Restart the application server.

Results

The container uses the times specified in the administrative console and not the RFC defaults.

SIP timer summary

Request for Comments (RFC) 3261, "SIP: Session Initiation Protocol," specifies various timers that SIP uses.

Table 8 summarizes for each SIP timer the default value, the section of RFC 3261 that describes the timer, and the meaning of the timer.

Table 8. Summary of SIP timers

Timer	Default value	Section	Meaning
T1	500 ms	17.1.1.1	Round-trip time (RTT) estimate
T2	4 sec.	17.1.2.2	Maximum retransmission interval for non-INVITE requests and INVITE responses
T4	5 sec.	17.1.2.2	Maximum duration that a message can remain in the network
Timer A	initially T1	17.1.1.2	INVITE request retransmission interval, for UDP only
Timer B	64*T1	17.1.1.2	INVITE transaction timeout timer
Timer D	> 32 sec. for UDP	17.1.1.2	Wait time for response retransmissions
	0 sec. for TCP and SCTP		
Timer E	initially T1	17.1.2.2	Non-INVITE request retransmission interval, UDP only
Timer F	64*T1	17.1.2.2	Non-INVITE transaction timeout timer
Timer G	initially T1	17.2.1	INVITE response retransmission interval
Timer H	64*T1	17.2.1	Wait time for ACK receipt

Table 8. Summary of SIP timers (continued)

Timer	Default value	Section	Meaning
Timer I	T4 for UDP	17.2.1	Wait time for ACK retransmissions
	0 sec. for TCP and SCTP		
Timer J	64*T1 for UDP	17.2.2	Wait time for retransmissions of non-INVITE requests
	0 sec. for TCP and SCTP		
Timer K	T4 for UDP	17.1.2.2	Wait time for response retransmissions
	0 sec. for TCP and SCTP		

Configuring digest authentication for SIP

You can configure Session Initiation Protocol (SIP) digest authentication settings that are different from those specified in RFC 3261. The default values are the same as those specified for RFC 3261.

Before you begin

Before you begin

Ensure that servers are created.

About this task

You can set values for the SIP timers from the administrative console.

Note: If you have already used the custom properties to specify a value for a SIP timer, then the custom property value is the primary value. Therefore, the SIP timer value specified from the administrative console is not used.

1. From the administrative console, click **Security** → **Global security** → **Web and SIP security** → **Digest authentication**.
2. Specify a value for one or more settings.
3. Click **OK**.
4. Restart the application server.

Results

The container uses the digest authentication values specified in the administrative console and not the RFC defaults.

Performing controlled failover of SIP applications

You can take an active application out of the loop via a controlled failover.

Before you begin

SipContainerMBean is used to initiate a server quiesce through wsadmin (command line interface). This MBean is used to set the container's weight to 0, which prevents new messages from being routed to it.

WebSphere Application Server PMI is used to monitor the server's active sessions. The remaining active sessions can be watched by enabling a counter on the server being quiesced. The server can be shut down once the number of active sessions reaches an acceptable level. A script can be written to monitor active sessions and shut down the server when an acceptable threshold is achieved.

About this task

Quiescing a single server

1. On an ND machine, start the wsadmin utility:
   a. Go to <nd_installation_path>/bin
   b. Run the command: ./setupCmdLine.sh
   c. Run the command: ./wsadmin.sh
   d. Verify that received: wsadmin>
2. Run the command: set scBean [$AdminControl queryNames type=SipContainerMBean,process=<server name>,*]
3. Run the command: $AdminControl invoke $scBean quiesce true

From the admin console, command line or scripts

Use the following commands to stop application servers from the admin console, command line or scripts:

- **Stop:** Quiesces the application server. The sessions will failover to another server. It is important to manually quiesce the server on shutdown.
- **Immediate Stop:** Stops the server, but bypasses the normal server quiesce process that supports in-flight requests to complete before shutting down the entire server process. This shutdown mode is faster than the normal server stop processing, but some application clients can receive exceptions.
- **Terminate:** Deletes the application server process. Use this if immediate stop fails to stop the server.

SIP PMI counters

The Session Initiation Protocol (SIP) provides the following counters in the WebSphere Performance Monitoring Infrastructure (PMI) to monitor the performance of SIP.

Counter definitions

Sip container module

Name	Key	Description	Granularity	Type	Level	Overhead	ID
Incoming traffic	incoming.traffic	Average number of messages handled by the container calculated over configurable period	Per server	CountStatistic	All	Low	3
New SIP App sessions	new.sip.app.session	Average number of new SIP application sessions created in the container and calculated over configurable period	Per server	CountStatistic	All	Low	4
Response Time	response.time	The average amount of time that it takes between when a message gets into the container and when a response is sent from the container.	Per server	CountStatistic	All	Low	5
Queue Size	queue.size	Size of the invoke queue in WebSphere Application Server	Per server	CountStatistic	All	Low	6

Session module

Name	Key	Description	Granularity	Type	Level	Overhead	ID
Number of active SIP sessions	active.sip.sessions	The number of SIP sessions belongs to each application	Per application	CountStatistic	All	Low	11
Number of active SIP application sessions	active.sip.app.sessions	The number of SIP application sessions belongs to each application	Per application	CountStatistic	All	Low	12

Inbound request module

Name	Key	Description	Granularity	Type	Level	Overhead	ID
Number of Inbound NOT SIP STANDARD requests	inbound.request.other	The number of inbound NOT SIP STANDARD requests that belong to each application	Per application	CountStatistic	All	Low	60
Number of Inbound REGISTER requests	inbound.request.register	The number of inbound REGISTER requests that belong to each application	Per application	CountStatistic	All	Low	61
Number of Inbound INVITE requests	inbound.request.invite	The number of inbound INVITE requests that belong to each application	Per application	CountStatistic	All	Low	62
Number of Inbound ACK requests	inbound.request.ack	The number of Inbound ACK requests that belong to each application	Per application	CountStatistic	All	Low	63
Number of Inbound OPTIONS requests	inbound.request.options	The number of Inbound OPTIONS requests that belong to each application	Per application	CountStatistic	All	Low	64
Number of Inbound BYE requests	inbound.request.bye	The number of Inbound BYE requests that belong to each application	Per application	CountStatistic	All	Low	65
Number of Inbound CANCEL requests	inbound.request.cancel	The number of Inbound CANCEL requests that belong to each application	Per application	CountStatistic	All	Low	66
Number of Inbound PRACK requests	inbound.request.prack	The number of Inbound PRACK requests that belong to each application	Per application	CountStatistic	All	Low	67
Number of Inbound INFO requests	inbound.request.info	The number of Inbound INFO requests that belong to each application	Per application	CountStatistic	All	Low	68
Number of Inbound SUBSCRIBE requests	inbound.request.subscribe	The number of Inbound SUBSCRIBE requests that belong to each application	Per application	CountStatistic	All	Low	69
Number of Inbound NOTIFY requests	inbound.request.notify	The number of Inbound NOTIFY requests that belong to each application	Per application	CountStatistic	All	Low	70
Number of Inbound MESSAGE requests	inbound.request.message	The number of Inbound MESSAGE requests that belong to each application	Per application	CountStatistic	All	Low	71
Number of Inbound PUBLISH requests	inbound.request.publish	The number of Inbound PUBLISH requests that belong to each application	Per application	CountStatistic	All	Low	72
Number of Inbound REFER requests	inbound.request.refer	The number of Inbound REFER requests that belong to each application	Per application	CountStatistic	All	Low	73
Number of Inbound UPDATE requests	inbound.request.update	The number of Inbound UPDATE requests that belong to each application	Per application	CountStatistic	All	Low	74

Inbound info response module

Name	Key	Description	Granularity	Type	Level	Overhead	ID
Number of Inbound 100 responses	inbound.response.info.100	The number of Inbound 100 (Trying) responses that belong to each application	Per application	CountStatistic	All	Low	1100
Number of Inbound 180 responses	inbound.response.info.180	The number of Inbound 180 (Ringing) responses that belong to each application	Per application	CountStatistic	All	Low	1180
Number of Inbound 181 responses	inbound.response.info.181	The number of Inbound 181 (Call being forwarded) responses that belong to each application	Per application	CountStatistic	All	Low	1181
Number of Inbound 182 responses	inbound.response.info.182	The number of Inbound 182 (Call Queued) responses that belong to each application	Per application	CountStatistic	All	Low	1182
Number of Inbound 183 responses	inbound.response.info.183	The number of Inbound 183 (Session Progress) responses that belong to each application	Per application	CountStatistic	All	Low	1183

Inbound success response module

Name	Key	Description	Granularity	Type	Level	Overhead	ID
Number of successful Inbound 200 responses	inbound.response.success.200	The number of Inbound 200 (OK) responses that belong to each application	Per application	CountStatistic	All	Low	1200
Number of successful Inbound 202 responses	inbound.response.success.202	The number of Inbound 202 (Accepted) responses that belong to each application	Per application	CountStatistic	All	Low	1202

Inbound redirect responses module

Name	Key	Description	Granularity	Type	Level	Overhead	ID
Number of Inbound 300 responses	inbound.response.redirect.300	The number of Inbound 300 (Multiple choices) responses that belong to each application	Per application	CountStatistic	All	Low	1300
Number of Inbound 301 responses	inbound.response.redirect.301	The number of Inbound 301 (Moved Permanently) responses that belong to each application	Per application	CountStatistic	All	Low	1301
Number of Inbound 302 responses	inbound.response.redirect.302	The number of Inbound 302 (Moved Temporarily) responses that belong to each application	Per application	CountStatistic	All	Low	1302
Number of Inbound 305 responses	inbound.response.redirect.305	The number of Inbound 305 (Use Proxy) responses that belong to each application	Per application	CountStatistic	All	Low	1305
Number of Inbound 380 responses	inbound.response.redirect.380	The number of Inbound 380 (Alternative Service) responses that belong to each application	Per application	CountStatistic	All	Low	1380

