WebSphere® Application Server—Express for Distributed Platforms, Version 6.1

Setting up the application serving environment

Note

Before using this information, be sure to read the general information under “Notices” on page 309.

Compilation date: May 10, 2006

Contents

How to send your comments

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

- To send comments on articles in the WebSphere Application Server Information Center
  1. Display the article in your Web browser and scroll to the end of the article.
  2. Click on the **Feedback** link at the bottom of the article, and a separate window containing an e-mail form appears.
  3. Fill out the e-mail form as instructed, and click on **Submit feedback** .
- To send comments on PDF books, you can e-mail your comments to: **wasdoc@us.ibm.com** or fax them to 919-254-0206.

  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.

Chapter 1. Planning the installation

Some Web server topologies are commonly installed with the IBM WebSphere Application Server - Express product.

Use this topic to understand the capabilities of your WebSphere Application Server - Express product. Knowing what you can do with the product might influence how you install the product and other installable components on the product disc.

Topology diagrams and descriptions show how to create a given topology by showing what components to install for each topology.

Installation roadmap

Install the following components from the product disc to create a production environment:

- The WebSphere Application Server - Express product
- The IBM HTTP Server V6 (or another supported Web server from another source)
- A binary plug-in module for your Web server

Optionally, use the product disc to install an application client environment on a client machine. Java 2 Platform, Enterprise Edition (J2EE) client applications and thin client applications run on a client workstation. The client applications communicate with WebSphere Application Server on a remote server machine. A client application requires some elements of the Application Server on the client machine. However, instead of installing the full WebSphere Application Server - Express product, you can install the Application Client to provide a stand-alone client run-time environment.

Optionally, use the Rational Web Developer disc in the primary packet of discs to install a fully integrated development environment that includes an application server for development testing.

Installation features

Installation features in V6.x include:

Feature	Description
The WebSphere Application Server - Express product includes application server nodes..	Use the Profile Management tool and its Profile Management tool to create additional stand-alone application server nodes after installing the product. You do not have to reinstall the product to create a new profile. All profiles on a machine share the core product files. If you do install again to create a second set of core product files, there are two Profile Management tools - one for each set of core product files.
The product disc includes all of the installable components that are required to create an e-business environment.	Use the product disc to install the IBM HTTP Server, the Web server plug-ins, and the WebSphere Application Client components. Installation programs for each component exist within component directories.
Each installable component has its own installation program.	Use the launchpad to install any component on the product disc. Or install each component directly using the install command in the component directory.
The launchpad can install any installable product in the primary packet of compact discs.	The launchpad for the WebSphere Application Server - Express product can also install the Rational Web Developer on Windows and Linux (Intel) systems. The Rational Web Developer is on its own disc. Change discs to launch the installation.

Review topology diagrams for each of the following installable components to determine which topology best fits your needs. The diagrams and their accompanying procedures can serve as a roadmap for installing a similar topology.

This topic describes installation scenarios for the following installable components:

- WebSphere Application Server - Express
- Web server plug-ins
- Application Client

In addition to product installation diagrams for the installable components, this topic also links to a roadmap for using the Profile Management tool. The Profile Management tool creates run-time environments for application server processes.

Each of the following installation scenarios includes topology diagrams and associated installation steps. Each step links to a specific procedure for installing a component or to a description of a command or tool.

1. **Diagrams:** Review the installation scenarios for the WebSphere Application Server - Express product, as described in "Planning to install WebSphere Application Server - Express."
2. **Diagrams:** Review the installation scenarios for the Web server plug-ins for WebSphere Application Server, as described in "Planning to install Web server plug-ins" on page 7.
3. **Diagram:** Review the installation scenarios for the Application Client, as described in "Planning to install WebSphere Application Client" on page 9.
4. **Diagrams:** Review the installation scenarios for profile creation, as described in "Planning to create application server environments" on page 11.
5. **Optional:** Review interoperability and coexistence diagrams to know what is possible with Version 6.x.

   WebSphere Application Server V6.x can interoperate with your other e-business systems, including other versions of WebSphere Application Server. *Interoperability* provides a communication mechanism for WebSphere Application Server nodes that are at different versions, running on separate machines. *Coexistence* describes multiple versions or instances running on the same machine at the same time.

   Interoperability support enhances migration scenarios with more configuration options. Interoperating is often more convenient or practical during the migration of a configuration from an earlier WebSphere Application Server version to a later one. Some machines can have the earlier product version and other machines can have the later version. An environment of machines and application components at different software version levels can involve both interoperability and coexistence.

   It is often impractical, or even physically impossible, to migrate all of the machines and applications within an enterprise at the same time. Understanding multiversion interoperability and coexistence is therefore an essential part of a migration between version levels.
6. **Optional:** Consider performance when designing your network, as described in "Queuing network" on page 11.

Following this procedure results in reviewing installation scenarios to identify specific steps to follow when installing more than one component.

After determining an appropriate installation scenario, install the necessary components and configure the products for the system that you selected.

Planning to install WebSphere Application Server - Express

Consider common installation scenarios for IBM WebSphere Application Server - Express to determine how to install your application serving environment. This topic links to procedures for installing each scenario.

IBM WebSphere Application Server - Express is an integrated platform that contains an application server, Web development tools, a Web server, and additional supporting software and documentation.

The following information describes some common scenarios for installing the product in various topologies on one or more machines:

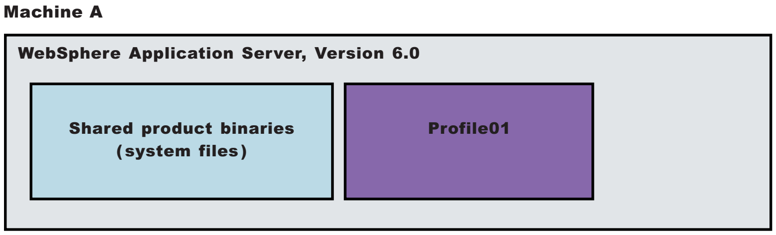
- **Scenario 1:** Single-machine installation of Express
- **Scenario 2:** Single-machine installation of Express and a Web server
- **Scenario 3:** Two-machine installation of Express and a Web server
- **Scenario 4:** Creating multiple profiles that use one installation of Express
- **Scenario 5:** Single-machine installation of Rational Web Developer
- **Scenario 6:** Multiple-machine installation of Express, Rational Web Developer, and a Web server

Each scenario includes a diagram and a list of detailed installation steps.

- **Scenario 1:** Install Express on a single machine.

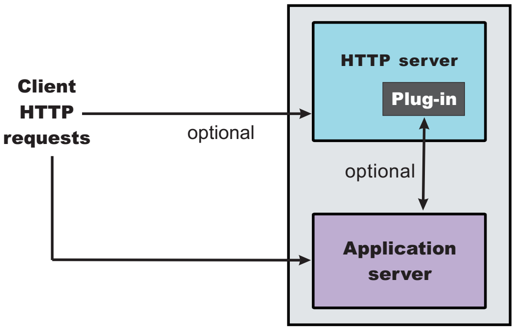
  Installing Express by itself on a single machine creates a stand-alone application server, which is server1. Installing Express creates a set of system files, called the *core product files*, and a *profile* for the application server. The profile is a separate set of files that define the application server environment.

  In this scenario, the application server uses its internal HTTP transport chain for communication, which is suitable for handling an application with a relatively low request work load. For example, this type of installation can support a simple test environment or a departmental intranet environment.


Machine A
WebSphere Application Server, Version 6.0
Shared product binaries (system files)
Profile01

- **Scenario 2:** Install Express and a Web server on a single machine.

  Installing a Web server, such as IBM HTTP Server, on the same machine as the application server provides more configuration options. Installing a Web server plug-in enables the Web server to communicate with the application server. This installation scenario supports rigorous testing environments or production environments that do not require a firewall. However, this scenario is not a typical production environment because everything is on one machine.



  1. Install the IBM HTTP Server or another supported Web server.

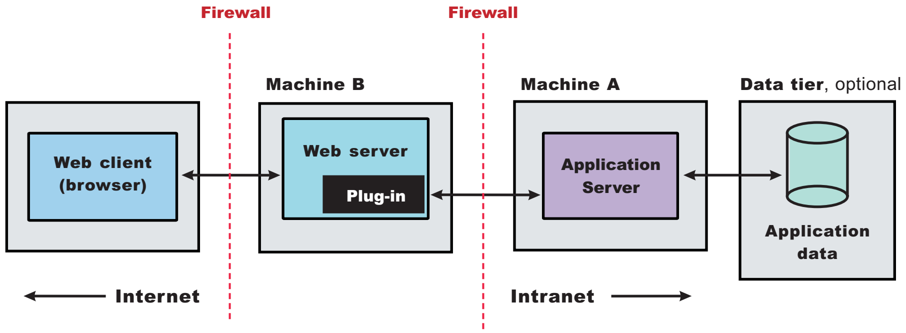
     While installing IBM HTTP Server, use its installation wizard to install the Web server plug-ins. After installing IBM HTTP Server, proceed to step 4.
  2. If you use another supported Web server or an existing installation of IBM HTTP Server, use the Plug-ins installation wizard to install the Web server plug-ins and configure the Web server. Because

the Web server and the application server are on the same machine, the Plug-ins installation wizard creates a Web server definition and you can skip step 4.

3. Create a Web server definition using the administrative console. Click **Servers > Web servers > New**.

- **Scenario 3:** Install Express and a Web server on separate machines.

  In the typical production environment, the application server on one machine communicates with a Web server on a separate (remote) machine through the Web server plug-in. Optional firewalls can provide additional security for the application server machine.



1. Install Express on Machine A.
2. Install IBM HTTP Server or another supported Web server on Machine B. While installing IBM HTTP Server, use its installation wizard to install the Web server plug-ins.
3. If you use another supported Web server or an existing installation of IBM HTTP Server, use the Plug-ins installation wizard to install the Web server plug-ins and configure the Web server on Machine B. The Plug-ins installation wizard creates a script named configure*Web_server_name* in the *plugins_root*/bin directory on Machine B.
4. Copy the configure*Web_server_name* script on Machine B to the *app_server_root*/*profile_name*/bin directory on Machine A.
5. Run the configure*Web_server_name* script on Machine A to create a Web server definition in the administrative console. You can then use the administrative console to manage the Web server.
6. Propagate the plugin-cfg.xml file on Machine A from the application server to the Web server using the administrative console. Click **Servers > Web servers >** *Web_server_name* **> Propagate Plug-in**. Web servers other than IBM HTTP Server require manual propagation.

- **Scenario 4:** Install multiple stand-alone application servers on one machine and a Web server on a separate machine.

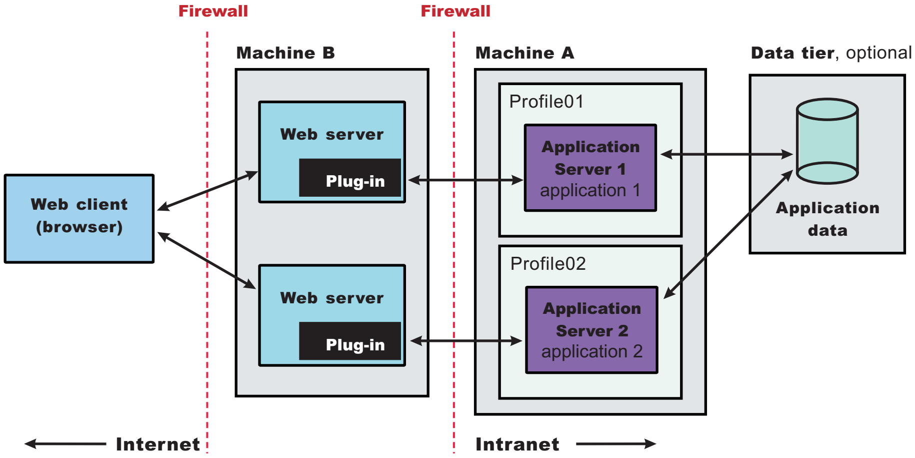
  A profile is a separate data partition containing the files that define the run-time environment for an application server. A default profile is created during the installation of the Express product. Create additional profiles using the Profile Management tool. Each profile defines a separate stand-alone application server that has its own administrative interface.

  After creating a profile and installing a dedicated Web server, use the Plug-ins installation wizard to install a Web server plug-in and to update the Web server configuration file. The Web server can then communicate with the application server.

  This topology lets each profile have unique applications, configuration settings, data, and log files, while sharing the same set of core product files. Creating multiple profiles creates multiple application server environments that you can dedicate to different purposes.

  For example, each application server on a Web site can serve a different application. In another example, each application server can be a separate test environment that you assign to a programmer or a development team.

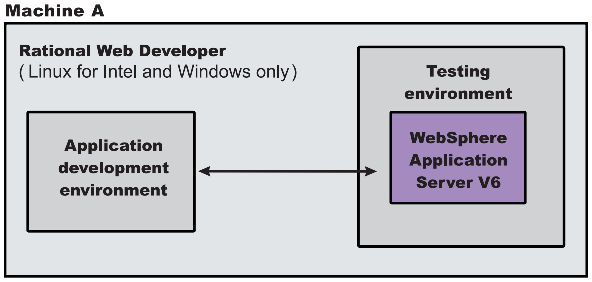
  Another feature of having multiple profiles is enhanced serviceability. For example, when a fix pack updates the system files, all application servers begin using the updated core product files.