Inbound fail response module

Name	Key	Description	Granularity	Type	Level	Overhead	ID
Number of Inbound 400 responses	inbound.response.fail.400	The number of Inbound 400 (Bad Request) responses that belong to each application	Per application	CountStatistic	All	Low	1400
Number of Inbound 401 responses	inbound.response.fail.401	The number of Inbound 401 (Unauthorized) responses that belong to each application	Per application	CountStatistic	All	Low	1401
Number of Inbound 402 responses	inbound.response.fail.402	The number of Inbound 402 (Payment Required) responses that belong to each application	Per application	CountStatistic	All	Low	1402
Number of Inbound 403 responses	inbound.response.fail.403	The number of Inbound 403 (Forbidden) responses that belong to each application	Per application	CountStatistic	All	Low	1403
Number of Inbound 404 responses	inbound.response.fail.404	The number of Inbound 404 (Not Found) responses that belong to each application	Per application	CountStatistic	All	Low	1404
Number of Inbound 405 responses	inbound.response.fail.405	The number of Inbound 405 (Method Not Allowed) responses that belong to each application	Per application	CountStatistic	All	Low	1405
Number of Inbound 406 responses	inbound.response.fail.406	The number of Inbound 406 (Not Acceptable) responses that belong to each application	Per application	CountStatistic	All	Low	1406
Number of Inbound 407 responses	inbound.response.fail.407	The number of Inbound 407 (Proxy Authentication Required) responses that belong to each application	Per application	CountStatistic	All	Low	1407
Number of Inbound 408 responses	inbound.response.fail.408	The number of Inbound 408 (Request Timeout) responses that belong to each application	Per application	CountStatistic	All	Low	1408
Number of Inbound 410 responses	inbound.response.fail.410	The number of Inbound 410 (Gone) responses that belong to each application	Per application	CountStatistic	All	Low	1410
Number of Inbound 413 responses	inbound.response.fail.413	The number of Inbound 413 (Request Entity Too Large) responses that belong to each application	Per application	CountStatistic	All	Low	1413
Number of Inbound 414 responses	inbound.response.fail.414	The number of Inbound 414 (Request URI Too Long) responses that belong to each application	Per application	CountStatistic	All	Low	1414

Number of Inbound 415 responses	inbound.response.fail.415	The number of Inbound 415 (Unsupported Media Type) responses that belong to each application	Per application	CountStatistic	All	Low	1415
Number of Inbound 416 responses	inbound.response.fail.416	The number of Inbound 416 (Unsupported URI Scheme) responses that belong to each application	Per application	CountStatistic	All	Low	1416
Number of Inbound 420 responses	inbound.response.fail.420	The number of Inbound 420 (Bad Extention) responses that belong to each application	Per application	CountStatistic	All	Low	1420
Number of Inbound 421 responses	inbound.response.fail.421	The number of Inbound 421 (Extention Required) responses that belong to each application	Per application	CountStatistic	All	Low	1421
Number of Inbound 423 responses	inbound.response.fail.423	The number of Inbound 423 (Interval Too Brief) responses that belong to each application	Per application	CountStatistic	All	Low	1423
Number of Inbound 480 responses	inbound.response.fail.480	The number of Inbound 480 (Temporarly Unavailable) responses that belong to each application	Per application	CountStatistic	All	Low	1480
Number of Inbound 481 responses	inbound.response.fail.481	The number of Inbound 481 (Call Leg Done) responses that belong to each application	Per application	CountStatistic	All	Low	1481
Number of Inbound 482 responses	inbound.response.fail.482	The number of Inbound 482 (Loop Detected) responses that belong to each application	Per application	CountStatistic	All	Low	1482
Number of Inbound 483 responses	inbound.response.fail.483	The number of Inbound 483 (Too Many Hops) responses that belong to each application	Per application	CountStatistic	All	Low	1483
Number of Inbound 484 responses	inbound.response.fail.484	The number of Inbound 484 (Address Incomplete) responses that belong to each application	Per application	CountStatistic	All	Low	1484
Number of Inbound 485 responses	inbound.response.fail.485	The number of Inbound 485 (Ambiguous) responses that belong to each application	Per application	CountStatistic	All	Low	1485
Number of Inbound 486 responses	inbound.response.fail.486	The number of Inbound 486 (Busy Here) responses that belong to each application	Per application	CountStatistic	All	Low	1486

Number of Inbound 487 responses	inbound.response.fail.487	The number of Inbound 487 (Request Terminated) responses that belong to each application	Per application	CountStatistic	All	Low	1487
Number of Inbound 488 responses	inbound.response.fail.488	The number of Inbound 488 (Not Acceptable Here) responses that belong to each application	Per application	CountStatistic	All	Low	1488
Number of Inbound 491 responses	inbound.response.fail.491	The number of Inbound 491 (Request Pending) responses that belong to each application	Per application	CountStatistic	All	Low	1491
Number of Inbound 493 responses	inbound.response.fail.493	The number of Inbound 493 (Undecipherable) responses that belong to each application	Per application	CountStatistic	All	Low	1493

Inbound server fail response module

Name	Key	Description	Granularity	Type	Level	Overhead	ID
Number of Inbound 500 responses	inbound.response.serverFail.500	The number of Inbound 500 (Server Internal Error) responses that belong to each application	Per application	CountStatistic	All	Low	1500
Number of Inbound 501 responses	inbound.response.serverFail.501	The number of Inbound 501 (Not Implemented) responses that belong to each application	Per application	CountStatistic	All	Low	1501
Number of Inbound 502 responses	inbound.response.serverFail.502	The number of Inbound 502 (Bad Gateway) responses that belong to each application	Per application	CountStatistic	All	Low	1502
Number of Inbound 503 responses	inbound.response.serverFail.503	The number of Inbound 503 (Service Unavailable) responses that belong to each application	Per application	CountStatistic	All	Low	1503
Number of Inbound 504 responses	inbound.response.serverFail.504	The number of Inbound 504 (Server Timeout) responses that belong to each application	Per application	CountStatistic	All	Low	1504
Number of Inbound 505 responses	inbound.response.serverFail.505	The number of Inbound 505 (Version Not Supported) responses that belong to each application	Per application	CountStatistic	All	Low	1505
Number of Inbound 513 responses	inbound.response.serverFail.513	The number of Inbound 513 (Message Too Large) responses that belong to each application	Per application	CountStatistic	All	Low	1513

Inbound global fail response module

Name	Key	Description	Granularity	Type	Level	Overhead	ID
Number of Inbound 600 responses	inbound.response.globalFail.600	The number of Inbound 600 (Busy Everywhere) responses that belong to each application	Per application	CountStatistic	All	Low	1600
Number of Inbound 603 responses	inbound.response.globalFail.603	The number of Inbound 603 (Decline) responses that belong to each application	Per application	CountStatistic	All	Low	1603
Number of Inbound 604 responses	inbound.response.globalFail.604	The number of Inbound 604 (Does Not Exit Anywhere) responses that belong to each application	Per application	CountStatistic	All	Low	1604
Number of Inbound 606 responses	inbound.response.globalFail.606	The number of Inbound 606 (Not Acceptable Anywhere) responses that belong to each application	Per application	CountStatistic	All	Low	1606

Outbound request module

Name	Key	Description	Granularity	Type	Level	Overhead	ID
Number of Outbound NOT SIP STANDARD requests	outbound.request.other	The number of Outbound NOT SIP STANDARD requests that belong to each application	Per application	CountStatistic	All	Low	80
Number of Outbound REGISTER requests	outbound.request.register	The number of Outbound REGISTER requests that belong to each application	Per application	CountStatistic	All	Low	81
Number of Outbound INVITE requests	outbound.request.invite	The number of Outbound INVITE requests that belong to each application	Per application	CountStatistic	All	Low	82
Number of Outbound ACK requests	outbound.request.ack	The number of Outbound ACK requests that belong to each application	Per application	CountStatistic	All	Low	83
Number of Outbound OPTIONS requests	outbound.request.options	The number of Outbound OPTIONS requests that belong to each application	Per application	CountStatistic	All	Low	84
Number of Outbound BYE requests	outbound.request.bye	The number of Outbound BYE requests that belong to each application	Per application	CountStatistic	All	Low	85
Number of Outbound CANCEL requests	outbound.request.cancel	The number of Outbound CANCEL requests that belong to each application	Per application	CountStatistic	All	Low	86
Number of Outbound PRACK requests	outbound.request.prack	The number of Outbound PRACK requests that belong to each application	Per application	CountStatistic	All	Low	87
Number of Outbound INFO requests	outbound.request.info	The number of Outbound INFO requests that belong to each application	Per application	CountStatistic	All	Low	88
Number of Outbound SUBSCRIBE requests	outbound.request.subscribe	The number of Outbound SUBSCRIBE requests that belong to each application	Per application	CountStatistic	All	Low	89
Number of Outbound NOTIFY requests	outbound.request.notify	The number of Outbound NOTIFY requests that belong to each application	Per application	CountStatistic	All	Low	90
Number of Outbound MESSAGE requests	outbound.request.message	The number of Outbound MESSAGE requests that belong to each application	Per application	CountStatistic	All	Low	91
Number of Outbound PUBLISH requests	outbound.request.publish	The number of Outbound PUBLISH requests that belong to each application	Per application	CountStatistic	All	Low	92
Number of Outbound REFER requests	outbound.request.refer	The number of Outbound REFER requests that belong to each application	Per application	CountStatistic	All	Low	93
Number of Outbound UPDATE requests	outbound.request.update	The number of Outbound UPDATE requests that belong to each application	Per application	CountStatistic	All	Low	94

Outbound info response module

Name	Key	Description	Granularity	Type	Level	Overhead	ID
Number of Outbound 100 responses	outbound.response.info.100	The number of Outbound 100 (Trying) responses that belong to each application	Per application	CountStatistic	All	Low	2100
Number of Outbound 180 responses	outbound.response.info.180	The number of Outbound 180 (Ringing) responses that belong to each application	Per application	CountStatistic	All	Low	2180
Number of Outbound 181 responses	outbound.response.info.181	The number of Outbound 181 (Call being forwarded) responses that belong to each application	Per application	CountStatistic	All	Low	2181
Number of Outbound 182 responses	outbound.response.info.182	The number of Outbound 182 (Call Queued) responses that belong to each application	Per application	CountStatistic	All	Low	2182
Number of Outbound 183 responses	outbound.response.info.183	The number of Outbound 183 (Session Progress) responses that belong to each application	Per application	CountStatistic	All	Low	2183