1. Install Express on Machine A.
2. Install IBM HTTP Server or another supported Web server on Machine B. While installing IBM HTTP Server, use its installation wizard to install the Web server plug-ins.
3. If you use another supported Web server or an existing installation of IBM HTTP Server, use the Plug-ins installation wizard to install the Web server plug-ins and configure the Web server on Machine B. The Plug-ins installation wizard creates a script named configure*Web_server_name* in the *plugins_root*/bin directory on Machine B.
4. Copy the configure*Web_server_name* script on Machine B to the *app_server_root*/*profile_name*/bin directory on Machine A.
5. Run the configure*Web_server_name* script on Machine A to create a Web server definition in the administrative console. You can then use the administrative console to manage the Web server.
6. Propagate the plugin-cfg.xml file on Machine A from the application server to the Web server using the administrative console. Click **Servers** > **Web servers** > *Web_server_name* > **Propagate Plug-in**. Web servers other than IBM HTTP Server require manual propagation.
7. Create subsequent Application Server profiles using the Profile Management tool on Machine A.
8. Install subsequent IBM HTTP Servers or other supported Web servers on Machine B. While installing an IBM HTTP Server, use its installation wizard to install the Web server plug-ins.
9. If you use another supported Web server or an existing installation of IBM HTTP Server, use the Plug-ins installation wizard to configure each additional Web server on Machine B. Installation of the Web server plug-ins creates a script named configure*Web_server_name* in the *plugins_root*/bin directory on Machine B.
10. Copy the configure*Web_server_name* script on Machine B to the *app_server_root*/*profile_name*/bin directory on Machine A.
11. Run the configure*web_server_name* script on Machine A to create a Web server definition in the administrative console. You can then use the administrative console to manage the Web server.
12. On Machine A, propagate the plugin-cfg.xml file from the second application server to the Web server using the administrative console. Click **Servers** > **Web servers** > *Web_server_name* > **Propagate Plug-in**. Web servers other than IBM HTTP Server require manual propagation.

- **Scenario 5:** Install Rational Web Developer on a single machine.

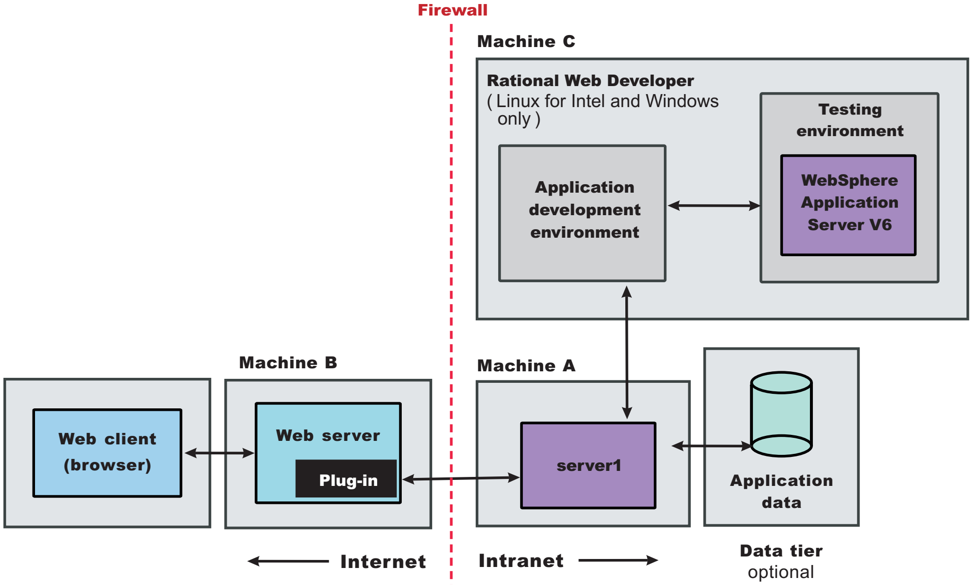
  Install Rational Web Developer by itself to create an integrated development environment. Developers can test applications in the test environment for V6 that is included in the Rational Web Developer product.


Machine A
Rational Web Developer
(Linux for Intel and Windows only)
Application development environment
Testing environment
WebSphere Application Server V6

1. Install Rational Web Developer.

- **Scenario 6:** Install Express, Rational Web Developer, and a Web server on multiple machines.

  In this integrated testing environment, install the application server behind an optional firewall on one machine and the Web server on a separate (remote) machine. The Web server plug-in lets the application server and Web server communicate with each other. Install Rational Web Developer on a separate Linux (Intel) or Windows machine. This installation scenario is useful for quality assurance because the test environment closely mirrors the typical production environment.



1. Install Express on Machine A.
2. Install IBM HTTP Server or another supported Web server on Machine B. While installing IBM HTTP Server, use its installation wizard to install the Web server plug-ins.
3. If you use another supported Web server or an existing installation of IBM HTTP Server, use the Plug-ins installation wizard to install the Web server plug-ins and configure the Web server on Machine B. Installation of the Web server plug-ins creates a script named configure*Web_server_name* in the *plugins_root*/bin directory on Machine B.
4. Copy the configure*Web_server_name* script on Machine B to the *app_server_root*/*profile_name*/bin directory on Machine A.
5. Run the configure*Web_server_name* script on Machine A to create a Web server definition in the administrative console. You can then use the administrative console to manage the Web server.
6. Propagate the plugin-cfg.xml file on Machine A from the application server to the Web server using the administrative console. Click **Servers** > **Web servers** > *Web_server_name* > **Propagate Plug-in**. Web servers other than IBM HTTP Server require manual propagation.
7. Install Rational Web Developer on Machine C.

You can review common installation scenarios to find a possible match for the topology that you intend to install. Each product installation diagram provides a high-level procedure for installing the components that comprise the topology.

After determining a possible topology, you are ready to follow the detailed installation instructions for each product that you plan to install.

Planning to install Web server plug-ins

This topic describes common installation scenarios and links to component installation procedures for each scenario.

The primary production configuration 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.

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:

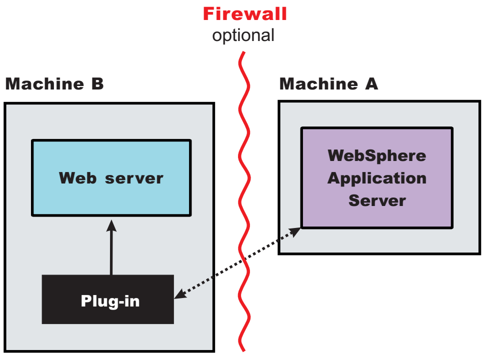
    Detects the default profile on a local application server machine and creates the Web server definition for it directly. Install the Web server and its plug-in on the same machine with the application server. When multiple profiles exist, the plug-ins installer allows the user to select the profile to configure. If there is only one profile (default), then it is used during configuration. In this case, the profile selection panel will not be displayed to the user. 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.**

  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

Step	Machine	Task
1	A	Install your WebSphere Application Server product.
2	B	Install IBM HTTP Server or another supported Web server.
3	B	Install the binary plug-in module using the Plug-ins installation wizard. . The script for creating and configuring the Web server is created under the *plugins_root/* bin directory.
4	B	Copy the configure*web_server_name* script to Machine A. If one machine is running under Linux or UNIX and the other machine is running under Windows, copy the script from the *plugins_root/* bin/ crossPlatformScripts directory.
5	A	Paste the configure*web_server_name* script from Machine B to the *app_server_root/* bin directory on Machine A.
6	A	Run the script from a command line.
7	A	Verify that the application server is running. Open the administrative console and save the changed configuration.
8	B	Linux Run the *plugins_root/*setupPluginCfg.sh script for a Domino Web Server before starting a Domino Web server.Otherwise, start the Web server.
9	B	Run the Snoop servlet. To verify with your own application, regenerate and propagate the plugin-cfg.xml file after installing the application.

Regeneration of the plugin-cfg.xml file

During the installation of the plug-ins, the temporary plugin-cfg.xml file is installed on Machine B in the *plugins_root/* config/ web_server_name directory.

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

To use the real plugin-cfg.xml file from the application server, propagate the plugin-cfg.xml file as described in the next section.

Propagation of the plugin-cfg.xml file

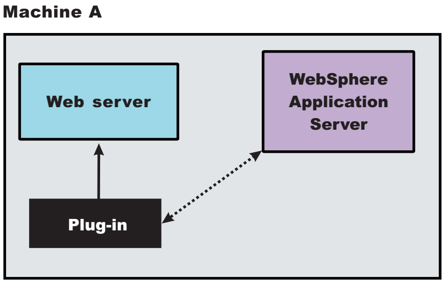
The Web server plug-in configuration service propagates the plugin-cfg.xml file automatically for IBM HTTP Server only.

For all other Web servers, propagate the plug-in configuration file manually. Copy the plugin-cfg.xml file from the *profile_root/* config/ cells/ *cell_name/* nodes/ *web_server_name*_node/ servers/ *web_server_name* directory on Machine A. Paste the file into the *plugins_root/* config/ *web_server_name* directory on Machine B.

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

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

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



Local installation scenario

Step	Machine	Task
1	A	Install your WebSphere Application Server product. See Installing the product and additional software.
2	A	Install IBM HTTP Server or another supported Web server.
3	A	Install the binary plug-in module using the Plug-ins installation wizard. The Web server definition is automatically created and configured during the installation of the plug-ins.
4	A	Verify that the application server is running. Open the administrative console and save the changed configuration.
5	B	Linux Run the *plugins_root*/setupPluginCfg.sh script for a Domino Web Server before starting a Domino Web server.Otherwise, start the Web server.
6	B	Run the Snoop servlet. To verify with your own application, regenerate and propagate the plugin-cfg.xml file after installing the application.

Regeneration of the plugin-cfg.xml file

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

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

Propagation of the plugin-cfg.xml file

The local file does not require propagation.

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

Planning to install WebSphere Application Client

Examine typical topologies and uses for WebSphere Application Client to determine how you might use this technology.

This topic is one in a series of topics described in Chapter 1, “Planning the installation,” on page 1. Consider all of the planning scenarios that are mentioned in the parent article to determine the best approach to installing your e-business network. This topic describes installing and using the WebSphere Application Server Clients.

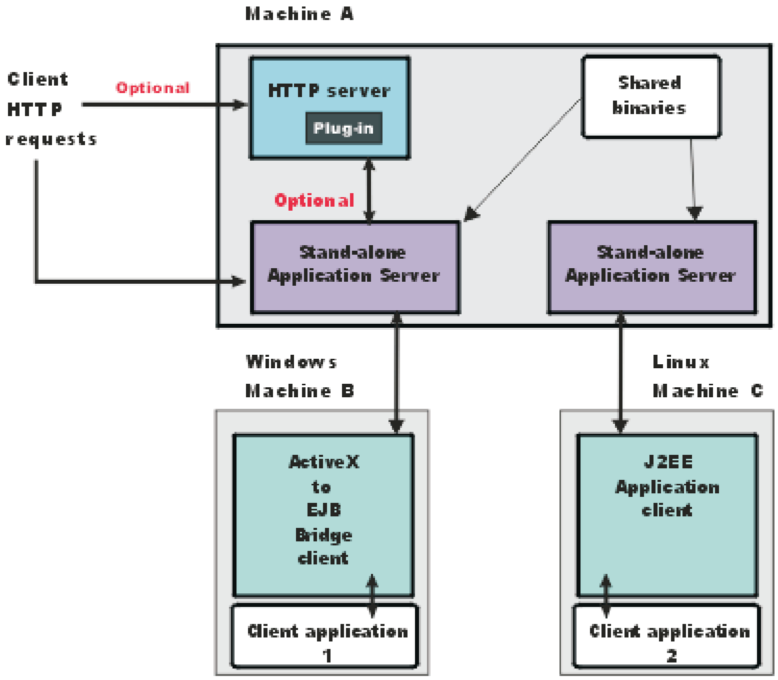
In a traditional client server environment, the client requests a service and the server fulfills the request. Multiple clients use a single server. Clients can also access several different servers. This model persists for Java clients except that now these requests use a client run-time environment.

In this model, the client application requires a servlet to communicate with the enterprise bean, and the servlet must reside on the same machine as the WebSphere Application Server.

The Application Client for WebSphere Application Server, Version 6 consists of the following models:

- ActiveX application client
- Applet client
- J2EE application client
- Pluggable and thin application clients

The following graphic shows a topology for installing the Application Client and using client applications:



The example shows two types of application clients installed in a topology that uses client applications to access applications and data on Machine A:

- The ActiveX application client on Machine B is a Windows only client that uses the Java Native Interface (JNI) architecture to programmatically access the Java virtual machine (JVM) API. The JVM code exists in the same process space as the ActiveX application (Visual Basic, VBScript, or Active Server Pages (ASP) files) and remains attached to the process until that process terminates.
- The J2EE application client on Machine C is a Java application program that accesses enterprise beans, Java Database Connectivity (JDBC) APIs, and Java Message Service message queues. The application program must configure the execution environment of the J2EE application client and use the Java Naming and Directory Interface (JNDI) name space to access resources.

Use the following procedure as a example of installing the Application Client.

1. Install the WebSphere Application Server product from your product CD on Machine A to establish the core product files.
2. Use the Profile Management tool to create the additional stand-alone application server profile.
3. Use the administrative console of each application server to deploy any user applications.
4. Use the administrative console of each application server to create a Web server configuration for the Web server.
5. Use the administrative console of each application server to regenerate each plugin-cfg.xml file in the local Web server configuration.
6. **Optional:** Install the IBM HTTP Server from the product CD on Machine A.
7. **Optional:** Use the Plug-ins installation wizard to install the plug-in for IBM HTTP Server on Machine A.

   The wizard automatically configures the HTTP Server to communicate with the first application server.
8. Install the Application Client package from the product disc.

   The diagram shows two different types of application clients on two different operating systems. Although this example shows two application clients, you do not have to install two application clients to have a working system. Each application client is fully functional and works independently of the other.

   Optionally install the Application Client from your product CD on Machine B.

a. Select the Custom installation type.
b. Select the ActiveX to EJB Bridge feature.
c. Select to add the Java run time to the system path.
d. Select the Java run time as the default JRE, which adds the Java run time path to the beginning of the system path.

Optionally install the Application Client from your product CD on Machine C.

a. Select the Custom install type.
b. Select the J2EE application client feature.

This topic can help you plan run-time environments for client applications.

Planning to create application server environments

Application server profiles are the run-time environments for application server processes. This topic describes common scenarios for creating application server profiles and provides links to profile creation procedures for each scenario.

Installing the core product files for a WebSphere Application Server product creates one or more profiles. Use the Profile Management tool to create additional application server run-time environments.

This topic describes how to create application server profiles. Each profile is a run-time environment for the application server that it defines, with data files, configuration files, applications, and an administrative console.

Create a stand-alone application server.

The installation procedure creates an application server during installation named server1.

However, you can use the Profile Management tool to create additional stand-alone application servers.

Following this procedure results in reviewing common installation scenarios to find a possible match for the topology that you intend to create. This procedure shows how to use the Profile Management tool to create additional profiles.

Use the Profile Management tool to create additional stand-alone application server profiles.

After determining a possible topology, follow the steps in the overall procedure.

One step not mentioned in any of these scenarios is to prepare the operating system for installation. Useful links to the installation procedures for each installable component are in the list of related topics.

Queuing network

WebSphere Application Server contains interrelated components that must be harmoniously tuned to support the custom needs of your end-to-end e-business application. These adjustments help the system achieve maximum throughput while maintaining the overall stability of the system.

This group of interconnected components is known as a queuing network. These queues or components include the network, Web server, Web container, EJB container, data source, and possibly a connection manager to a custom back-end system. Each of these resources represents a queue of requests waiting to use that resource.

Various queue settings include:

- IBM HTTP Server: MaxClients for UNIX and ThreadsPerChild for Windows NT and Windows 2000 systems
- Web container: **Maximum size** , **MaxKeepAliveConnections**, and **MaxKeepAliveRequests**.
- **Tuning Object Request Brokers**.
- Data source **connection pooling** and **statement cache size** .

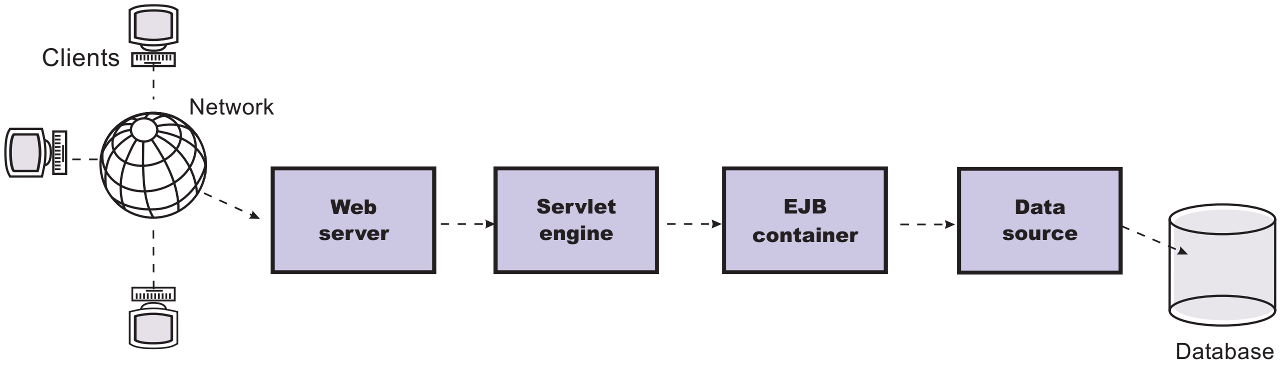


Figure Reference 1: WebSphere queuing network

Most of the queues that make up the queuing network are closed queues. A closed queue places a limit on the maximum number of requests present in the queue, while an open queue has no limit. A closed queue supports tight management of system resources. For example, the Web container thread pool setting controls the size of the Web container queue. If the average servlet running in a Web container creates 10MB of objects during each request, a value of 100 for thread pools limits the memory consumed by the Web container to 1GB.

In a closed queue, requests can be active or waiting. An active request is doing work or waiting for a response from a downstream queue. For example, an active request in the Web server is doing work, such as retrieving static HTML, or waiting for a request to complete in the Web container. A waiting request is waiting to become active. The request remains in the waiting state until one of the active requests leaves the queue.

All Web servers supported by WebSphere Application Server are closed queues, as are WebSphere Application Server data sources. You can configure Web containers as open or closed queues. In general, it is best to make them closed queues. EJB containers are open queues. If there are no threads available in the pool, a new one is created for the duration of the request.

If enterprise beans are called by servlets, the Web container limits the number of total concurrent requests into an EJB container, because the Web container also has a limit. The Web container limits the number of total concurrent requests only if enterprise beans are called from the servlet thread of execution. Nothing prevents you from creating threads and bombarding the EJB container with requests. Therefore, servlets should not create their own work threads.

Queuing and clustering

Cloning application servers to create a cluster can be a valuable asset in configuring highly-scalable production environments, especially when the application is experiencing bottlenecks that are preventing full CPU utilization of symmetric multiprocessing (SMP) servers.

When adjusting the WebSphere Application Server system queues in clustered configurations, remember that when a server is added to a cluster, the server downstream receives twice the load.

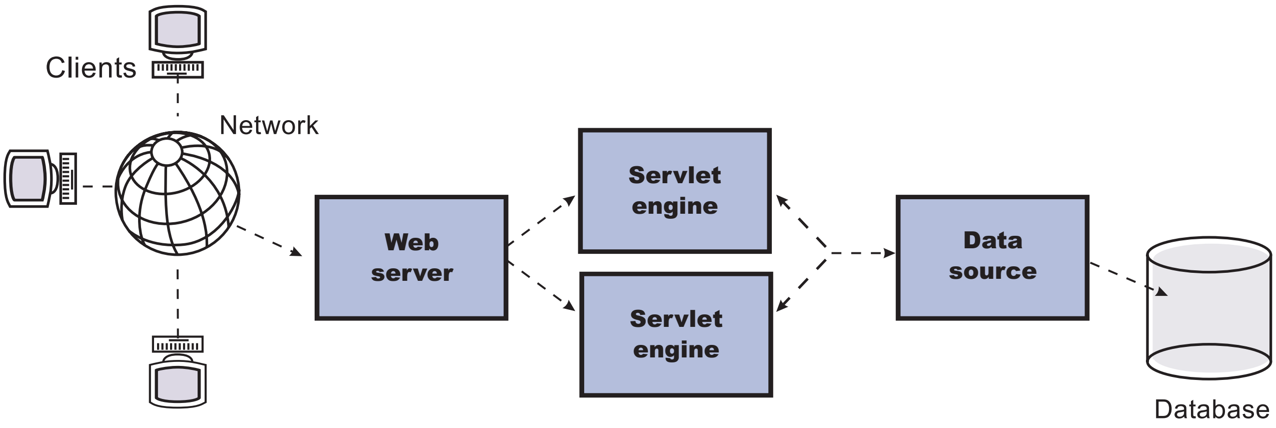


Figure Reference 1: Clustering and queuing

Two servlet engines are located between a Web server and a data source. It is assumed that the Web server, servlet engines and data source, but not the database, are all running on a single SMP server. Given these constraints, the following queue considerations must be made:

- Double the Web server queue settings to ensure ample work is distributed to each Web container.
- Reduce the Web container thread pools to avoid saturating a system resource like CPU or another resource that the servlets are using.
- Reduce the data source to avoid saturating the database server.
- Reduce Java heap parameters for each instance of the application server. For versions of the Java virtual machine (JVM) shipped with WebSphere Application Server, it is crucial that the heap from all JVMs remain in physical memory. For example, if a cluster of four JVMs is running on a system, enough physical memory must be available for all four heaps.

Queue configuration tips

A methodology exists for configuring the WebSphere Application Server queues. Moving the database server onto another machine or providing more powerful resources, for example a faster set of CPUs with more memory, can dramatically change the dynamics of your system.

There are four tips for queuing:

- **Minimize the number of requests in WebSphere Application Server queues**.

  In general, requests wait in the network in front of the Web server, rather than waiting in WebSphere Application Server. This configuration only supports those requests that are ready for processing to enter the queuing network. Specify that the queues furthest upstream or closest to the client are slightly larger, and queues further downstream or furthest from the client are progressively smaller.

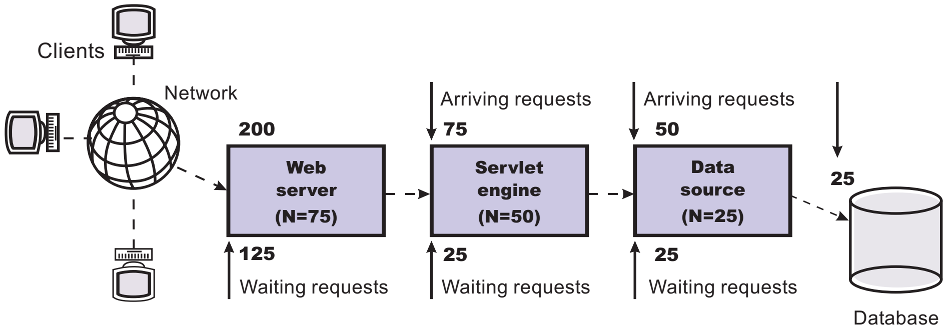
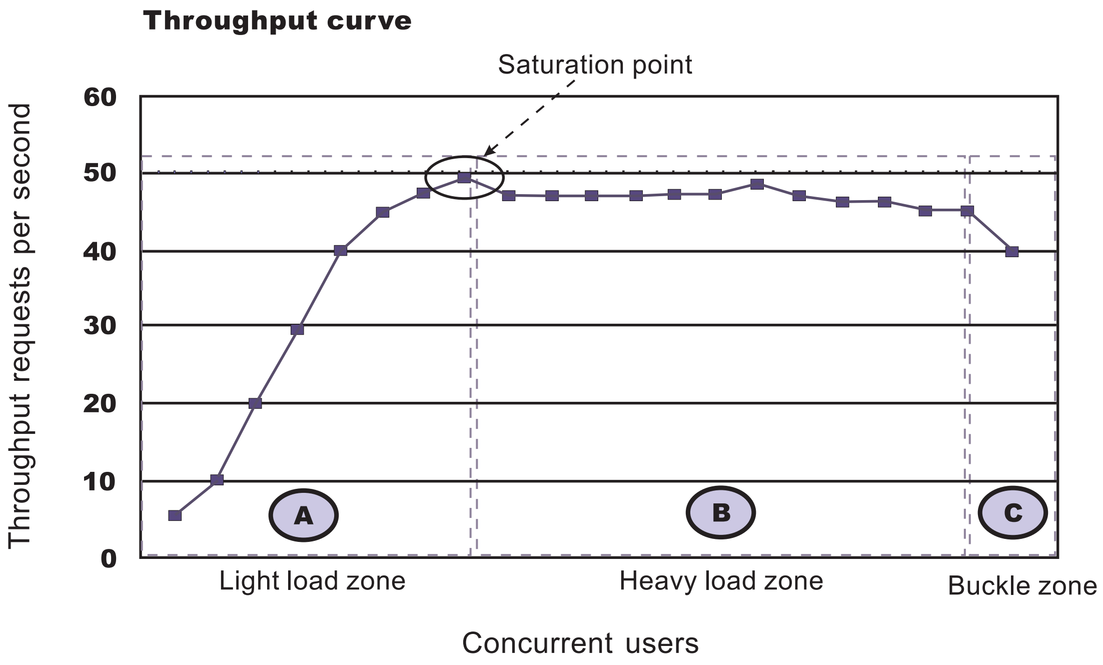


Figure Reference 1: Upstream queuing network

Queues in the queuing network become progressively smaller as work flows downstream. When 200 client requests arrive at the Web server, 125 requests remain queued in the network because the Web server is set to handle 75 concurrent clients. As the 75 requests pass from the Web server to the Web container, 25 requests remain queued in the Web server and the remaining 50 are handled by the Web container. This process progresses through the data source until 25 user requests arrive at the final destination, the database server. Because there is work waiting to enter a component at each point upstream, no component in this system must wait for work to arrive. The bulk of the requests wait in the network, outside of WebSphere Application Server. This type of configuration adds stability, because no component is overloaded.

- **Draw throughput curves to determine when the system capabilities are maximized**.

  You can use a test case that represents the full spirit of the production application by either exercising all meaningful code paths or using the production application. Run a set of experiments to determine when the system capabilities are fully stressed or when it has reached the saturation point. Conduct these tests after most of the bottlenecks are removed from the application. The goal of these tests is to drive CPUs to near 100% utilization. For maximum concurrency through the system, start the initial baseline experiment with large queues. For example, start the first experiment with a queue size of 100 at each of the servers in the queuing network: Web server, Web container and data source. Begin a series of experiments to plot a throughput curve, increasing the concurrent user load after each experiment. For example, perform experiments with one user, two users, five, 10, 25, 50, 100, 150 and 200 users. After each run, record the throughput requests per second, and response times in seconds per request. The curve resulting from the baseline experiments resembles the following typical throughput curve shown as follows:



The WebSphere Application Server throughput is a function of the number of concurrent requests present in the total system. Section A, the light load zone, shows that the number of concurrent user requests increases, the throughput increases almost linearly with the number of requests. At light loads, concurrent requests face very little congestion within the WebSphere Application Server system queues. At some point, congestion starts to develop and throughput increases at a much lower rate until it reaches a saturation point that represents the maximum throughput value, as determined by some bottleneck in the WebSphere Application Server system. The most manageable type of bottleneck occurs when the WebSphere Application Server machine CPUs become fully utilized because adding CPUs or more powerful CPUs fixes the bottleneck.