Outbound success response module

Name	**Key**	**Description**	**Granularity**	**Type**	**Level**	**Overhead**	ID
Number of Outbound 200 responses	outbound.response.success.200	The number of Outbound 200 (OK) responses that belong to each application	Per application	CountStatistic	All	Low	2200
Number of Outbound 202 responses	outbound.response.success.202	The number of Outbound 202 (Accepted) responses that belong to each application	Per application	CountStatistic	All	Low	2202

Outbound redirect response module

Name	Key	Description	Granularity	Type	Level	Overhead	ID
Number of Outbound 300 responses	outbound.response.redirect.300	The number of Outbound 300 (Multiple choices) responses that belong to each application	Per application	CountStatistic	All	Low	2300
Number of Outbound 301 responses	outbound.response.redirect.301	The number of Outbound 301 (Moved Permanently) responses that belong to each application	Per application	CountStatistic	All	Low	2301
Number of Outbound 302 responses	outbound.response.redirect.302	The number of Outbound 302 (Moved Temporarily) responses that belong to each application	Per application	CountStatistic	All	Low	2302
Number of Outbound 305 responses	outbound.response.redirect.305	The number of Outbound 305 (Use Proxy) responses that belong to each application	Per application	CountStatistic	All	Low	2305
Number of Outbound 380 responses	outbound.response.redirect.380	The number of Outbound 380 (Alternative Service) responses that belong to each application	Per application	CountStatistic	All	Low	2380

Outbound fail response module

Name	Key	Description	Granularity	Type	Level	Overhead	ID
Number of Outbound 400 responses	outbound.response.fail.400	The number of Outbound 400 (Bad Request) responses that belong to each application	Per application	CountStatistic	All	Low	2400
Number of Outbound 401 responses	outbound.response.fail.401	The number of Outbound 401 (Unauthorized) responses that belong to each application	Per application	CountStatistic	All	Low	2401
Number of Outbound 402 responses	outbound.response.fail.402	The number of Outbound 402 (Payment Required) responses that belong to each application	Per application	CountStatistic	All	Low	2402
Number of Outbound 403 responses	outbound.response.fail.403	The number of Outbound 403 (Forbidden) responses that belong to each application	Per application	CountStatistic	All	Low	2403
Number of Outbound 404 responses	outbound.response.fail.404	The number of Outbound 404 (Not Found) responses that belong to each application	Per application	CountStatistic	All	Low	2404
Number of Outbound 405 responses	outbound.response.fail.405	The number of Outbound 405 (Method Not Allowed) responses that belong to each application	Per application	CountStatistic	All	Low	2405
Number of Outbound 406 responses	outbound.response.fail.406	The number of Outbound 406 (Not Acceptable) responses that belong to each application	Per application	CountStatistic	All	Low	2406
Number of Outbound 407 responses	outbound.response.fail.407	The number of Outbound 407 (Proxy Authentication Required) responses that belong to each application	Per application	CountStatistic	All	Low	2407
Number of Outbound 408 responses	outbound.response.fail.408	The number of Outbound 408 (Request Timeout) responses that belong to each application	Per application	CountStatistic	All	Low	2408
Number of Outbound 410 responses	outbound.response.fail.410	The number of Outbound 410 (Gone) responses that belong to each application	Per application	CountStatistic	All	Low	2410
Number of Outbound 413 responses	outbound.response.fail.413	The number of Outbound 413 (Request Entity Too Large) responses that belong to each application	Per application	CountStatistic	All	Low	2413
Number of Outbound 414 responses	outbound.response.fail.414	The number of Outbound 414 (Request URI Too Long) responses that belong to each application	Per application	CountStatistic	All	Low	2414

Number of Outbound 415 responses	outbound.response.fail.415	The number of Outbound 415 (Unsupported Media Type) responses that belong to each application	Per application	CountStatistic	All	Low	2415
Number of Outbound 416 responses	outbound.response.fail.416	The number of Outbound 416 (Unsupported URI Scheme) responses that belong to each application	Per application	CountStatistic	All	Low	2416
Number of Outbound 420 responses	outbound.response.fail.420	The number of Outbound 420 (Bad Extention) responses that belong to each application	Per application	CountStatistic	All	Low	2420
Number of Outbound 421 responses	outbound.response.fail.421	The number of Outbound 421 (Extention Required) responses that belong to each application	Per application	CountStatistic	All	Low	2421
Number of Outbound 423 responses	outbound.response.fail.423	The number of Outbound 423 (Interval Too Brief) responses that belong to each application	Per application	CountStatistic	All	Low	2423
Number of Outbound 480 responses	outbound.response.fail.480	The number of Outbound 480 (Temporarly Unavailable) responses that belong to each application	Per application	CountStatistic	All	Low	2480
Number of Outbound 481 responses	outbound.response.fail.481	The number of Outbound 481 (Call Leg Done) responses that belong to each application	Per application	CountStatistic	All	Low	2481
Number of Outbound 482 responses	outbound.response.fail.482	The number of Outbound 482 (Loop Detected) responses that belong to each application	Per application	CountStatistic	All	Low	2482
Number of Outbound 483 responses	outbound.response.fail.483	The number of Outbound 483 (Too Many Hops) responses that belong to each application	Per application	CountStatistic	All	Low	2483
Number of Outbound 484 responses	outbound.response.fail.484	The number of Outbound 484 (Address Incomplete) responses that belong to each application	Per application	CountStatistic	All	Low	2484
Number of Outbound 485 responses	outbound.response.fail.485	The number of Outbound 485 (Ambiguous) responses that belong to each application	Per application	CountStatistic	All	Low	2485
Number of Outbound 486 responses	outbound.response.fail.486	The number of Outbound 486 (Busy Here) responses that belong to each application	Per application	CountStatistic	All	Low	2486

Number of Outbound 487 responses	outbound.response.fail.487	The number of Outbound 487 (Request Terminated) responses that belong to each application	Per application	CountStatistic	All	Low	2487
Number of Outbound 488 responses	outbound.response.fail.488	The number of Outbound 488 (Not Acceptable Here) responses that belong to each application	Per application	CountStatistic	All	Low	2488
Number of Outbound 491 responses	outbound.response.fail.491	The number of Outbound 491 (Request Pending) responses that belong to each application	Per application	CountStatistic	All	Low	2491
Number of Outbound 493 responses	outbound.response.fail.493	The number of Outbound 493 (Undecipherable) responses that belong to each application	Per application	CountStatistic	All	Low	2493

Outbound server fail response module

Name	Key	Description	Granularity	Type	Level	Overhead	ID
Number of Outbound 500 responses	outbound.response.serverFail.500	The number of Outbound 500 (Server Internal Error) responses that belong to each application	Per application	CountStatistic	All	Low	2500
Number of Outbound 501 responses	outbound.response.serverFail.501	The number of Outbound 501 (Not Implemented) responses that belong to each application	Per application	CountStatistic	All	Low	2501
Number of Outbound 502 responses	outbound.response.serverFail.502	The number of Outbound 502 (Bad Gateway) responses that belong to each application	Per application	CountStatistic	All	Low	2502
Number of Outbound 503 responses	outbound.response.serverFail.503	The number of Outbound 503 (Service Unavailable) responses that belong to each application	Per application	CountStatistic	All	Low	2503
Number of Outbound 504 responses	outbound.response.serverFail.504	The number of Outbound 504 (Server Timeout) responses that belong to each application	Per application	CountStatistic	All	Low	2504
Number of Outbound 505 responses	outbound.response.serverFail.505	The number of Outbound 505 (Version Not Supported) responses that belong to each application	Per application	CountStatistic	All	Low	2505
Number of Outbound 513 responses	outbound.response.serverFail.513	The number of Outbound 513 (Message Too Large) responses that belong to each application	Per application	CountStatistic	All	Low	2513

Outbound global fail response module

Name	Key	Description	Granularity	Type	Level	Overhead	ID
Number of Outbound 600 responses	outbound.response.globalFail.600	The number of Outbound 600 (Busy Everywhere) responses that belong to each application	Per application	CountStatistic	All	Low	2600
Number of Outbound 603 responses	outbound.response.globalFail.603	The number of Outbound 603 (Decline) responses that belong to each application	Per application	CountStatistic	All	Low	2603
Number of Outbound 604 responses	outbound.response.globalFail.604	The number of Outbound 604 (Does Not Exit Anywhere) responses that belong to each application	Per application	CountStatistic	All	Low	2604
Number of Outbound 606 responses	outbound.response.globalFail.606	The number of Outbound 606 (Not Acceptable Anywhere) responses that belong to each application	Per application	CountStatistic	All	Low	2606

Developing SIP applications

A SIP application is a set of SIP servlets packaged in a SIP application archive file (SAR).

About this task

A SIP servlet is an application component managed by the SIP container that performs SIP signaling. The programming and deployment models are analogous to Web servlets and therefore will be mapped to the WebSphere administrative model accordingly. It is possible to include Web servlets in a SAR file (along with the required web.xml deployment descriptor) to create what is known as a converged application. See JSR 116 for details on SIP applications, servlets, converged applications, and status codes.

Developing a custom trust association interceptor

When you develop Session Initiation Protocol (SIP) applications, you can create a custom trust association interceptor (TAI).

Before you begin

You may want to familiarize yourself with the general TAI information contained in the Trust Associations documentation. Developing a SIP TAI is similar to developing any other custom interceptors used in trust associations. In fact, a custom TAI for a SIP application is actually an extension of the trust association interceptor model. Refer to the Developing a custom interceptor for trust associations section for more details.

About this task

TAI can be invoked by a SIP servlet request or a SIP servlet response. To implement a custom SIP TAI, you need to write your own Java class.

1. Write a Java class that extends the `com.ibm.wsspi.security.tai.BaseTrustAssociationInterceptor` class and implements the `com.ibm.websphere.security.tai.SIPTrustAssociationInterceptor` interface. Those classes are defined in the *WASProductDir*/plugins/com.ibm.ws.sip.container_1.0.0.jar file, where *WASProductDir* is the fully qualified path name of the directory in which WebSphere Application Server is installed.
2. Declare the following Java methods:

   `public int initialize(Properties properties) throws WebTrustAssociationFailedException;`
   This is invoked before the first message is processed so that the implementation can allocate any resources it needs. For example, it could establish a connection to a database. `WebTrustAssociationFailedException` is defined in the *WASProductDir*/plugins/ com.ibm.ws.runtime_1.0.0.jar file. The value of the `properties` argument comes from the **Custom Properties** set in this step.

   `public void cleanup();`
   This is invoked when the TAI should free any resources it holds. For example, it could close a connection to a database.

   `public boolean isTargetProtocolInterceptor(SipServletMessage sipMsg) throws WebTrustAssociationFailedException;`
   Your custom TAI should use this method to handle the `sipMsg` message. If the method returns `false`, WebSphere ignores your TAI for `sipMsg`.

   `public TAIResult negotiateValidateandEstablishProtocolTrust (SipServletRequest req, SipServletResponse resp) throws WebTrustAssociationFailedException;`
   This method returns a TAIResult that indicates the status of the message being processed and a user ID or the unique ID for the user who is trying to authenticate. If authentication succeeds, the TAIResult should contain the status HttpServletResponse.SC_OK and a principal. If authentication fails, the TAIResult should contain a return code of

HttpServletResponse.SC_UNAUTHORIZED (401), SC_FORBIDDEN (403), or SC_PROXY_AUTHENTICATION_REQUIRED (407). This only indicates whether or not the container should accept a message for further processing. To challenge an incoming request, the TAI implementation must generate and send its own SipServletResponse containing a challenge. The exception should be thrown for internal TAI errors. Table 9 describes the argument values and resultant actions for the negotiateValidateandEstablishProtocolTrust method.

Table 9. Description of negotiateValidateandEstablishProtocolTrust arguments and actions

Argument or action	For a SIP request	For a SIP response
Value of `req` argument	The incoming request	Null
Value of `resp` argument	Null	The incoming response
Action for valid response credentials	Return TAIResult.status containing SC_OK and a user ID or unique ID	Return TAIResult.status containing SC_OK and a user ID or unique ID
Action for incorrect response credentials	Return the TAIResult with the 4*xx* status	Return the TAIResult with the 4*xx* status

The sequence of events is as follows:

a. The SIP container maps initial requests to applications by using the rules in each applications deployment descriptor; subsequent messages are mapped based on JSR 116 mechanisms.

b. If any of the applications require security, the SIP container invokes any defined TAI implementations for the message.

c. If the message passes security, the container invokes the corresponding applications.

Your TAI implementation can modify a SIP message, but the modified message will not be usable within the request mapping process, because it finishes before the container invokes the TAI.

The com.ibm.wsspi.security.tai.TAIResult class, defined in the *WASProductDir*/plugins/ com.ibm.ws.runtime_1.0.0.jar file, has three static methods for creating a TAIResult. The TAIResult `create` methods take an int type as the first parameter. WebSphere Application Server expects the result to be a valid HTTP request return code and is interpreted as follows:

If the value is HttpServletResponse.SC_OK, this response tells WebSphere Application Server that the TAI has completed its negotiation. The response also tells WebSphere Application Server use the information in the TAIResult to create a user identity.

The created TAIResults have the meanings shown in Table 10.

Table 10. Meanings of TAIResults

TAIResult	Explanation
public static TAIResult create(int status);	Indicates a status to WebSphere Application Server. The status should not be SC_OK because the identity information is provided.
public static TAIResult create(int status, String principal);	Indicates a status to WebSphere Application Server and provides the user ID or the unique ID for this user. WebSphere Application Server creates credentials by querying the user registry.
public static TAIResult create(int status, String principal, Subject subject);	Indicates a status to WebSphere Application Server, the user ID or the unique ID for the user, and a custom Subject. If the Subject contains a Hashtable, the principal is ignored. The contents of the Subject becomes part of the eventual user Subject.

`public String getVersion();`

This method returns the version number of the current TAI implementation.

public String getType();

This method's return value is implementation-dependent.

3. Compile the implementation after you have implemented it. For example: `/opt/WebSphere/AppServer/java/bin/javac -classpath  /opt/WebSphere/AppServer/plugins/com.ibm.ws.runtime_1.0.0.jar;/opt/WebSphere/AppServer/lib/j2ee.jar;/opt/WebSphere/AppServer/plugins/com.ibm.ws.sip.container_1.0.0.jar myTAIImpl.java`
   a. For each server within a cluster, copy the class file to a location in the WebSphere class path (preferably the *WASProductDir*/plugin/ directory).
   b. Restart all the servers.
4. Delete the default WebSEAL interceptor in the administrative console and click **New** to add your custom interceptor. Verify that the class name is dot-separated and appears in the class path.
5. Click the **Custom Properties** link to add additional properties that are required to initialize the custom interceptor. These properties are passed to the `initialize(Properties properties)` method of your implementation when it extends the `com.ibm.websphere.security.WebSphereBaseTrustAssociationInterceptor` as described in the previous step.
6. Save and synchronize (if applicable) the configuration.
7. Restart the servers for the custom interceptor to take effect.

Developing SIP applications that support PRACK

A SIP response to an INVITE request can be final or provisional. Final responses are always sent reliably, but provisional responses typically are not. For cases where you need to send a provisional response reliably, you can use the PRACK (Provisional response acknowledgement) method.

Before you begin

For you to be able to develop applications that support PRACK, the following criteria must be met:

- The client that sends the INVITE request must put a 100rel tag in the Supported or the Require header to indicate that the client supports PRACK.
- The SIP servlet must respond by invoking the sendReliably() method instead of the send() method to send the response.

About this task

PRACK is described in the following standards:

- RFC 3262 ("Reliability of Provisional Responses in the Session Initiation Protocol (SIP)"), which extends RFC 3261 ("SIP: Session Initiation Protocol"), adding PRACK and the option tag 100rel.
- Section 6.7.1 ("Reliable Provisional Responses") of JSR 116 ("SIP Servlet API Version 1.0").
- For an application acting as a proxy, do this:
  – Make your application generate and send a reliable provisional response for any INVITE request that has no tag in the To field.
- For an application acting as a user agent client (UAC), do this:
  – Make your application add the 100rel tag to outgoing INVITE requests. The option tag must appear in either the Supported header or the Require header.
  – Within your application's doProvisionalResponse(...) method, prepare the application to create and send PRACK requests for incoming reliable provisional responses. The application must create the PRACK request on the response's dialog through a SipSession.createRequest(...) method, and it must set the RAck header according to RFC 3262 Section 7.2 ("RAck").

  - The application that acts as an UAC will not receive doPrack( ) methods. The UAC sends INVITE and receives Reliable responses. When the UAC receives the Reliable response, it sends PRACK a request to the UAS and receives a 200 OK on the PRACK so it should next implement doResponse( ) in order to receive it.
- For an application acting as a user agent server (UAS), do this:
  - If an incoming INVITE request requires the 100rel tag, trying to send a 101-199 response unreliably by using the send() method causes an Exception.
  - Make the application declare a SipErrorListener to receive noPrackReceived() events when a reliable provisional response is not acknowledged within 64*T1 seconds, where T1 is a SIP timer. Within the noPrackReceived() event processing, the application should generate and send a 5*xx* error response for the associated INVITE request per JSR 116 Section 6.7.1.
  - Make the application have at most one outstanding, unacknowledged reliable provisional response. Trying to send another one before the first's acknowledgement results in an Exception.
  - Make sure that the application enforces the RFC 3262 offer/answer semantics surrounding PRACK requests containing session descriptions. Specifically, a servlet must not send a 2*xx* final response if any unacknowledged provisional responses contained a session description.

Setting up SIP application composition

The JSR 116 standard for SIP applications states in section 2.4 that multiple applications may be invoked for the same SIP request. The process of setting up applications to comply with this standard is called application composition.

Before you begin

About this task

Application composition requires that implementations use a cascaded services model. The cascaded services model requires that service applications triggered on the same host are triggered in sequence, as if the triggering occurred on different hosts. Therefore responses flow upstream and hit applications in the reverse order of the corresponding requests.

The JSR 116 standard does not specify how to implement application composition, thus there are many ways to comply with this standard. For WebSphere Application Server, composition of the application depends on the deployed application order, and on the order of mapping rules within the deployment descriptor of each application.

- For an initial incoming request, the SIP container tries each potential rule in order. When the container finds the n^{th} match, the container invokes the corresponding servlet.
- If the servlet must proxy the request, the container scans the rules again to search for additional matches. When the container finds the $(n+1)th$ match, the container invokes the corresponding servlet.
- Any servlet in the same application as the previously invoked servlet is excluded from the matching process. No servlet can be invoked twice for the same SIP request.

You can specify load on start-up priority. The `<load-on-startup>` in the sip.xml defines the order in which servlets are initialized on startup. If this value is lower than zero, the servlets are initialized when the first request is matched to them according to matching rule and composition order. Zero is a legitimate weight for startup initialization order. If this tag does not exist or if it contains a negative value, the servlet does not initialize at startup.

You should also add `<load-on-startup>` to the same tag in the web.xml if you are changing it manually. It is the WebContainer that loads servlets (and siplets), and it looks only at the web.xml. When deploying a SAR, only the sip.xml needs to be changed. The web.xml is automatically constructed correctly after deployment.

The load-on-startup tag embedded in the SIP deployment descriptor tag for a servlet dictates the order that the application is loaded on start up of the server. It does not dictate the order that an application gets called when the application is a member of an application composition chain that matches rules to process a new message coming in.

The starting weight for applications and their modules is specified in the deployment.xml file. The order in which modules pickup requests on composition is evaluated by applications weight first and then modules weight. The following steps can be completed in any order to specify applications weight or modules weight from the administrative console.

1. To specify the applications (EARs) weight, expand **Enterprise Applications** → ***applicationName*** → **Startup Behavior** and set the startup order.
2. To specify the modules (WARs) weight, expand **Enterprise Applications** → ***applicationName*** → **Manage Modules** and set the starting weight.
3. Restart the changed applications.