In the heavy load zone or Section B, as the concurrent client load increases, throughput remains relatively constant. However, the response time increases proportionally to the user load. That is, if the user load is doubled in the heavy load zone, the response time doubles. At some point, represented by Section C, the buckle zone, one of the system components becomes exhausted. At this point, throughput starts to degrade. For example, the system might enter the buckle zone when the network connections at the Web server exhaust the limits of the network adapter or if the requests exceed operating system limits for file handles.

If the saturation point is reached by driving CPU utilization close to 100%, you can move on to the next step. If the saturation CPU occurs before system utilization reaches 100%, it is likely that another bottleneck is being aggravated by the application. For example, the application might be creating Java objects causing excessive garbage collection bottlenecks in the Java code.

There are two ways to manage application bottlenecks: remove the bottleneck or clone the bottleneck. The best way to manage a bottleneck is to remove it. You can use a Java-based application profiler, such as Rational Application Developer, Performance Trace Data Visualizer (PTDV), Borland's Optimizeit, JProbe or Jinsight to examine overall object utilization.

- **Decrease queue sizes while moving downstream from the client**.

  The number of concurrent users at the throughput saturation point represents the maximum concurrency of the application. For example, if the application saturates WebSphere Application Server at 50 users, using 48 users might produce the best combination of throughput and response time. This value is called the Max Application Concurrency value. Max Application Concurrency becomes the preferred value for adjusting the WebSphere Application Server system queues. Remember, it is desirable for most users to wait in the network; therefore, queue sizes should increase when moving downstream farther from the client. For example, given a Max Application Concurrency value of 48, start with system queues at the following values: Web server 75, Web container 50, data source 45. Perform a set of additional experiments adjusting these values slightly higher and lower to find the best settings.

To help determine the number of concurrent users, view the Servlet Engine Thread Pool and Concurrently Active Threads metric in the Tivoli Performance Viewer.

- **Adjust queue settings to correspond to access patterns**.

  In many cases, only a fraction of the requests passing through one queue enters the next queue downstream. In a site with many static pages, a number of requests are fulfilled at the Web server and are not passed to the Web container. In this circumstance, the Web server queue can be significantly larger than the Web container queue. In the previous example, the Web server queue was set to 75, rather than closer to the value of Max Application Concurrency. You can make similar adjustments when different components have different execution times.

  For example, in an application that spends 90% of its time in a complex servlet and only 10% of its time making a short JDBC query, on average 10% of the servlets are using database connections at any time, so the database connection queue can be significantly smaller than the Web container queue. Conversely, if the majority of servlet execution time is spent making a complex query to a database, consider increasing the queue values at both the Web container and the data source. Always monitor the CPU and memory utilization for both the WebSphere Application Server and the database servers to verify that the CPU or memory are not saturating.

Chapter 2. Configuring the product after installation

This topic summarizes how to configure the application serving environment.

Use the First steps console to configure and test the WebSphere Application Server environment after creating a profile.

This procedure uses the First steps console to launch the installation verification test (IVT) that tests and verifies your WebSphere Application Server - Express environment. This procedure also uses the First steps console to launch the Profile Management tool to create an additional application server.

1. Start the First steps console by selecting the check box on the last panel of the wizard.

   The First steps console can start automatically at the end of the installation. Select the check box on the last panel of the Installation wizard.

   The First steps console is an easy way to start using the product. The console provides one-stop access to the administrative console, Samples Gallery, Profile Management tool, installation verification test, Migration wizard, and other activities.

   See the description of the “firststeps command” for more information.
2. Click **Installation verification** on the First steps console.

   The installation verification test starts the application server process named server1 and runs several tests to verify that the server1 process can start without errors.

   See “Using the installation verification tools” on page 20 for more information.
3. Click Profile Management tool on the First steps console to create an application server profile.

   You can create multiple application servers on your system without installing the product again.
4. Start the First steps console by selecting the check box on the last panel of the Profile Management tool.

   This First steps console belongs to the Application Server profile that you just created. Each profile has its own First steps console.
5. Click **Installation verification** on the First steps console.

   The installation verification test starts the new application server process named server1 and runs several tests to verify that the server1 process can start without error.

This procedure results in configuring and testing the application server environment.

See “Planning to install WebSphere Application Server - Express” on page 2 for diagrams of topologies that you can create using the First steps console and the Profile Management tool.

firststeps command

The firststeps command starts the First steps console. The First steps console is a post-installation ease-of-use tool for directing WebSphere Application Server - Express elements from one place. Options display dynamically on the First steps console, depending on features that you install and the availability of certain elements on a particular operating system platform. Options include verifying the installation, starting and stopping application server processes, creating profiles, accessing the administrative console, launching the Migration wizard, accessing the online information center, and accessing the Samples gallery.

First steps overview

Select the check box to start the First steps console at the end of the product installation.

You can also start the First steps console from the command line as described later in this topic.

Option descriptions

Installation verification

This option starts the installation verification test (IVT). The test consists of starting and monitoring the application server during its start up.

If this is the first time that you have used the First steps console for this application server profile, click **Installation verification** to verify the installation. The verification process starts the application server.

The **Start the server** option is unavailable while the IVT runs.

The IVT provides the following useful information about the application server:

- The server name: server1
- The name of the profile
- The profile file path
- The type of profile: default
- The cell name
- The node name
- The current encoding
- The port number for the administrative console
- Various informational messages that include the location of the SystemOut.log file and how many errors are listed within the file
- A completion message

Start the server

This option toggles to **Stop the server** when the application server runs.

This option displays when the First steps console is in a stand-alone application server profile.

After selecting the **Start the server** option, an output screen displays with status messages. The success message informs you that the server is open for e-business. Then the menu item toggles to **Stop the server** and both the **Administrative console** option and the **Samples gallery** option enable.

If you select the **Start the server** option, the **Installation verification** option is unavailable while the application server runs.

Administrative console

This option is unavailable until the application server runs.

The administrative console is a configuration editor that runs in one of the supported Web browsers. The administrative console lets you work with XML configuration files for the stand-alone application server .

To launch the administrative console, click **Administrative console** or point your browser to http://localhost:9060/ibm/console. Substitute the host name for localhost if the address does not load. Verify the installation to verify the administrative console port number, if 9060 does not load.

The administrative console prompts for a login name. This is not a security item, but merely a tag to identify configuration changes that you make during the session. Secure signon is also available when administrative security is enabled.

The installation procedure in the information center cautions you to write down the administrative user ID and password when security is enabled during installation. Without the ID and password, you cannot use the administrative console or scripting.

Profile Management tool

This option starts the Profile Management tool. The PMT creates additional stand-alone application server profiles.

Each profile has its own administrative interface.

Each profile also has its own First steps console. The location of the command is within the profile as described later in this topic.

Samples gallery

This option starts the Samples gallery. The option is unavailable until you start the application server. The option displays when you have installed the Samples during installation.

To launch the Samples gallery, click **Samples gallery** or point your browser to http://localhost:9080/WSsamples. The Web address is case sensitive.

Substitute your own host name and default host port number if the address does not load. Verify the port number by opening the administrative console and clicking **Servers > Application servers > server1 > [Communications] Ports**. Use the WC_defaulthost port number value or the WC_defaulthost_secure value instead of 9080, which is the default.

If you do not install the Samples during the initial installation of the product, the option does not display on the First steps console. You can perform an incremental installation to add the Samples feature. After adding the Samples, the option displays on the First steps console.

Information center for WebSphere Application Server

This option links you to the online information center at http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp.

Migration wizard

This option starts the Migration wizard, which is the graphical interface to the migration tools.

The migration tools are described in the *Migrating, coexisting, and interoperating* PDF.

Exit This option closes the First steps console.

Location of the command file

The location of the firststeps command that starts the First steps console for a profile is:

- ▶ Linux *profile_root*/firststeps/firststeps.sh
- ▶ Windows *profile_root*\firststeps\firststeps.bat

Parameters

No parameters are associated with this command.

Syntax for the firststeps command

Use the following syntax for the command:

- ▶ Linux ./firststeps.sh
- ▶ Windows firststeps.bat

Link tips

The following links exist on the First steps console for the WebSphere Application Server - Express product:

Option	Link
Installation verification	Calls the ivt command. The location of the installation verification test command is: • Linux *profile_root*/bin/ivt.sh • Windows *profile_root*\bin\ivt.bat
Start the server	Calls the startServer command. The location of the startServer command is: • Linux *profile_root*/bin/startServer.sh server1 • Windows *profile_root*\bin\startServer.bat server1 When you have more than one application server on the same machine, the command starts the same application server that is associated with the First steps console.
Stop the server	Calls the stopServer command. The location of the stopServer command is: • Linux *profile_root*/bin/stopServer.sh server1 • Windows *profile_root*\bin\stopServer.bat server1
Administrative console	Opens the default browser to the http://localhost:9060/ibm/console Web address. When you have more than one application server on the same machine, the port varies. The First steps console starts the administrative console that is associated with the First steps console.
Profile Management tool	The command file name is: • Linux *app_server_root*/bin/ProfileManagement/pmt.sh • Windows *app_server_root*/bin/ProfileManagement/pmt.bat
Samples gallery	Opens the default browser to the http://localhost:9080/WSsamples Web address.
Information center for WebSphere Application Server products	Opens the default browser to the online information center: http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp
Migration wizard	Calls the **migration** command to start the Migration wizard. The location of the **migration** command is: • Linux *app_server_root*/bin/migration.sh • Windows *app_server_root*\bin\migration.bat The migration tools are also in the /migration folder on the supplements disc.

Using the installation verification tools

Use the installation verification tools to verify that the installation of the product and the application server profile is successful. A *profile* consists of files that define the runtime environment for an application server. Verify the core product files with the installver checksum tool. Verify each profile from its First steps console with the IVT tool.

After installing the product, you are ready to use the installation verification tools.

Use the installation verification tools to gain assurance that the product is successfully installed. Two tools exist. This task describes when and how to use to both tools. One tool is the checksum tool (the installver tool). The installver tool compares the checksum of each installed file to the correct checksum value for each file and reports differences. The other tool is the installation verification test (IVT). The IVT tests deployment manager profiles and stand-alone application server profiles to make sure that the server processes can start.

The IVT program scans product log files for errors and verifies core functionality of the product installation.

After installing the product, the Installation wizard displays a prompt for starting the First steps console.

Installation verification is the first option on the First steps console.

The test consists of starting and monitoring the application server during its start up.

1. Run the installver tool to verify that all files are correctly installed.

   See “Verifying checksums of installed files” on page 23 for more information.
2. Use the installation verification test to verify the proper creation of profiles.
   a. Select **Installation verification** on the First steps console after installing the product.

      You can also start the First steps console from the command line as described in “firststeps command” on page 17.

      You can also start the “ivt command” on page 22 directly from the bin directory of the profile:
      - Linux *profile_root*/bin/ivt.sh
      - Windows *profile_root*\bin\ivt.bat

      If you create additional profiles, the ivt script location is within the *plugins_root*/bin directory.

   b. Observe the results in the First steps status window.

      The log file for installation verification is the *profile_root*/logs/ivtClient.log.

      The IVT provides the following useful information about the application server:
      - The application server name
      - The name of the profile
      - The profile file path
      - The type of profile
      - The node name
      - The current encoding
      - The port number for the administrative console
      - Various informational messages that include the location of the SystemOut.log file and how many errors are listed within the file
      - A completion message

      As the IVT starts the application server on a Windows platform, the IVT attempts to start the Windows service for the application server, if a Windows service exists. This is true even though the Windows service might have a manual startup type.

   c. If the log shows that errors occurred during the installation verification, correct the errors and run the IVT again. If necessary, create a new profile after correcting the error, and run the IVT on the new profile.

The installver tool validates the integrity of all installed files.

The IVT tool starts the server process of a profile automatically if the server is not running. Once the server initializes, the IVT runs a series of verification tests. The tool displays pass or fail status in a console window. The tool also logs results to the *plugins_root*/logs/ivtClient.log file. As the IVT verifies your system, the tool reports any detectable errors in the SystemOut.log file.

After installing the product and verifying the installation, you can configure the installation by creating more profiles.

You can also install other packages on the product installation image, such as IBM HTTP server, the Web server plug-ins, or the Application Client.

ivt command

The ivt command starts the installation verification test (IVT) program. The IVT verifies that the installation of the application server profile was successful. A *profile* consists of files that define the run-time environment for an application server. Each profile has its own IVT command.

The IVT program starts the application server automatically if the server process is not already running. After the server process initializes, the IVT runs a series of verification tests and displays pass or fail status in a console window.

The IVT program scans the SystemOut.log file for errors and verifies core functionality of the profile.

You can start the IVT program from the command line or from the First steps console.

Location of the command file

The location of the installation verification test script for a profile is the *profile_root*/bin directory. The script file name is:

- Linux ivt.sh
- Windows ivt.bat

Parameters

The following parameters are associated with this command.

server_name
Required parameter that identifies the name of the server process, such as server1.

profile_name
Required parameter that identifies the name of the profile that contains the server definition.

-p *server_port_number*
Optional parameter that identifies the default_host port when the port is not 9080, which is the default.

-host *machine_host_name*
Optional parameter that identifies the host machine of the profile to test. The default is localhost.

Syntax for the ivt command

Use the following syntax for the command:

- Linux *profile_root*/bin/ivt.sh
- Windows *profile_root*\bin\ivt.bat

Logging

The ivt command logs results to the *profile_root*/logs/ivtClient.log file.

Example

The following examples test the server1 process in the profile01 profile on the myhost machine using the default_host on port 9081.

Windows

```
ivt.bat server1 profile01 -p 9081 -host myhost
```

Linux

```
ivt.sh server1 profile01 -p 9081 -host myhost
```

Verifying checksums of installed files

After installing the product or after installing maintenance packages, you can use the installation verification utility (IVU) to compute checksums of the installed file set to verify the checksum against the checksum in the product bill of materials.

Installing the product also installs the IVU, which is the installver command-line tool.