Example

Note:

```
sip.xml

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE sip-app
PUBLIC "-//Java Community Process//DTD SIP Application 1.0//EN"
"http://www.jcp.org/dtd/sip-app_1_0.dtd">
<sip-app>
 <display-name>SIPSampleProxy</display-name>

 <servlet>
  <servlet-name>SIPSampleProxy</servlet-name>
  <servlet-class>sipes.test.container.proxy.SIPSampleProxy</servlet-class>
  <load-on-startup>1</load-on-startup>
 </servlet>

 <servlet-mapping>
  <servlet-name>SIPSampleProxy</servlet-name>
  <pattern>
   <equal>
    <var>request.uri.user</var>
    <value>SIPSampleProxy</value>
   </equal>
  </pattern>
 </servlet-mapping>

 <proxy-config>
  <sequential-search-timeout>1000</sequential-search-timeout>
 </proxy-config>
 <session-config>
  <session-timeout>12</session-timeout>
 </session-config>
</sip-app>

web.xml

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE web-app PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN" "http://java.sun.com/dtd/web-app_2_3.dtd">
<web-app id="WebApp">
 <display-name>SIPSampleProxy</display-name>
 <servlet>
  <servlet-name>SIPSampleProxy</servlet-name>
  <display-name>SIPSampleProxy</display-name>
  <servlet-class>sipes.test.container.proxy.SIPSampleProxy</servlet-class>
 </servlet>
 <servlet-mapping>
  <servlet-name>SIPSampleProxy</servlet-name>
  <url-pattern>/SIPSampleProxy</url-pattern>
 </servlet-mapping>
 <welcome-file-list>
  <welcome-file>index.html</welcome-file>
  <welcome-file>index.htm</welcome-file>
  <welcome-file>index.jsp</welcome-file>
  <welcome-file>default.html</welcome-file>
  <welcome-file>default.htm</welcome-file>
  <welcome-file>default.jsp</welcome-file>
 </welcome-file-list>
</web-app>
```

Note:

The following example is for a standalone server.

```
deployment.xml

<?xml version="1.0" encoding="UTF-8" ?>
- <appdeployment:Deployment xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI"
xmlns:appdeployment="http://www.ibm.com/websphere/appserver/schemas/5.0/appdeployment.xmi" xmi:id="Deployment_1137951186883">
- <deployedObject xmi:type="appdeployment:ApplicationDeployment" xmi:id="ApplicationDeployment_1137951186883"
deploymentId="0" startingWeight="1" binariesURL="$(APP_INSTALL_ROOT)/OrangeNode08Cell/SipContainerTestSuite.ear"
useMetadataFromBinaries="false" enableDistribution="true" createMBeansForResources="true" reloadEnabled="false"
appContextIDForSecurity="href:OrangeNode08Cell/SipContainerTestSuite"
filePermission=".*\.dll=755#.*\.so=755#.*\.a=755#.*\.sl=755" allowDispatchRemoteInclude="false"
allowServiceRemoteInclude="false">
  <targetMappings xmi:id="DeploymentTargetMapping_1137951186883" enable="true" target="ServerTarget_1137951186883" />
  <classloader xmi:id="Classloader_1137951186883" mode="PARENT_FIRST" />
- <modules xmi:type="appdeployment:WebModuleDeployment" xmi:id="WebModuleDeployment_1137951186883"
deploymentId="1" startingWeight="10000" uri="sipunit.war">
  <targetMappings xmi:id="DeploymentTargetMapping_1137951186884" target="ServerTarget_1137951186883" />
  <classloader xmi:id="Classloader_1137951186884" /> </modules>
  <properties xmi:id="Property_1137951186883" name="validateinstall" value="warn" /> </deployedObject>
  <deploymentTargets xmi:type="appdeployment:ServerTarget" xmi:id="ServerTarget_1137951186883"
name="server1" nodeName="OrangeNode10" /> </appdeployment:Deployment>
```

SIP servlets

This topic describes SIP servlets.

The SIP Servlet 1.0 specification (JSR 116) is standardized through Java Specification Request (JSR) 116. The idea behind the specification is to provide a Java application programming interface (API) similar to HTTP servlets, which provides an easy-to-use SIP programming model. Like the popular HTTP servlet programming model, some flexibility is limited to optimize ease-of-use and time-to-value.

However, the SIP Servlet API is different in many ways from HTTP servlets because the protocol is so different. While SIP is a request-response protocol, there is not necessarily only one response to every one request. This complexity and a need for a high performing solution meant that it was easier to make the SIP servlets natively asynchronous. Also, unlike HTTP servlets, the programming model for SIP servlets sought to make client requests easy to create alongside the other logic being written because many applications act as a client or proxy to other servers or proxies.

SipServlet requests

Like HTTP servlets, each SIP servlet extends a base javax.servlet.sip.SipServlet class. All messages come in through the service method, which you can extend. However, because there is not a one-to-one mapping of requests to responses in SIP, the suggested practice is to extend the doRequest or doResponse methods instead. When extending the doRequest or doResponse methods, it is important to call the extended method for the processing to complete.

Each request method, which the specification must support, has a doxxx method just like HTTP. In HTTP, methods such as doGet and doPost exist for GET and POST requests. In SIP, doInvite, doAck, doOptions, doBye, doCancel, doRegister, doSubscribe, doNotify, doMessage, doInfo, and doPrack methods exist for each SIP request method.

Unlike an HTTP servlet, SIP servlets have methods for each of the response types that are supported. So, SIP servlets include the doProvisionalResponse, doSuccessResponse, doRedirectResponse, and doErrorResponse responses. Specifically, the provisional responses (1xx responses) are used to indicate status, the success responses (2xx responses) are used to indicate a successful completion of the transaction, the redirect responses (3xx responses) are used to redirect the client to a moved resource or entity, and the error responses (4xx, 5xx, and 6xx responses) are used to indicate a failure or a specific error condition. These types of response messages are similar to HTTP, but because the SIP Servlet programming model includes a client programming model, it is necessary to have responses handled programmatically as well.

Clarifications of JSR 116

JSR 289 has made some clarifications to JSR 116, as follows:

- JSR 289 Section 4.1.3: Contact Header Field
- JSR 289 Section 5.2: Implicit Transaction State
- JSR 289 Section 5.8: Accessibility of SIP Servlet Messages

SIP SipServletRequest and SipServletResponse classes

The SipServletRequest and SipServletResponse classes are similar to the HttpServletRequest and HttpServletResponse classes.

SipServletRequest and SipServletResponse classes

Each class gives you the capability to access the headers in the SIP message and manipulate them. Because of the asynchronous nature of the requests and responses, this class is also the place to create new responses for the requests. When you extend the doInvite method, only the SipServletRequest class is passed to the method. To send a response to the client, you must call the createResponse method on the Request object to create a response. For example:

```
protected void doInvite(SipServletRequest req) throws
 javax.servlet.ServletException, java.io.IOException {

 //send back a provisional Trying response
 SipServletResponse resp = req.createResponse(100);
resp.send();
```

Because of their asynchronous nature, SIP servlets can seem complicated. However, something as simple as the previous code sample sends a response to a client.

Here is a more complex example of a SIP servlet. With the following method included in a SIP servlet, the servlet blocks all of the calls that do not come from the example.com domain.

```
protected void doInvite(SipServletRequest req) throws
 javax.servlet.ServletException, java.io.IOException {

//check to make sure that the URI is a SIP URI
      if (req.getFrom().getURI().isSipURI()){
                 SipURI uri = (SipURI)req.getFrom.getURI();
                 if (!uri.getHost().equals("example.com")) {
                    //send forbidden response for calls outside domain
                    req.createResponse(SipServletResponse.SC_FORBIDDEN).send();
                    return;
                 }
              }
              //proxy all other requests on to their original destination
              req.getProxy().proxyTo(req.getRequestURI);
        }
```

SIP SipSession and SipApplicationSession classes

Possibly the most complex portions of the SIP Servlet 1.0 specification are the SipSession and SipApplicationSession classes.

SIP SipSession and SipApplicationSession classes

Both of these classes have some useful purposes and can act as the primary place to store data in applications that are designed for distributed or highly available environments.

The SipSession class is the best representative of a specific point-to-point communication between two entities and is the closest to the HttpSession object. Because historically no proxying or forking existed for the HTTP request in HTTP servlets, the need for something higher than a single point-to-point session did not exist. However, even HTTP users can see the growing need for this type of function since portlets began essentially forking HTTP requests. The SIP users expect the proxying and forking activities that require multiple layers of SIP session management. The SipSession class is the lowest point-to-point layer.

The SipApplicationSession class represents the higher layer of SIP session management. One SipApplicationSession class can own one or more SipSession objects. However, each SipSession class can be related to one SipSession object only. The SipApplicationSession class also supports the attachment of any number of other protocol sessions. Currently, only HTTP sessions are supported by any implementations. The SipApplicationSession class has a getSessions method, which takes the requested protocol type as an argument.

You might find it useful for many applications to combine HTTP and SIP. For example, you might use this approach to tie together HTTP and SIP sessions to monitor a phone call or to start a phone call through a rich HTTP graphical user interface.

Example: SIP servlet simple proxy

This is a servlet example of a simple proxy.