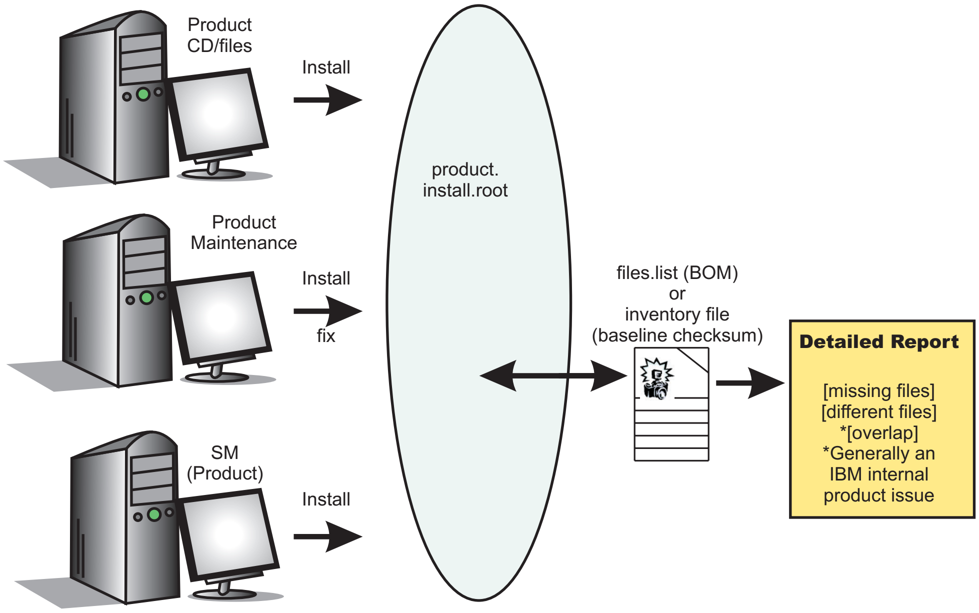
This topic describes using the **installver** command to compute a checksum on the installed files and compare the checksum to the product bill of materials.

The IVU tool is installed during the installation of the following product components:

- WebSphere Application Server - Express
- Application Client
- IBM HTTP Server
- Web server plug-ins
- Update Installer for WebSphere Application Server

You can also use the IVU to compute a new checksum for a system after you make significant configuration changes. The installver tool computes a new baseline checksum for each file in the inventory of a configured system to use to identify file changes in the later comparisons. Such a comparison is useful for detecting file tampering on the configured system, for example.

You can use the new checksums to compare installations on multiple systems. The following graphic illustrates the main use-cases where you can perform product verification at any point of the product life-cycle or use the inventory file, which is part of the baseline checksum feature, to provide full-file verification of a configured system.



Although the most common use of the tool is to compare the product bill of materials to the installed file set after installation or after installing maintenance, other tasks are also possible.

- Verify the installed files against the bill of materials.

  See "Verifying against the bill of materials" on page 25 for more information.
- Create and use a new baseline checksum.

  See "Computing a new baseline checksum for an inventory of configured files" on page 28 for more information.
- Exclude files and components from the comparison.

  See "Excluding files from a checksum comparison" on page 30 for more information.
- Include only specific files and components in the comparison.

  See "Comparing specific file and component checksums" on page 33 for more information.
- Change the default message digest algorithm for computing checksums.

  See "Changing the default message digest algorithm" on page 35 for more information.
- Handle out-of-memory conditions.

  See "Handling out-of-memory situations" on page 36 for more information.
- Verify the installver command files.

  See "Verifying the installver command" on page 36 for more information.

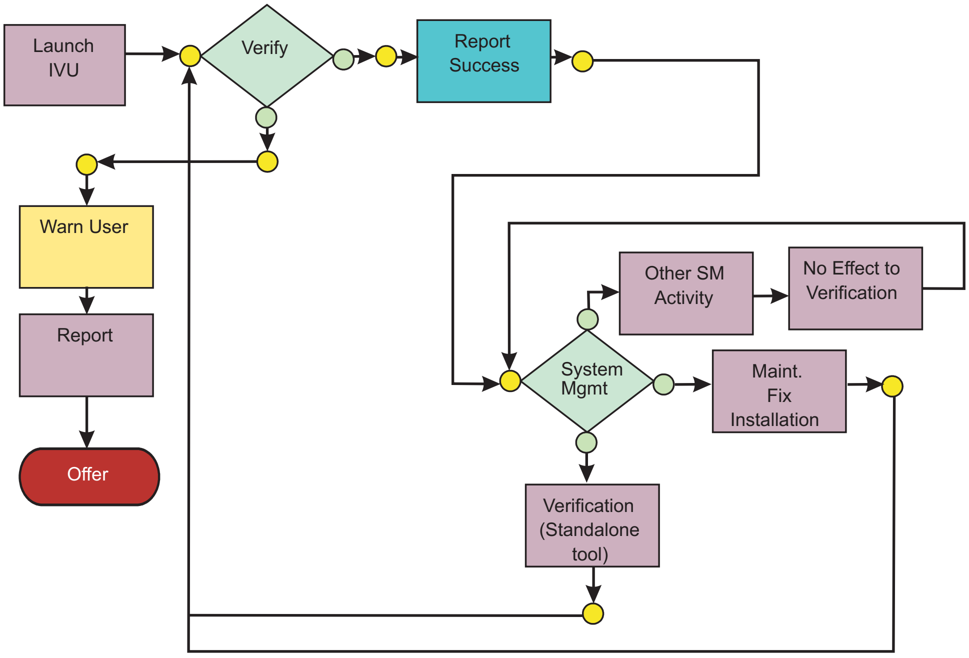
When you are satisfied that your installed or updated file set matches the product bill of materials, you are finished installing and verifying the product or updating and verifying the product.

If you detect a problem, see if the problem is a known problem by checking the Support page.

IBM Support has documents and tools that can save you time gathering information needed to resolve problems. Before opening a problem report, see the Support page:

- http://www.ibm.com/software/webservers/appserv/was/support/

The IVU performs the tasks using the logic described in the following graphic:



Verifying against the bill of materials

After installation, verify actual checksums of installed files against a bill of materials that ships with the product. If the checksums match, the installed product is installed correctly. If the checksums differ, review the differences to determine whether a problem exists.

Install the product before attempting to compare checksums of the installed files to the shipped bill of materials.

This topic describes using the installver command to compare a set of bill-of-material files against a checksum of the installed files to verify that all installed files are correct. The product includes a bill-of-materials file for each component to provide this system of verifying installation files.

The installver tool dynamically generates a list of total components found in the installation.

- Check the bill of materials against the installed file system.

  Compare the checksum of product files to the correct checksum in the bill-of-material files:

  – Linux *app_server_root*`/bin/installver.sh`

  – Windows *app_server_root*`\bin\installver.bat`
- Compare checksums and display trace results:

  – Linux `./installver.sh -trace`

  – Windows `installver.bat -trace`
- Display usage information:

  – Linux `./installver.sh -help`

  – Windows `installver.bat -help`
- Compare checksums and include only specified files and components in the comparison:

  See “Comparing specific file and component checksums” on page 33 for more information about comparing only files and components that you list in the command.
- Compare checksums and ignore user-excluded files:

See “Excluding files from a checksum comparison” on page 30 for information about creating a list of user-excluded files.

Ignore the list of files to exclude and include the files in the comparison:

- Linux ./installver.sh -ignoreuserexclude
- Windows installver.bat -ignoreuserexclude

- Compare checksums and ignore IBM-excluded files:

  Ignore the list of IBM-excluded files and include the files in the comparison:

  - Linux ./installver.sh -ignoreibmexclude
  - Windows installver.bat -ignoreibmexclude

When you issue one of the checksum commands from the *app_server_root*/`bin` directory, the status of the command is displayed on the terminal console.

Logging results: The installver tool creates messages for each component. The tool also reports overall success based on the verification of all components in the bill of materials. The following messages indicate completion:

- I CWNVU0400I: [ivu] Total issues found : 625
- I CWNVU0340I: [ivu] Done.

The messages report the total number of issues found. If the issue count is zero, all of the components exist and no problems exist. The tool logs the results of the command to the *app_server_root*/logs/installver.log file if you use the -log parameter without specifying a file name for the log.

You can redirect the output using the -log parameter and an argument. The directory that you specify must already exist. For example:

```
./installver.sh -log /tmp/waslogs/my_installver.log
```

The following command produces this example, which shows the results of comparing the installed product against the product bill of materials.

- Linux `./installver.sh`
- Windows `installver.bat`

Example output from components with errors

This example shows errors that the comparison finds.

```
I CWNVU0160I: [ivu] Verifying.
I CWNVU0170I: [ivu] The installation root directory is C:\Program Files\eclipse\eclipse3.x\workspace\
  WAS.installver\test\v6x\product.installroot\
I CWNVU0300I: [ivu] The total number of user excluded files found are 0.
I CWNVU0300I: [ivu] The total number of IBM excluded files found are 0.
I CWNVU0180I: [ivu] Searching directory properties\version\install\6.0.0.0\backup for file listing:
  files.list
I CWNVU0260I: [ivu] The total number of components found is: 6
I CWNVU0270I: [ivu] Gathering installation root data.
W CWNVU0280W: [ivu] Component mismatch: expected mismatchcomponentname
  but found mismatchingname
I CWNVU0360I: [ivu] The following bill of materials issue is found for component
  nullvaluesample: Hash must not be null or an empty string.
I CWNVU0360I: [ivu] The following bill of materials issue is found for component
  nullvaluesample: Name must not be null or an empty string.
I CWNVU0360I: [ivu] The following bill of materials issue is found for component
  nullvaluesample: Hash must not be null or an empty string.
I CWNVU0360I: [ivu] The following bill of materials issue is found for component
  nullvaluesample: Permission must not be null or an empty string.
I CWNVU0360I: [ivu] The following bill of materials issue is found for component
```

```
  symlinksample: Hash must not be null or an empty string.
I CWNVU0290I: [ivu] Starting the verification for 6 components.

I CWNVU0470I: [ivu] Starting to analyze: _binarycomponentsample
I CWNVU0480I: [ivu] Done analyzing: _binarycomponentsample

I CWNVU0470I: [ivu] Starting to analyze: nullvaluesample
I CWNVU0430I: [ivu] The following file is missing: testpath
I CWNVU0390I: [ivu] Component issues found : 1
I CWNVU0480I: [ivu] Done analyzing: nullvaluesample

I CWNVU0470I: [ivu] Starting to analyze: overlapbinarycomponentsample
W CWNVU0422W: [ivu] The following file is overlapped: lib/binaryTest.jar
W CWNVU0425W: [ivu] The overlap is caused by: _binarycomponentsample
I CWNVU0390I: [ivu] Component issues found : 1
I CWNVU0480I: [ivu] Done analyzing: overlapbinarycomponentsample

I CWNVU0470I: [ivu] Starting to analyze: regularcomponentsample
I CWNVU0440I: [ivu] The following file is different: lib/different.jar
I CWNVU0410I: [ivu] fc19318dd13128ce14344d066510a982269c241b is the checksum in the bill of materials.
I CWNVU0420I: [ivu] 517d5a7240861ec297fa07542a7bf7470bb604fe is the checksum on the file system.
I CWNVU0440I: [ivu] The following file is different: lib/ibmtemplateexclude.jar
I CWNVU0410I: [ivu] d3ac7a4ef1a8ffb4134f2f6e7f3c0d249d74b674 is the checksum in the bill of materials.
I CWNVU0420I: [ivu] d3ac7a4ef1a838b4134f2f6e7f3c0d249d74b674 is the checksum on the file system.
I CWNVU0430I: [ivu] The following file is missing: lib/missing.jar
I CWNVU0440I: [ivu] The following file is different: lib/usertemplateexclude.jar
I CWNVU0410I: [ivu] 12dea96fec20593566ab75ff2c9949596833adc9 is the checksum in the bill of materials.
I CWNVU0420I: [ivu] 12dea96fec20593566ab75692c9949596833adc9 is the checksum on the file system.
I CWNVU0430I: [ivu] The following file is missing: missingfilebutwithbaddirectory/missingBadDirectory.jar
I CWNVU0390I: [ivu] Component issues found : 5
I CWNVU0480I: [ivu] Done analyzing: regularcomponentsample

I CWNVU0470I: [ivu] Starting to analyze: symlinksample
I CWNVU0480I: [ivu] Done analyzing: symlinksample

I CWNVU0400I: [ivu] Total issues found : 7
I CWNVU0340I: [ivu] Done.
```

Example output from a typical successful installation

This example shows typical results from checking a successful installation.

You can safely ignore issues from the prereq.wccm component. These known issues do not cause overall system issues.

Carefully examine an issue before assuming that the issue is a problem.

```
I CWNVU0160I: [ivu] Verifying.
I CWNVU0170I: [ivu] The installation root directory is C:\WebSphere\V6x-ND\
I CWNVU0300I: [ivu] The total number of user excluded files found are 1.
I CWNVU0300I: [ivu] The total number of IBM excluded files found are 75.
I CWNVU0180I: [ivu] Searching directory properties\version\install\6.0.0.0\backup for file listing: files.list
I CWNVU0260I: [ivu] The total number of components found is: 285
I CWNVU0270I: [ivu] Gathering installation root data.
I CWNVU0460I: [ivu] The utility is running.
I CWNVU0460I: [ivu] The utility is running.
I CWNVU0290I: [ivu] Starting the verification for 285 components.

I CWNVU0470I: [ivu] Starting to analyze: activity
I CWNVU0480I: [ivu] Done analyzing: activity

I CWNVU0470I: [ivu] Starting to analyze: activity.impl
I CWNVU0480I: [ivu] Done analyzing: activity.impl

I CWNVU0470I: [ivu] Starting to analyze: activity.session
I CWNVU0480I: [ivu] Done analyzing: activity.session
```

```
I CWNVU0470I: [ivu] Starting to analyze: activity.session.impl
I CWNVU0480I: [ivu] Done analyzing: activity.session.impl

I CWNVU0470I: [ivu] Starting to analyze: acwa
I CWNVU0480I: [ivu] Done analyzing: acwa

I CWNVU0470I: [ivu] Starting to analyze: adapter
I CWNVU0480I: [ivu] Done analyzing: adapter
...
I CWNVU0470I: [ivu] Starting to analyze: prereq.wccm
I CWNVU0430I: [ivu] The following file is missing: web/configDocs/activitysessionejbext/
  ActivitySessionEJBJarExtension.html
I CWNVU0430I: [ivu] The following file is missing: web/configDocs/activitysessionejbext/
  ActivitySessionEnterpriseBeanExtension.html
I CWNVU0430I: [ivu] The following file is missing: web/configDocs/activitysessionejbext/
  ContainerActivitySession.html
I CWNVU0430I: [ivu] The following file is missing: web/configDocs/activitysessionejbext/
  classes-frame.html
I CWNVU0430I: [ivu] The following file is missing: web/configDocs/activitysessionservice/
  ActivitySessionService.html

...
I CWNVU0430I: [ivu] The following file is missing: web/configDocs/wssecurity/consumer-binding.html
I CWNVU0430I: [ivu] The following file is missing: web/configDocs/wssecurity/default-bindings.html
I CWNVU0430I: [ivu] The following file is missing: web/configDocs/wssecurity/generator-binding.html
I CWNVU0390I: [ivu] Component issues found : 625
I CWNVU0480I: [ivu] Done analyzing: prereq.wccm

I CWNVU0470I: [ivu] Starting to analyze: prereq.webtlsrn
I CWNVU0480I: [ivu] Done analyzing: prereq.webtlsrn

I CWNVU0470I: [ivu] Starting to analyze: prereq.wsadie
I CWNVU0480I: [ivu] Done analyzing: prereq.wsadie

...
I CWNVU0470I: [ivu] Starting to analyze: wsexception
I CWNVU0480I: [ivu] Done analyzing: wsexception

I CWNVU0470I: [ivu] Starting to analyze: wsif
I CWNVU0480I: [ivu] Done analyzing: wsif

I CWNVU0400I: [ivu] Total issues found : 627
I CWNVU0340I: [ivu] Done.
```

See the description of the “installver command” on page 38 for explanations of each parameter.

Go to “Computing a new baseline checksum for an inventory of configured files” to learn more about the installver command.

Computing a new baseline checksum for an inventory of configured files

After installation, you can verify the actual checksums of installed files against a bill of materials that ships with the product. After configuring your system, create a new checksum so that you can compare the system periodically to the new checksum. Use the result to analyze changes to your configured system.

After configuring the product, save a new baseline checksum to establish a new checksum standard for your system.

This topic describes using the installver command to create and compare an inventory of configured files to the currently installed files.

The installver tool can compute a new baseline checksum for the inventory of all files in the installation root directory. The tool stores the new checksum by default in the sys.inv file within the current working directory. You can specify a different file path and file name. Create the file outside of the installation root directory or exclude the file from comparisons.

Later, compare the checksums in the sys.inv file (or the file that you specified when creating the inventory) to the checksums of the currently installed files to see what files have changed.

The baseline checksum report identifies missing files, additional files, and changed files.

- Create an inventory list of the files that are currently installed in the installation root directory:
  - Linux `./installver.sh -createinventory`
  - Windows `installver.bat -createinventory`

  For example, the following messages might display on a Windows system when you issue the `installver.bat -createinventory` command to create the default *app_server_root*\bin\sys.inv file:

```
W CWNVU0320W: [ivu] The
C:\IBM\WebSphere\AppServerV602\bin\sys.inv
inventory file is within the product installation root directory:
C:\IBM\WebSphere\AppServerV602.

Create the file outside of the installation root directory to omit the file from the verification.
I CWNVU0300I: [ivu] The total number of user excluded files found are 2.
I CWNVU0300I: [ivu] The total number of IBM excluded files found are 78.
I CWNVU0310I: [ivu] Creating the following inventory file:
C:\IBM\WebSphere\AppServerV602\bin\sys.inv
I CWNVU0460I: [ivu] The utility is running.
```

  When it finishes running, a completion message is displayed:

```
 I CWNVU0340I: [ivu] Done.
```

  The sys.inv contains the new inventory:

```
#C:\IBM\WebSphere\AppServer\
#2005.10.10_06.24.06PM_EDT
#user_ID
#-createinventory -log
241fe4e309abfd8f2c5911216dbabd61dd4751a6
   |_jvm\bin\appletviewer.exe
   |42032
   |2004.10.28 05.37.02AM EDT
e00c6ea688ab67e004ec6cfac26ec48541a5b9ff
   |_jvm\bin\dbghelp.dll
   |712192
   |2004.10.28 05.36.50AM EDT
916e244deeb44b9d3218aafa3b56c8680aa31f2f
   |_jvm\bin\extcheck.exe
   |42040
   |2004.10.28 05.37.02AM EDT
...
7fc3bb38e8b90fed05cd0440953000c2cc965b44
   |web\spidocs\stylesheet.css
   |1240
   |2005.10.09 12.14.17AM EDT
22706a0d900c52f1c015c870ddeee25581c5d57b
   |web\spidocs\toHTML\index.html
   |867
   |2005.10.09 12.14.17AM EDT
```

- Create the inventory file in a directory outside of the installation root directory to exclude the inventory file from the comparison.
  - Linux `./installver.sh -createinventory /tmp/system.inv`
  - Windows `installver.bat -createinventory "C:\temp\system.inv"`
- Compare the inventory list to files that are currently installed in the installation root directory:

  - Linux `./installver.sh -compare`
  - Windows `installver.bat -compare`

  If you created the inventory file somewhere other than the default location, use the following syntax:

  - Linux `./installver.sh -compare /tmp/system.inv`
  - Windows `installver.bat -compare "C:\temp\system.inv"`

- Compare and display trace results:
  - Linux `./installver.sh -compare -trace`
  - Windows `installver.bat -compare -trace`
- Compare and exclude specified files from the inventory comparison:
  - Linux `./installver.sh -compare -exclude` *fn1;fn2;fn3;...*
  - Windows `installver.bat -compare -exclude` *fn1;fn2;fn3;...*
- Compare and include only specified files in the inventory comparison:
  - Linux `./installver.sh -compare -include` *fn1;fn2;fn3;...*
  - Windows `installver.bat -compare -include` *fn1;fn2;fn3;...*

When you issue an installver commands from the *app_server_root*/`bin` directory, the status of the command is displayed on the terminal console. To create a log, use the -log parameter.

See the description of the “installver command” on page 38 for explanations of each parameter.

Go to “Excluding files from a checksum comparison” to learn more about the installver command.

Excluding files from a checksum comparison

Specify individual files to exclude from a comparison, specify individual components to exclude, or create a single configurable properties file to specify a list of files to exclude from the bill of materials verification.

Install the product before comparing checksums and using exclusion properties.

This topic describes how to use exclusion properties.

By default, IBM excludes some files from the checksum comparison. You can also exclude files. The number of files excluded is reported within the first few messages:

```
I CWNVU0160I: [ivu] Verifying.
I CWNVU0170I: [ivu] The installation root directory is C:\IBM\WebSphere\AppServerV602\
I CWNVU0300I: [ivu] The total number of user excluded files found are 0.
I CWNVU0300I: [ivu] The total number of IBM excluded files found are 78.
I CWNVU0180I: [ivu] Searching directory properties\version\install\6.0.0.0\
                    backup for file listing: files.list
I CWNVU0260I: [ivu] The total number of components found is: 285
I CWNVU0270I: [ivu] Gathering installation root data.
I CWNVU0460I: [ivu] The utility is running.
I CWNVU0290I: [ivu] Starting the verification for 285 components.

I CWNVU0470I: [ivu] Starting to analyze: activity
I CWNVU0480I: [ivu] Done analyzing: activity
```

...

Several different methods are provided to exclude files from the comparison.

- Compare checksums and exclude all of the files within one or more components from the comparison:
  - Linux ./installver.sh -excludecomponent *comp1;comp2;comp3;...*

- Windows installver.bat -excludecomponent *comp1;comp2;comp3;...*

For example, you might exclude the prereq.wccm component to avoid known but acceptable issues in the component:

```
./installver.sh -log -excludecomponent prereq.wccm
```

The resulting messages show the exclusion:

```
I CWNVU0160I: [ivu] Verifying.
I CWNVU0170I: [ivu] The installation root directory is /opt/IBM/WebSphere/AppServerV602/
I CWNVU0300I: [ivu] The total number of user excluded files found are 0.
I CWNVU0300I: [ivu] The total number of IBM excluded files found are 78.
I CWNVU0180I: [ivu] Searching directory properties/version/install/6.0.0.0/
                    backup for file listing: files.list
I CWNVU0260I: [ivu] The total number of components found is: 285
I CWNVU0270I: [ivu] Gathering installation root data.
I CWNVU0460I: [ivu] The utility is running.
I CWNVU0290I: [ivu] Starting the verification for 284 components.
...
I CWNVU0400I: [ivu] Total issues found : 0
I CWNVU0340I: [ivu] Done.
```

- Exclude certain files from the comparison:
  - Linux *app_server_root*`/bin/installver.sh -exclude` *fn1;fn2;fn3*
  - Windows *app_server_root*`\bin\installver.bat -exclude` *fn1;fn2;fn3*

For example, suppose that you include only the prereq.wccm component for comparison:

```
installver.bat -log -includecomponent prereq.wccm
```

The comparison shows all of the missing file issues in the component:

```
I CWNVU0160I: [ivu] Verifying.
I CWNVU0170I: [ivu] The installation root directory is C:\IBM\WebSphere\AppServerV602\
I CWNVU0300I: [ivu] The total number of user excluded files found are 0.
I CWNVU0300I: [ivu] The total number of IBM excluded files found are 78.
I CWNVU0180I: [ivu] Searching directory properties\version\install\6.0.0.0\
                    backup for file listing: files.list
I CWNVU0260I: [ivu] The total number of components found is: 285
I CWNVU0270I: [ivu] Gathering installation root data.
I CWNVU0460I: [ivu] The utility is running.
I CWNVU0290I: [ivu] Starting the verification for 1 components.

I CWNVU0470I: [ivu] Starting to analyze: prereq.wccm
I CWNVU0430I: [ivu] The following file is missing:
web/configDocs/activitysessionejbext/ActivitySessionEJBJarExtension.html
I CWNVU0430I: [ivu] The following file is missing:
web/configDocs/activitysessionejbext/ActivitySessionEnterpriseBeanExtension.html
I CWNVU0430I: [ivu] The following file is missing:
web/configDocs/activitysessionejbext/ContainerActivitySession.html
...
I CWNVU0390I: [ivu] Component issues found : 625
I CWNVU0480I: [ivu] Done analyzing: prereq.wccm

I CWNVU0400I: [ivu] Total issues found : 625
I CWNVU0340I: [ivu] Done.
```

You can exclude files. For example, exclude the highlighted files in the previous example:

```
installver -log -includecomponent prereq.wccm
 -exclude web/configDocs/activitysessionejbext/
ActivitySessionEJBJarExtension.html;web\configDocs\activitysessionejbext\
ActivitySessionEnterpriseBeanExtension.html
```

Tip: Windows Use Windows-style slashes or UNIX-style slashes to delimit directories.

The result shows that the excluded files were not compared:

```
I CWNVU0160I: [ivu] Verifying.
I CWNVU0170I: [ivu] The installation root directory is C:\IBM\WebSphere\AppServerV602\
I CWNVU0300I: [ivu] The total number of user excluded files found are 0.
I CWNVU0300I: [ivu] The total number of IBM excluded files found are 78.
I CWNVU0180I: [ivu] Searching directory properties\version\install\6.0.0.0\
                    backup for file listing: files.list
I CWNVU0260I: [ivu] The total number of components found is: 285
I CWNVU0270I: [ivu] Gathering installation root data.
I CWNVU0460I: [ivu] The utility is running.
I CWNVU0290I: [ivu] Starting the verification for 1 components.

I CWNVU0470I: [ivu] Starting to analyze: prereq.wccm
I CWNVU0430I: [ivu] The following file is missing:
web/configDocs/activitysessionejbext/ContainerActivitySession.html
...
I CWNVU0390I: [ivu] Component issues found : 623
I CWNVU0480I: [ivu] Done analyzing: prereq.wccm

I CWNVU0400I: [ivu] Total issues found : 623
I CWNVU0340I: [ivu] Done.
```

If the two files were in the comparison, they would be in the list and the count would be 625, as in the previous example.

Tip: The highlighted line in the example is reserved for excluded files listed in the user template file, as described in the next step. The highlighted line does not count files that you list in the installver command line with the -exclude parameter.

- Compare checksums and exclude certain files from the comparison by creating and using a user template file:

  A single user configurable properties file is available to specify a list of files to exclude from the bill of materials verification.

  1. Create an empty template file.
     - Linux *app_server_root*/bin/installver.sh -createtemplate *template_name*
     - Windows *app_server_root*\bin\installver.bat -createtemplate *template_name*

     For example, create the default user template file on a Windows system:

     ```
     installver -createtemplate
     I CWNVU0200I: [ivu] Creating template:
     C:\IBM\WebSphere\AppServerV602\profiles\
     Dmgr01\properties\ivu_user.template
     I CWNVU0340I: [ivu] Done.
     ```

     The ivu_user.template file is created in the properties directory of the default profile, which in this case, is a deployment manager profile.

     The -template_name parameter is optional. However, a template file must reside in the properties directory of the default profile, such as the *app_server_root*/profiles/Dmgr01/properties directory.
  2. List files to exclude in the templates file.