Simple proxy

```
import java.io.IOException;

import javax.servlet.Servlet;
import javax.servlet.ServletException;
import javax.servlet.sip.Proxy;
import javax.servlet.sip.SipFactory;
import javax.servlet.sip.SipServlet;
import javax.servlet.sip.SipServletRequest;
import javax.servlet.sip.SipServletResponse;
import javax.servlet.sip.SipSession;
import javax.servlet.sip.SipURI;
import javax.servlet.sip.URI;


public class SimpleProxy extends SipServlet implements Servlet {


 final static  private String  SHUTDOWN_KEY = new String("shutdown");
 final static  private String  STATE_KEY = new String("state");
 final static  private int   INVITE_RECEIVED = 1;

 /* (non-Java-doc)
  * @see javax.servlet.sip.SipServlet#SipServlet()
  */
 public SimpleProxy() {
  super();
 }

 /* (non-Javadoc)
  * @see javax.servlet.sip.SipServlet#doInvite(javax.servlet.sip.SipServletRequest)
  */
 protected void doInvite(SipServletRequest request) throws ServletException,
   IOException {

  //log("SimpleProxy: doInvite: TOP");

  try {
   if (request.isInitial() == true)
   {
    // This should cause the sip session to be created. This sample only uses the session on receiving
    // a BYE but the Tivoli performance viewer can be used to track the creation of calls by viewing the
    // active session count.
    Integer state = new Integer(INVITE_RECEIVED);
    SipSession session = request.getSession();
    session.setAttribute(STATE_KEY, state);
```

```
            log("SimpleProxy: doInvite: setting attribute");

   Proxy proxy = request.getProxy();

   SipFactory sipFactory = (SipFactory) getServletContext().getAttribute(SIP_FACTORY);
         if (sipFactory == null) {
             throw new ServletException("No SipFactory in context");
         }

         String callingNumber = request.getTo().toString();
         if (callingNumber != null)
         {
             String destStr = format_lookup(callingNumber);
             URI dest = sipFactory.createURI(destStr);

             //log("SimpleProxy: doInvite: Proxying to dest URI = " + dest.toString());

          if (((SipURI)request.getRequestURI()).getTransportParam() != null)
           ((SipURI)dest).setTransportParam(((SipURI)request.getRequestURI()).getTransportParam());

          proxy.setRecordRoute(true);
    proxy.proxyTo(dest);
         }
   else
   {
             //log("SimpleProxy: doInvite: Request is invalid. Did not contain a To: field.");
    SipServletResponse sipresponse = request.createResponse(400);
    sipresponse.send();
   }
  }
  else
  {
         //log("SimpleProxy: doInvite: target refresh, let container handle invite");
   super.doInvite(request);
  }
 }
 catch (Exception e){
  e.printStackTrace();
 }
}

/* (non-Javadoc)
 * @see javax.servlet.sip.SipServlet#doResponse(javax.servlet.sip.SipServletResponse)
 */
protected void doResponse(SipServletResponse response) throws ServletException,
  IOException {
  super.doResponse(response);

  // Example of using the session object to store session state.
 SipSession session = response.getSession();
 if (session.getAttribute(SHUTDOWN_KEY) != null)
 {
  //log("SimpleProxy: doResponse: invalidating session");
  session.invalidate();
 }
}

/* (non-Javadoc)
 * @see javax.servlet.sip.SipServlet#doBye(javax.servlet.sip.SipServletRequest)
 */
protected void doBye(SipServletRequest request) throws ServletException,
  IOException {

 SipSession session = request.getSession();
 session.setAttribute(SHUTDOWN_KEY, new Boolean(true));

       //log("SimpleProxy: doBye: invalidate session when responses is received.");
  super.doBye(request);
 }

 protected String format_lookup(String toFormat){
  int start_index = toFormat.indexOf('<') + 1;
  int end_index = toFormat.indexOf('>');

  if(start_index == 0){
   //don't worry about it
  }
  if(end_index == -1){
   end_index = toFormat.length();
  }

  return toFormat.substring(start_index, end_index);
 }
}
```

Example: SIP servlet SendOnServlet class

The SendOnServlet class is a simple SIP servlet that would perform the basic function of being called on each INVITE and sending the request on from there.

SendOnServlet class

Function could easily be inserted to log this invite request or reject the INVITE based on some specific criteria.

```
package com.example;
import java.io.IOException;
import javax.servlet.sip.*;
import java.servlet.ServletException;
public class SendOnServlet extends SipServlet {
  public void doInvite(SipServletRequest req)
       throws ServletException, java.io.IOException {
     //send on the request
     req.getProxy().proxyTo(req.getRequestURI);
  }
}
```

The doInvite method could be altered to do something such as reject the invite for some specific criteria simply. In the example doInvite method below, all requests from domains outside of example.com will be rejected with a Forbidden response.

```
   public void doInvite(SipServletRequest req)
throws ServletException, java.io.IOException {
if (req.getFrom().getURI().isSipURI()){
    SipURI uri = (SipURI)req.getFrom.getURI();
    if (!uri.getHost().equals("example.com")) {
  //send forbidden response for calls outside domain
req.createResponse(SipServletResponse.SC_FORBIDDEN, "Calls outside example.com not accepted").send();
return;
    }
 }
 //proxy all other requests on to their original destination
 req.getProxy().proxyTo(req.getRequestURI());
   }


SendOnServlet deployment descriptor:
<sip-app>
    <display-name>Send-on Servlet</display-name>
    <servlet>
        <servlet-name>SendOnServlet</servlet-name>
        <servlet-class>com.example.SendOnServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>SendOnServlet</servlet-name>
        <pattern>
              <equal>
                <var>request.method</var>
                <value>INVITE</value>
              </equal>
        </pattern>
    </servlet-mapping>
</sip-app>
```

Example: SIP servlet Proxy servlet class

Proxy servlet class

After the initial INVITE, this application will be called on every subsequent SIP message. For each Request and Response, this class will simply print out the action and who it is to or from.

```
package com.example;
import java.io.IOException;
import javax.servlet.sip.*;
import java.servlet.ServletException;
public class ProxyServlet extends SipServlet {
  public void doInvite(SipServletRequest req)
       throws ServletException, java.io.IOException {
     //get the Proxy
     Proxy p=req.getProxy();
     //turn on supervised mode so that all events come through us
     //The default on this is true but it is set to emphasize the function.
     p.setSupervised(true);
     //set record route so we see the ACK, BYE, and OK
     p.setRecordRoute(true);
     //proxy on the request
     p.proxyTo(req.getRequestURI());
  }
public void doRequest(SipServletRequest req)
     throws ServletException, java.io.IOException {
   System.out.println(req.getMethod()+" Request from "+req.getFrom().getDisplayName());
   super.doRequest(req);
}
public void doResponse(SipServletResponse resp)
     throws ServletException, java.io.IOException {
   System.out.println(resp.getReasonPhrase()+" Response from "+resp.getTo().getDisplayName());
   super.doResponse(resp);
}
}

Proxy deployment descriptor
<sip-app>
    <display-name>ProxyServlet</display-name>
    <servlet>
        <servlet-name>ProxyServlet</servlet-name>
        <servlet-class>com.example.ProxyServlet</servlet-class>
    </servlet>

    <servlet-mapping>
        <servlet-name>ProxyServlet</servlet-name>
        <pattern>
              <equal>
                <var>request.method</var>
                <value>INVITE</value>
              </equal>
        </pattern>
    </servlet-mapping>
</sip-app>
```

Deploying SIP applications

Use the administrative console to customize your Session Initiation Protocol (SIP) application installation

About this task

When you deploy a Session Initiation Protocol (SIP) application, you can perform various tasks such as installing, starting, stopping, upgrading, and uninstalling the application.

SIP applications are installed as Java Platform, Enterprise Edition (Java EE) applications. You can deploy a SIP application from a graphical interface or from a command line.

Deploying SIP applications through the console

You can deploy a Session Initiation Protocol (SIP) application through the administrative console.

Before you begin

SIP applications are deployed as Java 2 Platform Enterprise Edition (J2EE) applications. In order to process requests, a virtual host must be defined when deploying the SIP application. If there is no virtual host defined for the configured SIP container listen port, the installed application will be inaccessible.

1. Open the administrative console.

   In a browser, go to URL `http://`*`hostname`*`:9090/admin`, where *hostname* is the name of the host computer. Enter the appropriate login information, and click **OK**.
2. In the left frame click **Applications** → **Install New Application**.
3. Browse and select a SAR file. Specify the context root, beginning with a slash (/), in the **Context Root** field. For example, if your application is named ThisApplication, type `/ThisApplication`.
4. Click **Next** (under the **Context Root** field not beside the WebSphere Status title). If the SAR file has been assembled correctly, the screen will still have the title “Preparing for the application installation”, but the content will change. If an error message appears, check the contents of the SAR file; in particular, verify the web.xml file contents, and try to reload the SAR file.
5. Click **Next**. If you see a screen indicating “Application Security Warnings”, click **Continue**.
6. The **Install New Application** screen should appear with “Step 1: Select application options” highlighted. Select the options you need and click **Next**.
7. “Step 2: Map modules to servers” should appear highlighted now. You can choose the cluster or server where you want to install the application’s modules.
   - If you are installing the application in a stand-alone system, click **Next**.
   - If you are installing the application in a clustered system, select **WebSphere:cell=*cellname*,cluster=*cluster_name*** in the **Clusters and Servers** field, select the check box beside the Web module that you want to install, and click **Apply** and **Next**.
8. Now “Step 3: Map virtual hosts for Web modules” should appear highlighted. To the right of the application name there should be a drop-down labeled **Virtual Host**.
   - If you are installing the application in a standalone system, set the value of the drop-down to **default_host**, and click **Next**.
   - If you are installing the application in a clustered system, set the value of the drop-down to the name of the virtual host that was chosen during setup, and click **Next**.

   Note: You must define a virtual host for your configured SIP container listen port or else you will not be able to access the application.
9. You should now see “Step 4: Summary” highlighted. In the right panel you will see a **Summary of installation options** table that details your selected options and their values. If you need to change an option, click **Previous** to return to the section where you can make your change. Click **Finish** to

install the application with your settings. The screen should display, `Application` *appname*`_sar installed successfully`, where *appname* is the name of the application.

10. Click the **Save to Master Configuration** link. A Save to Master Configuration window appears.
11. In the Save to Master Configuration window, click **Save**. The application has now been saved in the current configuration.
12. To confirm that the installation succeeded, in the left frame click **Applications** → **Enterprise Applications**. The newly installed application should appear in the list of installed applications as *appname*_sar.
13. To start the application so that it can service SIP requests, check the box beside *appname*_sar, and click **Start**. You might also want to look at the logs for a successful startup message.

Results

The application can service SIP requests now.

Deploying SIP applications through scripting

You can deploy a Session Initiation Protocol (SIP) application not only from the GUI but also from the command line.

- Launch a scripting client. For more information, see AdminApp object for scripted administration.
- List applications.
- Install standalone archive files. For more information about installation, see Installation options for the AdminApp object.
- Edit application configurations.
- Uninstall applications.

Securing SIP applications

You can apply digest authentication and Trust Association Interceptor (TAI) for a SIP application by applying Lightweight Directory Access Protocol (LDAP) security to the application.

Before you begin

Before you can apply security, you must first deploy an application that has been developed to support security (with the web.xml file configured for security) and roles. The following software must also be installed:

1. Install a supported LDAP server. For a list of supported LDAP servers, see the IBM Web site for WebSphere Application Server supported hardware, software, and APIs.
2. Set up and activate Lightweight Third Party Authentication. For more information, see theConfiguring the Lightweight Third Party Authentication mechanism topic.

About this task

To apply LDAP security to a SIP application, click **Applications** → **Enterprise Applications** → ***applicationName*** and complete the following steps:

1. Click **Detail Properties** → **Security role to user/group mapping**.
2. Check **All Authenticated**.
3. Save all changes.
4. Restart the server.

Configuring security for the SIP container

This section provides instructions specific to security for the SIP container.

Before you begin

Before you can configure security for your SIP container, you will need to:

1. Set up and activate Lightweight Third Party Authentication. For more information, see the Lightweight Third Party Authentication section.
2. Install a supported LDAP server.

You may also need to:

- Adjust key group settings. Refer to Lightweight Third Party Authentication key sets and key set groups for LTPA key information.
- Establish and configure Trust Association Interceptor (TAI) settings. Refer to Trust association interceptor settings.

About this task

You must know the name of the key set group and the management scope where the key set group is defined in order to activate and secure LTPA with keys. Refer to Activating Lightweight Third Party Authentication key versions for the setup and activation procedures.

To configure security based on the Lightweight Directory Access Protocol (LDAP), you can configure digest authentication for your supported LDAP server.

- To configure digest authentication and TAI on WebSphere Application Server for Tivoli, select "Configuring digest authentication and TAI for SIP."
- To configure digest authentication on WebSphere Application Server for Oracle Internet Directory, select "Configuring digest authentication for Oracle Internet Directory" on page 156.

To define an LDAP connection between WebSphere Application Server and LDAP, use the security wizard. It can also be defined by selecting it from available realms and defining the proper connection properties to connect LDAP.

To set up a TAI, you must specify the trust information for any reverse security proxy servers. See Trust association interceptor settings to configure TAI settings.

Configuring digest authentication and TAI for SIP

You can configure digest authentication and Trust Association Interceptor (TAI) for the Session Initiation Protocol (SIP).

Before you begin

Before you can configure digest authentication and TAI, you must either install a supported LDAP server, or configure digest TAI to work without LDAP.

To configure digest TAI to work without LDAP, complete these steps:

1. Create a class that implements the interface: com.ibm.ws.sip.security.digest.DigestPasswordServer.
2. In the administrative console, click **Global security > Digest authentication > Custom Properties > New**, and enter `DigestPasswordServerClass` in the **Name** field, and the name of the class that you created in the **Value** field.
3. Ensure that all users that implement the impl class are declared in the user registry configured for WebSphere Application Server security.

LDAP servers automatically provide password support. Unless you enable the LDAP server to use hashed values, the LDAP server stores user passwords and then the request processing component uses these passwords to validate a request. Because this method of authentication exposes user passwords to potential internet theft, you should enable the use of hashed credentials to authenticate a request.

When you enable the use of hashed credentials, the LDAP server stores a hash value for the user, password and realm information. The SIP container then requests this hash value from the LDAP server instead of asking for a user password. This methodology protects the passwords even if the hash data is compromised through internet theft. However, this methodology has the following limitations:

- The LDAP attribute must store a byte value or a string value. Other attribute types are not supported.
- All of your applications must share the same realm, or you must define a different attribute for each realm.
- The hash function might be different than MD5. In this situation, the SIP container sends a algorithm that is different from the calculated value for the attribute. When this situation occurs, user authentication might fail even if the user provided the proper credentials.

To enable the LDAP server to use hashed credentials, you must define the following two custom properties:

- hashedCredentials=*value*, where *value* is the name of LDAP attribute which stores the hash value for user, password, realm
- hashedCredentialsRealms=*value*, where *value* is the realm, on which the hashed value is calculated.

About this task

The SIP container supports digest authentication. When this type of authentication is used, the client does not send a clear text password to the server. Instead, SIP authenticates each request using user data from LDAP. Typically, a component that uses LDAP for authentication, verifies that the response that the client provides equals the response that the component calculates using LDAP data, the component authorizes the request. However,

Howto define: One should d

Complete the following procedure to configure digest authentication and TAI for the SIP container.

1. In the administrative console, click **Security** → **Global security** → **Authentication mechanisms** to verify that **Lightweight Third Party Authentication (LTPA)** is configured for use on your server.

   In the **Configuration** tab on the **Authentication mechanisms and expiration** page you should see the **Password** field already filled in.
2. Click **Security** → **Global security**.
   a. Under **Authentication**, expand **Web security** and click **Trust association**.
   b. On the **Configuration** tab, in the General properties section, verify that the **Enable trust association** box is selected, and then click **Apply**.
3. On the **Interceptors** page of the administration console look for `com.ibm.ws.sip.security.digest.DigestTAI` in the **Interceptor class name** list.
   a. If this class name in not present, click **New** to open the Configuration tab and enter `com.ibm.ws.sip.security.digest.DigestTAI` in the **Interceptor class name** field, and then click **Apply**.
   b. If this interceptor class is present, click **com.ibm.ws.sip.security.digest.DigestTAI** → **Custom Properties** to set up a realm in digest authentication.
   c. Click **OK**.
4. Click **Security** → **Global security** → **Authentication mechanisms and expiration**, and then click the **Configuration** tab.
   a. In the **Key generation** section, click **Generate keys**. You do not have to import or export the key.
   b. In the Cross-cell Single Sign-on section, specify values in the **Password** fields and the **Internal server ID** field.
   c. Click **OK**.
5. Click **Security** → **Global security**.

   a. If the box **Use Java 2 security to restrict application access to local resources** is selected, click to deselect it.
   b. In the **User account repository** section of the page, select your LDAP registry from the **Available realm definitions** list.
   c. Click **Set as current**, and then click**Apply**.
6. Save all changes.
7. Restart the server.
8. Verify that the following message appears in the SystemOut.log file after the server restarts:

   ```
   SECJ0121I: Trust Association Init class com.ibm.ws.sip.security.digest.DigestTAI loaded successfully
   ```

   If this message does not appear in the log file, digest authentication is not active

Configuring digest authentication for Oracle Internet Directory

You can configure digest authentication for Oracle Internet Directory, an implementation of the Lightweight Directory Access Protocol (LDAP) that uses the Oracle database as a repository for directory entries.

Before you begin

To configure digest authentication for Oracle Internet Directory, you will need to:

- Install Oracle Internet Directory version 9.0.2.
- Set up and activate Lightweight Third Party Authentication. For more information, see the Lightweight Third Party Authentication section.

About this task

Complete the following procedure to configure digest authentication for Oracle Internet Directory on WebSphere Application Server:

1. To set up digest authentication, verify that **Lightweight Third Party Authentication (LTPA)** is configured for use on your server by selecting **Security** → **Global security** → **Authentication mechanisms**. In the **Configuration** tab on the **Authentication mechanisms and expiration** page you should see the **Password** field already filled in.
2. In the administrative console, click **Security** → **Global security**.
   a. Under **Authentication**, expand **Web security** and click on **Trust association**.
   b. On the **Configuration** tab, under **General properties**, make sure the **Enable trust association** box is checked. Then click **Apply**.
3. On the **Interceptors** page of the administration console look for `com.ibm.ws.sip.security.digest.DigestTAI` in the **Interceptor class name** list:
   a. If this class name in not present, click **New** to open the Configuration tab and enter `com.ibm.ws.sip.security.digest.DigestTAI` in the **Interceptor class name** field and click **Apply**. Then proceed to the following steps.
   b. If this interceptor class is present, you may set up custom properties for it. To do this, click **com.ibm.ws.sip.security.digest.DigestTAI** → **Custom Properties**:
   c. Click **OK**.
4. Navigate through **Security** → **Global security** → **Authentication mechanisms and expiration** to the **Configuration** tab.
   a. In the **Key generation** section, click **Generate Keys**. (No import or export of the key is necessary.)
   b. Under the Cross-cell single sign-on section fill in the **Password** fields.
   c. Fill in the **Internal server ID** field.
   d. Click **OK**.
5. Click to **Security** → **Global security**.

a. If the box **Use Java 2 security to restrict application access to local resources** is checked, then Java 2 security is enabled. Click the box if you want to disable Java 2 security.

b. In the **User account repository** section of the page, select your LDAP registry from the **Available realm definitions** drop-down box.

c. Click **Set as current** and then click**Apply**.

6. Save all changes.
7. Restart the server.
8. Be sure you see the following message appear in the SystemOut.log after the server has restarted:

```
SECJ0121I: Trust Association Init class
 com.ibm.ws.sip.security.digest.DigestTAI loaded successfully
```

   If this message does not appear in the log, digest authentication has not been activated.

Tracing a SIP container

You can trace a Session Initiation Protocol (SIP) container, starting either immediately or after the next server startup. This tracing writes a record of SIP events to a log file.

About this task

Follow these steps to start tracing a SIP container:

1. Open the administrative console. For more information about the console, read the Using the administrative console chapter of the *Administering applications and their environment* PDF book.
2. In the administrative console, click **Troubleshooting** → **Logs and trace**.
3. Select the name of the server for the SIP container.
4. From the General Properties section, click **Diagnostic Trace Service**.
5. Under the Additional Properties section, click **Change Log Detail Levels**
6. Select one of the following options:

Option	Description
Configuration	To start tracing after the next server startup
Runtime	To start tracing immediately

7. Replace the content of the trace specification with the following code: `com.ibm.ws.sip.*=all=enabled`.

   Note: If you want monitor only specific pieces of SIP containers, expand the **com.ibm.ws.sip** section and select the individual items you wish to trace.
8. Make sure that the **Enable trace with following specification** check box is checked.
9. Click **Apply** → **Save**.

What to do next

When the changes take effect (refer to step 6 above), SIP-level tracing messages appear in *WASProductDir*/logs/*serverName*/trace.log, where *WASProductDir* is the fully qualified path name of the directory in which the product is installed and *serverName* is the name of the specific instance of the application server that is running the SIP container to be traced. These messages include application load events as well as SIP request and response parsing and SIP servlet invocation.

Troubleshooting SIP applications

Use this page to troubleshoot SIP applications.