     The properties file has the following format:

     ```
     <template>
      <componentfiles componentname="name_of_component">
        <file>
          <relativepath action="exclude">file_name</relativepath>
        </file>
      </componentfiles>
     </template>
     ```

     For example, list the component and files from the previous example:

     ```
     <template>
      <componentfiles componentname="prereq.wccm">
       <file>
        <relativepath action="exclude">
     ```

```
web/configDocs/activitysessionejbext/ActivitySessionEnterpriseBeanExtension.html
   </relativepath>
  </file>
  <file>
   <relativepath action="exclude">
web/configDocs/activitysessionejbext/ActivitySessionEJBJarExtension.html
   </relativepath>
  </file>
 </componentfiles>
</template>
```

Tip: Do not use quotation marks or double quotation marks to delimit a file name.

3. Use the template file to exclude files from the comparison:

   ```
   installver -log
   ```

   If the ivu_user.template file exists in the properties directory of the default profile, the installver command uses it.

   The result shows that some user files are excluded:

   ```
   I CWNVU0160I: [ivu] Verifying.
   I CWNVU0170I: [ivu] The installation root directory is C:\IBM\WebSphere\AppServerV602\
   I CWNVU0300I: [ivu] The total number of user excluded files found are 2.
   I CWNVU0300I: [ivu] The total number of IBM excluded files found are 78.
   I CWNVU0180I: [ivu] Searching directory properties\version\install\6.0.0.0\
                       backup for file listing: files.list
   I CWNVU0260I: [ivu] The total number of components found is: 285
   I CWNVU0270I: [ivu] Gathering installation root data.
   I CWNVU0460I: [ivu] The utility is running.
   I CWNVU0290I: [ivu] Starting the verification for 1 components.

   I CWNVU0470I: [ivu] Starting to analyze: prereq.wccm
   I CWNVU0430I: [ivu] The following file is missing:
   web/configDocs/activitysessionejbext/ContainerActivitySession.html

   ...
   I CWNVU0430I: [ivu] The following file is missing:
   web/configDocs/wssecurity/generator-binding.html
   I CWNVU0390I: [ivu] Component issues found : 623
   I CWNVU0480I: [ivu] Done analyzing: prereq.wccm

   I CWNVU0400I: [ivu] Total issues found : 623
   I CWNVU0340I: [ivu] Done.
   ```

When you issue one of the checksum commands from the *app_server_root*/`bin` directory, the status of the command is displayed on the terminal console or in a log file.

Go to “Comparing specific file and component checksums” to learn more about the installver command.

Comparing specific file and component checksums

Specify individual files or components to include in the bill of materials verification.

Install the product before comparing checksums of individual files and components.

This topic describes how to use inclusion properties to specify individual files and components.

By default, IBM includes all files in the checksum comparison except for the 78 IBM excluded files.

```
 I CWNVU0160I: [ivu] Verifying.
 I CWNVU0170I: [ivu] The installation root directory is C:\IBM\WebSphere\AppServerV602\
 I CWNVU0300I: [ivu] The total number of user excluded files found are 0.
 I CWNVU0300I: [ivu] The total number of IBM excluded files found are 78.
 I CWNVU0180I: [ivu] Searching directory properties\version\install\6.0.0.0\
                     backup for file listing: files.list
```

```
I CWNVU0260I: [ivu] The total number of components found is: 285
I CWNVU0270I: [ivu] Gathering installation root data.
I CWNVU0460I: [ivu] The utility is running.
I CWNVU0290I: [ivu] Starting the verification for 285 components.

I CWNVU0470I: [ivu] Starting to analyze: activity
I CWNVU0480I: [ivu] Done analyzing: activity

...
```

Several different methods are provided to include only certain files in the comparison.

- Compare checksums and include specified components only in the comparison:
  - Linux ./installver.sh -includecomponent *comp1;comp2;comp3;...*
  - Windows installver.bat -includecomponent *comp1;comp2;comp3;...*

  For example, you might include the activity component:
  - Linux ./installver.sh -log -includecomponent activity
  - Windows installver.bat -log -includecomponent activity

  The resulting messages show the inclusion:

  ```
  I CWNVU0160I: [ivu] Verifying.
  I CWNVU0170I: [ivu] The installation root directory is C:\IBM\WebSphere\AppServerV602\
  I CWNVU0300I: [ivu] The total number of user excluded files found are 0.
  I CWNVU0300I: [ivu] The total number of IBM excluded files found are 78.
  I CWNVU0180I: [ivu] Searching directory properties\version\install\6.0.0.0\
                      backup for file listing: files.list
  I CWNVU0260I: [ivu] The total number of components found is: 285
  I CWNVU0270I: [ivu] Gathering installation root data.
  I CWNVU0460I: [ivu] The utility is running.
  I CWNVU0460I: [ivu] The utility is running.
  I CWNVU0290I: [ivu] Starting the verification for 1 components.

  I CWNVU0470I: [ivu] Starting to analyze: activity
  I CWNVU0480I: [ivu] Done analyzing: activity

  I CWNVU0400I: [ivu] Total issues found : 0
  I CWNVU0340I: [ivu] Done.
  ```
- Include only certain files in the comparison:
  - Linux *app_server_root*/bin/installver.sh -include *fn1;fn2;fn3*
  - Windows *app_server_root*\bin\installver.bat -include *fn1;fn2;fn3*

  For example, you might include only the properties/version/proxy.server.component file, which was changed to generate the checksum difference in this example.

  ```
  installver.bat -log
   -include properties/version/proxy.server.component
  ```

  The result shows that the included file was in the comparison, which scanned 285 components looking for all components that refer to the file:

  ```
  I CWNVU0160I: [ivu] Verifying.
  I CWNVU0170I: [ivu] The installation root directory is C:\IBM\WebSphere\AppServerV602\
  I CWNVU0300I: [ivu] The total number of user excluded files found are 0.
  I CWNVU0300I: [ivu] The total number of IBM excluded files found are 78.
  I CWNVU0180I: [ivu] Searching directory properties\version\install\6.0.0.0\
                      backup for file listing: files.list
  I CWNVU0260I: [ivu] The total number of components found is: 285
  I CWNVU0270I: [ivu] Gathering installation root data.
  I CWNVU0460I: [ivu] The utility is running.
  I CWNVU0290I: [ivu] Starting the verification for 285 components.

  I CWNVU0470I: [ivu] Starting to analyze: activity
  I CWNVU0480I: [ivu] Done analyzing: activity
  ```

```
...
I CWNVU0470I: [ivu] Starting to analyze: proxy.server
I CWNVU0440I: [ivu] The following file is different: properties/version/
                    proxy.server.component
I CWNVU0410I: [ivu] f385fc95977092e0482d52f9d1d5bebbc39fbb10 is the checksum
                    in the bill of materials.
I CWNVU0420I: [ivu] b43bda7f1e7202d1f9495fc74ac14b8d85830aab is the checksum
                    on the file system.
I CWNVU0390I: [ivu] Component issues found : 1
I CWNVU0480I: [ivu] Done analyzing: proxy.server
...
I CWNVU0400I: [ivu] Total issues found : 1
I CWNVU0340I: [ivu] Done.
```

If you know that a file is in only one component, you can speed up the comparison by restricting the comparison of the file to the relevant component:

```
installver.bat -log -includecomponent proxy.server -include properties/version/
   proxy.server.component
```

The result shows the comparison was restricted to one component:

```
I CWNVU0160I: [ivu] Verifying.
I CWNVU0170I: [ivu] The installation root directory is C:\IBM\WebSphere\AppServerV602\
I CWNVU0300I: [ivu] The total number of user excluded files found are 0.
I CWNVU0300I: [ivu] The total number of IBM excluded files found are 78.
I CWNVU0180I: [ivu] Searching directory properties\version\install\6.0.0.0\
                    backup for file listing: files.list
I CWNVU0260I: [ivu] The total number of components found is: 285
I CWNVU0270I: [ivu] Gathering installation root data.
I CWNVU0460I: [ivu] The utility is running.
I CWNVU0290I: [ivu] Starting the verification for 1 components.

I CWNVU0470I: [ivu] Starting to analyze: proxy.server
I CWNVU0440I: [ivu] The following file is different: properties/version/
                    proxy.server.component
I CWNVU0410I: [ivu] f385fc95977092e0482d52f9d1d5bebbc39fbb10 is the checksum
                    in the bill of materials.
I CWNVU0420I: [ivu] b43bda7f1e7202d1f9495fc74ac14b8d85830aab is the checksum
                    on the file system.
I CWNVU0390I: [ivu] Component issues found : 1
I CWNVU0480I: [ivu] Done analyzing: proxy.server

I CWNVU0400I: [ivu] Total issues found : 1
I CWNVU0340I: [ivu] Done.
```

When you issue one of the checksum commands from the *app_server_root*/`bin` directory, the status of the command is displayed on the terminal console or in a log file.

Go to “Changing the default message digest algorithm” to learn more about the installver command.

Changing the default message digest algorithm

Change the default message digest algorithm from SHA to MD5 only if absolutely necessary. Edit the installver.bat file or the installver.sh file to make the change. Changing the algorithm invalidates the SHA-based checksums in the product bill of materials. For this reason, verify the product files before changing the message digest algorithm.

Install the product before attempting to change the default message digest algorithm from SHA to MD5.

The default message digest algorithm is one of the secure hash algorithms (SHA) that are part of the Secure Hash Standard (SHS) from the National Institute of Standards and Technology (NIST). See the Federal Information Processing Standards (FIPS) publication 180-2 for more information. SHA-1 is the standard hash function of the U.S. government.

Also available is the older MD5 message digest algorithm. MD5 is a deprecated type of message algorithm that is not as secure as SHA and is provided only for backward compatibility.

This topic describes changing the default message digest algorithm. You must edit the installver.bat file or the installver.sh file to change the algorithm.

Add the following environmental property to the script file:

```
-Dchecksum.type=MD5
```

The default value is:

```
-Dchecksum.type=SHA
```

Go to “Handling out-of-memory situations” to learn more about the installver command.

Handling out-of-memory situations

Memory requirements for using the installver command are related to the size of the installed file set for the product. For the basic verification scenario, comparing an installed file set with the provided bill of materials might require a maximum heap size of 128 MB to 256 MB.

Install the product at the V6.0.2 level or later to install the installver command tool. After installing the command, you can increase the maximum heap size setting.

If you need more memory for either a product verification or a baseline checksum verification, increase the maximum heap size setting for your Java virtual machine (JVM) by including a setting in the installver command script.

- Edit the installver command script:
  - Linux Edit the *app_server_root*/bin/installver.sh file.
  - Windows Edit the *app_server_root*\bin\installver.bat file.
- Add or increase the maximum heap size setting:
  - Linux Change the following line:

    ```
    "$JAVA_HOME"/bin/java \
    ```

    to:

    ```
    "$JAVA_HOME"/bin/java -Xmx256M \
    ```

  - Windows Change the following line:

    ```
    "%JAVA_HOME%\bin\java" "-Dproduct.home=%WAS_HOME%"
    ```

    to:

    ```
     "%JAVA_HOME%\bin\java" -Xmx256M "-Dproduct.home=%WAS_HOME%"
    ```

After you change the setting, run the installver command to verify that it works correctly.

Go to “Verifying the installver command” to learn more about the installver command.

Verifying the installver command

Use the verifyinstallver command to perform a checksum on the files that comprise the installver command.

Install the product at the V6.0.2 level or later before attempting to verify the checksums of the files that comprise the installver command.

This topic describes using the verifyinstallver command to compute a checksum on the installed files that comprise the installver command. Verifying the checksums of the command files ensures that the command can function properly.

1. Change directories to the *app_server_root*/bin directory.
   - Linux `cd app_server_root/bin`
   - Windows `cd app_server_root\bin`
2. Start the verifyinstallver command script from the command line.
   - Linux `./verifyinstallver.sh`
   - Windows `verifyinstallver.bat`

   You can use the -trace parameter to show verbose results. You can use the -help parameter to see usage information.

If the command successfully verifies the installver files, you see messages that are similar to the following example:

```
I CWNVU0160I: [ivu] Verifying.
I CWNVU0170I: [ivu] The installation root directory is C:\WebSphere\V6X-ND\
I CWNVU0300I: [ivu] The total number of user excluded files found are 0.
I CWNVU0300I: [ivu] The total number of IBM excluded files found are 75.
I CWNVU0180I: [ivu] Searching directory properties\version\install\6.0.0.0\backup for file listing:
 files.list
I CWNVU0260I: [ivu] The total number of components found is: 285
I CWNVU0270I: [ivu] Gathering installation root data.
I CWNVU0460I: [ivu] The utility is running.
I CWNVU0290I: [ivu] Starting the verification for 1 components.

I CWNVU0470I: [ivu] Starting to analyze: installver
I CWNVU0480I: [ivu] Done analyzing: installver

I CWNVU0400I: [ivu] Total issues found : 0
I CWNVU0340I: [ivu] Done.
```

Go to the description of the “verifyinstallver command” to learn more.

verifyinstallver command

The verifyinstallver command is a verification program that performs a checksum on the files that comprise the installver command.

Purpose

This topic describes the command-line syntax for the verifyinstallver command. Start the verifyinstallver command script from the command line. This topic describes how to locate the command and use it to verify that the installver command is valid.

Use the following syntax for the command to automatically check the bill of materials against the installed file system:

- Linux *app_server_root*/bin/verifyinstallver.sh
- Windows *app_server_root*\bin\verifyinstallver.bat

Display verbose results:

```
verifyinstallver.bat -trace
```

Display usage information:

```
verifyinstallver.bat -help
```

The location of the verification program for installver command is:

- Linux *app_server_root*/bin/verifyinstallver.sh
- Windows *app_server_root*\bin\verifyinstallver.bat

Parameters

The command has optional parameters.

path_name_and_file_name
Optional parameter to identify the location of the installver command file.