About this task

SIP container troubleshooting basics

- The Average CPU usage of the system should go no higher than 60%-70%.
- The container should use no more than 70% of the allocated VM heap size. Be sure that the system has enough physical memory to accommodate the VM heap size. Call loads and session timeouts will have a big affect on heap usage.
- The maximum garbage collection (GC) time of the VM on which the container is running should not exceed 500 ms and the average should be less than 400 ms. Verbose GC can be used to measure this and the PMI viewer can be used to view GC times, heap usage, active sessions, etc., in graphical form.

Initial troubleshooting checklist:

- Check the listening ports in the configuration.
- Use netstat –an to see listening ports.
- Check to see if virtual hosts are defined
- Check to see if host aliases are defined.
- Is an application installed? Is it started?
- For a proxy configuration: Is a default cluster configured? If proxy and server are on the machine, is there a port conflict?

Results

SIP container symptoms and solutions

If the problem is not resolved, check for specific symptoms.

- **Symptom:** Lots of retransmissions, CPU periodically drops to zero.

  Solution: This is typically a DNS issue caused by Reverse DNS lookups and can be confirmed using a tool like Ethereal. If you do a network capture and send lots of DNS queries that contain an IP address and get back a host name in the response, this could be the problem. Make sure that nscd is running if you are on HP or other platforms that require name service caching. (Windows does not require this.) Another solution is to add host names to the /etc/hosts file.
- **Symptom:** Lots of retransmissions, CPU periodically spikes to 100%.

  Solution: This is typically due to garbage collection and can be verified by turning on verbose GC (accessible on the admin console) and looking at the length of the GC cycles. The solution here is to enable Generational Garbage Collection by setting the JVM optional args to -Xgcpolicy:gencon.
- **Symptom:** Lots of retransmissions, CPU spikes to 100% for long periods of time and Generational Garbage Collection is enabled.

  Solution: This is typically due to SIP session objects either not being invalidated or not timing out over a long period of time. One solution is to set the session timeout value in the sip.xml of the application to a smaller value. The most efficient way to handle this is for the application to call invalidate on the session when the dialog completes (i.e. after receiving a BYE). The following entry in the SystemOut.log file will indicate the session timeout value is for each application installed on the container:

```
SipXMLParser  3 SipXMLParser getAppSessionTTL Setting Expiration time: 7 Minutes, For  App: TCK back-to-back user agent"
```

- **Symptom:** Lots of “480 Service Not Available” messages received from the container when sending new INVITE messages to the SIP container. You will also likely see the following message show up in the SystemOut.log when the server is in this state: ″LoadManager E LoadManager warn.server.oveloaded″.

  Solution: This is typically due to one of the SIP container configurable metrics being exceeded. This includes the ″Maximum Application Sessions″ value and the ″Maximum messages per averaging period″ value. The solution is to adjust these values higher.

- **Symptom:** Lots of resends and calls are not completing accompanied by OutOfMemory exceptions in the SystemErr.log.

  Solution: This usually means that the VM heap size associated with your container is not large enough and should be adjusted upwards. You can adjust this value from the admin console.
- **Symptom:** You receive a "503 Service Unavailable" when sending a SIP request to a SIP proxy.

  Solution: This usually means there is no default cluster (or cluster routing rule that matches the message) set up at the proxy. This can also happen when the SIP proxy is configured well but the backend SIP containers are stopped or have crashed.
- **Symptom:** You receive a "404 Not Found" when sending a SIP request to a SIP proxy.

  Solution: This usually means there is no virtual host set up for the containers that reside in the default cluster. It could also mean that the servers in the proxy's default cluster do not contain a SIP application or that the message does not match one of the applications installed in the default cluster.
- **Symptom:** An ″out of memory″ type behavior is occurring.

  Solution: This may be due to the maximum heap size being set too low. SIP applications can consume a significant amount of memory because the sessions exist for a long call hold time. The maximum heap size of 512 MB does not provide sufficient memory for the SIP traffic workload. Set the maximum heap size for SIP applications to the minimum recommended value of 768 MB or higher.
- **Symptom:** You receive a "403 Forbidden" when sending a SIP request to a SIP container.

  Solution: This usually means there is no appropriate SIP application found to handle the received SIP request (no match rule that matched the message).

SIP timer summary

Request for Comments (RFC) 3261, "SIP: Session Initiation Protocol," specifies various timers that SIP uses.

Table 8 on page 103 summarizes for each SIP timer the default value, the section of RFC 3261 that describes the timer, and the meaning of the timer.

Table 11. Summary of SIP timers

Timer	Default value	Section	Meaning
T1	500 ms	17.1.1.1	Round-trip time (RTT) estimate
T2	4 sec.	17.1.2.2	Maximum retransmission interval for non-INVITE requests and INVITE responses
T4	5 sec.	17.1.2.2	Maximum duration that a message can remain in the network
Timer A	initially T1	17.1.1.2	INVITE request retransmission interval, for UDP only
Timer B	64*T1	17.1.1.2	INVITE transaction timeout timer
Timer D	> 32 sec. for UDP 0 sec. for TCP and SCTP	17.1.1.2	Wait time for response retransmissions
Timer E	initially T1	17.1.2.2	Non-INVITE request retransmission interval, UDP only
Timer F	64*T1	17.1.2.2	Non-INVITE transaction timeout timer
Timer G	initially T1	17.2.1	INVITE response retransmission interval
Timer H	64*T1	17.2.1	Wait time for ACK receipt
Timer I	T4 for UDP 0 sec. for TCP and SCTP	17.2.1	Wait time for ACK retransmissions

Table 11. Summary of SIP timers (continued)

Timer	Default value	Section	Meaning
Timer J	64*T1 for UDP	17.2.2	Wait time for retransmissions of non-INVITE requests
	0 sec. for TCP and SCTP		
Timer K	T4 for UDP	17.1.2.2	Wait time for response retransmissions
	0 sec. for TCP and SCTP		

Appendix. Directory conventions

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

Default product locations (i5/OS)

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

app_client_root
: The default installation root directory for the Java EE WebSphere Application Client is the /QIBM/ProdData/WebSphere/AppClient/V7/client directory.

app_client_user_data_root
: The default Java EE WebSphere Application Client user data root is the /QIBM/UserData/WebSphere/AppClient/V7/client directory.

app_client_profile_root
: The default Java EE WebSphere Application Client profile root is the /QIBM/UserData/WebSphere/AppClient/V7/client/profiles/*profile_name* directory.

app_server_root
: The default installation root directory for WebSphere Application Server is the /QIBM/ProdData/WebSphere/AppServer/V7/Base directory.

cip_app_server_root
: The default installation root directory is the /QIBM/ProdData/WebSphere/AppServer/V7/Base/cip/*cip_uid* directory for a customized installation package (CIP) produced by the Installation Factory.

A CIP is a WebSphere Application Server product bundled with optional maintenance packages, an optional configuration archive, one or more optional enterprise archive files, and other optional files and scripts.

cip_profile_root
: The default profile root directory is the /QIBM/UserData/WebSphere/AppServer/V7/Base/cip/*cip_uid*/profiles/*profile_name* directory for a customized installation package (CIP) produced by the Installation Factory.

cip_user_data_root
: The default user data root directory is the /QIBM/UserData/WebSphere/AppServer/V7/Base/cip/*cip_uid* directory for a customized installation package (CIP) produced by the Installation Factory.

if_root
: This directory represents the root directory of the IBM WebSphere Installation Factory. Because you can download and unpack the Installation Factory to any directory on the file system to which you have write access, this directory's location varies by user. The Installation Factory is an Eclipse-based tool which creates installation packages for installing WebSphere Application Server in a reliable and repeatable way, tailored to your specific needs.

iip_root
: This directory represents the root directory of an *integrated installation package* (IIP) produced by the IBM WebSphere Installation Factory. Because you can create and save an IIP to any directory on the file system to which you have write access, this directory's location varies by user. An IIP is an aggregated installation package created with the Installation Factory that can include one or more generally available installation packages, one or more customized installation packages (CIPs), and other user-specified files and directories.

java_home
: The following directories are the root directories for all supported Java Virtual Machines (JVMs).

© IBM Corporation 2005, 2007

JVM	**Directory**
Classic JVM	/QIBM/ProdData/Java400/jdk6
32–bit IBM Technology for Java	/QOpenSys/QIBM/ProdData/JavaVM/jdk60/32bit
64–bit IBM Technology for Java	/QOpenSys/QIBM/ProdData/JavaVM/jdk60/64bit

plugins_profile_root
The default Web server plug-ins profile root is the /QIBM/UserData/WebSphere/Plugins/V7/webserver/profiles/*profile_name* directory.

plugins_root
The default installation root directory for Web server plug-ins is the /QIBM/ProdData/WebSphere/Plugins/V7/webserver directory.

plugins_user_data_root
The default Web server plug-ins user data root is the /QIBM/UserData/WebSphere/Plugins/V7/webserver directory.

product_library
product_lib
This is the product library for the installed product. The product library for each Version 7.0 installation on the system contains the program and service program objects (similar to .exe, .dll, .so objects) for the installed product. The product library name is QWAS7*x* (where *x* is A, B, C, and so on). The product library for the first WebSphere Application Server Version 7.0 product installed on the system is `QWAS7A`. The *app_server_root*/properties/product.properties file contains the value for the product library of the installation, `was.install.library`, and is located under the *app_server_root* directory.

profile_root
The default directory for a profile named *profile_name* for WebSphere Application Server is the /QIBM/UserData/WebSphere/AppServer/V7/Base/profiles/*profile_name* directory.

shared_product_library
The shared product library, which contains all of the objects shared by all installations on the system, is `QWAS7`. This library contains objects such as the product definition, the subsystem description, the job description, and the job queue.

updi_root
The default installation root directory for the Update Installer for WebSphere Software is the /QIBM/ProdData/WebSphere/UpdateInstaller/V7/updi directory.

user_data_root
The default user data directory for WebSphere Application Server is the /QIBM/UserData/WebSphere/AppServer/V7/Base directory.

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

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

Notices

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.

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

© Copyright IBM Corp. 2008

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 is a trademark of Sun Microsystems, Inc. in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

© Copyright IBM Corp. 2008