The default location and filename are:

- Linux *app_server_root*/bin/installver.sh
- Windows *app_server_root*\bin\installver.bat

-help
Displays usage information.

-trace
Provides trace output of what the tool checks and what the tool discovers.

Next

Go to the description of the “installver command” to learn more.

installver command

The installver command performs two main functions. The tool computes a checksum on the installed files and compares the checksum to the shipped bill of materials for the product. The installver command can also compute a new baseline checksum for each file in the inventory of a configured system to use to identify file changes in later comparisons.

Purpose

This topic describes the location of the command file, command syntax, and links to task information about using the tool.

The default log file is the *app_server_root*/logs/installver.log file. You can redirect the output using the -log parameter and an argument. Use the -log parameter without the file argument to generate the default log file.

Computing the checksum

The installver tool computes a checksum for each installed file in the product. The tool compares each computed checksum to the correct checksum for the file. The correct checksums are shipped in the bill-of-material files. One bill-of-materials file exists for each component.

The tool parses the bill-of-materials file for each component to find the correct checksum value for each file in the component. Each product file has an entry in some bill-of-materials file. The entry for a product file lists the product file path and the correct checksum value.

Shipped bill-of-material files

Each bill-of-materials file is named files.list. Each component has one files.list file. Each files.list file is in one of the *app_server_root*/properties/version/install/6.1.0.0/backup/ *component_name* directories. A *component_name* directory exists for each component.

For instance, the files.list file for the activity component is in the *app_server_root* /properties/version/install/6.1.0.0/backup/activity directory. The file resembles the following example:

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

<componentfiles componentname="activity">
       <file>
              <relativepath>properties/version/activity.component</relativepath>
              <checksum>1a20dc54694e81fccd16c80f7c1bb6b46bba8768</checksum>
              <permissions>644</permissions>
              <installoperation>remove</installoperation>
       </file>
       <file>
              <relativepath>lib/activity.jar</relativepath>
              <checksum>2f056cc01be7ff42bb343e962d26328d5332c88c</checksum>
              <permissions>644</permissions>
              <installoperation>remove</installoperation>
       </file>
</componentfiles>
```

Comparing the computed checksum to the correct checksum

As the tool processes each product file in each bill-of-materials file, the tool also computes the actual checksum value of the corresponding installed product file. The tool then compares the checksum of the product file to the correct checksum value in the bill-of-materials file. The tool then reports any differences.

Location of the command file

The installver command file is located in the `bin` directory of the installation root directory:

- ▶ Linux `*app_server_root*/bin/installver.sh`
- ▶ Windows `*app_server_root*\bin\installver.bat`

Change directories to the `bin` directory to start the installver tool from the command line. The tool runs on any supported operating system except z/OS.

For example, use the following command to run the tool from a user profile that has *ALLOBJ authority on iSeries systems:

```
installver
```

Or use the following command to start the tool on a Linux system or a UNIX system:

```
./installver.sh
```

See the following topics for more information about the purpose of the command:

- "Verifying against the bill of materials" on page 25
- "Computing a new baseline checksum for an inventory of configured files" on page 28
- "Excluding files from a checksum comparison" on page 30
- "Comparing specific file and component checksums" on page 33
- "Changing the default message digest algorithm" on page 35
- "Handling out-of-memory situations" on page 36
- "Verifying the installver command" on page 36

Syntax for displaying usage information

- ▶ Linux `./installver.sh -help`
- ▶ Windows `installver.bat -help`

Syntax for listing all components

- Linux ./installver.sh -listcomponents
- Windows installver.bat -listcomponents

Syntax for comparing product files to the bill-of-material files

Use the following command syntax to automatically check the bill of materials against the installed file system.

- Linux *app_server_root*/bin/installver.sh
- Windows *app_server_root*\bin\installver.bat

See “Verifying against the bill of materials” on page 25 for examples of using the command to compare the installed files to the product bill-of-materials files.

Example comparisons and command usage

Compare checksums and include specified files only in the comparison:

- Linux ./installver.sh -include *fn1;fn2;fn3*
- Windows installver.bat -include *fn1;fn2;fn3*

See “Comparing specific file and component checksums” on page 33 for examples of using the command to compare only files or components that you specify.

Compare checksums and include specified components only in the comparison:

- Linux ./installver.sh -includecomponent *comp1;comp2;comp3;...*
- Windows installver.bat -includecomponent *comp1;comp2;comp3;...*

Compare checksums and exclude certain components from the comparison:

- Linux ./installver.sh -excludecomponent *comp1;comp2;comp3;...*
- Windows installver.bat -excludecomponent *comp1;comp2;comp3;...*

See “Excluding files from a checksum comparison” on page 30 for examples of using the command to exclude files from the comparison.

Compare checksums and ignore user-excluded files:

- Linux ./installver.sh -ignoreuserexclude
- Windows installver.bat -ignoreuserexclude

Compare checksums and ignore IBM-excluded files:

- Linux ./installver.sh -ignoreibmexclude
- Windows installver.bat -ignoreibmexclude

List all components only:

- Linux ./installver.sh -listcomponents
- Windows installver.bat -listcomponents

Create template (for listing excluded files) only:

- Linux ./installver.sh -createtemplate
- Windows installver.bat -createtemplate

Parameters for comparing checksums against the bill of materials

The following parameters are associated with the command when comparing product file checksums to the correct checksums in the bill of material files.

-componentdir *directory_1***;** *directory_2***;** *directory_n*

Optional parameter that identifies the names of directories where WebSphere Application Server products store the individual bill-of-material lists for each component.

The default value is the *app_server_root* /properties/version/install/6.1.0.0/backup directory.

The directory path delimiter can be either a colon (:) or a semi-colon (;).

-createtemplate [*file_name* **]**

Creates a template properties file for excluding files from the checksum comparison. Edit the template properties file to add a line for each file that you want to exclude from verification.

Without a file specification argument, the installver tool creates the *profile_root*/properties/ivu_user.template file.

If you specify a file name, the installver tool creates the file in the working directory, which is the *profile_root*/bin directory by default.

- Linux Perform the following procedure:
  1. cd *app_server_root*/bin
  2. ./installver.sh -createtemplate
- Windows Perform the following procedure:
  1. cd *app_server_root*\bin
  2. installver.bat -createtemplate

The installver tool creates the template properties file in the properties directory of the default profile.

- Linux *profile_root*/properties/ivu.user.template
- Windows *profile_root*\properties\ivu.user.template

-exclude *file1;file2;file3; ...*

Excludes files from verification.

Use a semi-colon (;) or a colon (:) to delimit file names.

-excludecomponent *component1;component2;component3; ...*

Excludes components from verification.

Use a semi-colon (;) or a colon (:) to delimit component names.

-filelist *file_name*

Optional parameter that identifies the name of the file that IBM uses to identify the correct checksums of product files in a particular product component.

The default value is files.list.

-help

Displays usage information.

-ignoreuserexclude

Ignores the default *profile_root*/properties/ivu_user.template file, if the file exists and compares the files listed in the template.

If you use the -createtemplate parameter with a file specification to create a template file in another location, the -ignoreusertemplate parameter has no effect.

-ignoreibmexclude

Compares checksums for all of the files in the installation root directory. IBM specifies certain files to exclude from the verification by default. You can cause the installver tool to verify those files as well by using the -ignoreibmexclude parameter.

-include *file1;file2;file3; ...*

Includes files in the verification and excludes all other files.

Use a semi-colon (;) or a colon (:) to delimit file names.

-includecomponent *component1;component2;component3; ...*

Includes components in the verification and excludes all other components.

Use a semi-colon (;) or a colon (:) to delimit component names.

-installroot *directory_name*

Overrides the default installation root directory.

-listcomponents

Displays a list of components in the product. Each component must have a files.list file.

-log [*file_path_and_file_name_of_log_file***]**

The default log file is the *app_server_root*/logs/installver.log file. You can redirect the output using the -log parameter and an argument.

-profilehome *directory_name*

Overrides the default profiles directory in the installation root directory.

-trace

Provides trace output of what the tool checks and what the tool discovers.

Syntax for creating and using a new baseline checksum for an inventory of configured files

Use the following syntax to create and compare an inventory of configured files to the currently installed files.

Create an inventory list of the files that are currently installed in the installation root directory

- Linux `./installver.sh -createinventory` [*path/file_name*], such as `./installver.sh -createinventory /tmp/system.inv`
- Windows `installver.bat -createinventory` [*path\file_name*], such as `installver.bat -createinventory C:\temp\system.inv`

Compare the inventory list to files that are currently installed in the installation root directory

- Linux `./installver.sh -compare` */path/file_name*
- Windows `installver.bat -compare` *path\file_name*

Compare and display trace results

- Linux `./installver.sh -compare` */path/file_name* `-trace`
- Windows `installver.bat -compare` */path/file_name* `-trace`

Display usage information

- Linux `./installver.sh -help`
- Windows `installver.bat -help`

Compare and exclude specified files from the inventory comparison

- Linux `./installver.sh -compare  /path/file_name -exclude fn1;fn2;fn3;...`
- Windows `installver.bat -compare \path\file_name -exclude fn1;fn2;fn3;...`

Compare and include only specified files in the inventory comparison

- Linux `./installver.sh -compare /path/file_name -include fn1;fn2;fn3;...`
- Windows `installver.bat -compare /path/file_name-include fn1;fn2;fn3;...`

Parameters for creating and using checksums for a file inventory

The following parameters are associated with this command.

-compare *file_path_and_file_name_of_existing_inventory_file*
Compares the existing inventory list to the existing files to determine changes.

First use the -createinventory parameter to create an inventory list. Then use the -compare parameter to compare the inventory list to the actual files that exist in the system at the time of the comparison.

The result of the comparison shows changed classes, changed files, missing files, and added files. Such a comparison is very useful for verifying the absence of virus files, for example.

-createinventory *directory_name*
Creates the new checksum by default in the sys.inv file within the current working directory, such as the *profile_root*/bin directory. You can specify a file path and file name. Create the file outside of the installation root directory or exclude the file from comparisons.

You can point the installver tool at any directory. The default directory is the installation root directory.

You can exclude files or components from the inventory.

The installver tool computes a checksum for each file. Each file entry in the inventory has the following general pattern:

```
checksum|relativepath/file_name|file_size|last_modfiied_time
```

After creating an inventory list, use the -compare parameter to compare the list to the actual files that exist in the system at the time of the comparison.

-exclude *file1;file2;file3;...*
Excludes files from comparison.

Use a semi-colon (;) or a colon (:) to delimit file names.

-help
Displays usage information.

-include *file1;file2;file3; ...*
Includes files in the comparison and excludes all other files.

Use a semi-colon (;) or a colon (:) to delimit file names.

-installroot *directory_name*
Overrides the *default installation root directory.*

-log [*file_path_and_file_name_of_log_file***]**

The default log file is the *app_server_root*/logs/installver.log file. You can redirect the output using the -log parameter and an argument.

-trace
Provides trace output of what the tool checks and what the tool discovers.

Example issues

The following examples show issues that might occur when you run the installver command to compare checksums.

Ignore entries for checksum mismatches that you introduce on purpose, such as might occur when you extend a component

The checksums differ for each file that you change:

```
I CWNVU0470I: [ivu] Starting to analyze: regularcomponentsample
I CWNVU0440I: [ivu] The following file is different: lib/different.jar
I CWNVU0410I: [ivu] fc19318dd13128ce14344d066510a982269c241b is the checksum in the bill of materials.
I CWNVU0420I: [ivu] 517d5a7240861ec297fa07542a7bf7470bb604fe is the checksum on the file system.
I CWNVU0390I: [ivu] Component issues found : 1
I CWNVU0480I: [ivu] Done analyzing: regularcomponentsample
```

Ignore issues that are obvious informational (I) messages

Some messages indicate deviations from the normally expected result, but are not indicators of a serious issue:

```
I CWNVU0360I: [ivu] The following bill of materials issue is found for component nullvaluesample:
  Hash must not be null or an empty string.
```

Overlapped files are either a potential product issue or potential tampering with the IBM provided bill of materials

```
I CWNVU0470I: [ivu] Starting to analyze: overlapbinarycomponentsample
W CWNVU0422W: [ivu] The following file is overlapped: lib/binaryTest.jar
W CWNVU0425W: [ivu] The overlap is caused by: _binarycomponentsample
I CWNVU0390I: [ivu] Component issues found : 1
I CWNVU0480I: [ivu] Done analyzing: overlapbinarycomponentsample
```

Contact IBM support for the following issue

If you see any messages with the following format, contact IBM support:

```
W CWNVU0280W: [ivu] Component mismatch: expected ... but found ...
```

IBM Support has documents that can save you time gathering information needed to resolve this problem. Before opening a PMR, see this IBM Support page.

Next

After verifying your installation, you can create profiles or deploy an application on an existing application server profile.

After installing, updating, and verifying, the next step is to use the product. If you created a stand-alone application server or a cell, start the application server, or the deployment manager, node agent, and federated application server, to use the administrative console to deploy an existing application.

Chapter 3. Configuring ports

When you configure WebSphere Application Server resources or assign port numbers to other applications, you must avoid conflicts with other assigned ports. In addition, you must explicitly enable access to particular port numbers when you configure a firewall.

1. Review the port number settings, especially when you are planning to coexist.
2. **Optional:** Change the port number settings.

   During installation, you can use the Installation wizard.

   You can set port numbers when configuring the product after installation.

   - During profile creation using the Profile Management tool, you can accept the port settings recommended by the tool or you can specify your port settings.
   - During profile creation using the **manageprofiles** command, you can accept the default port values or you can specify your port settings. If you want to specify ports, you can do so in any of the following ways:
     - Specify the use of a port file that contains the port values.
     - Specify the use of a starting port value.
     - Specify the use of the default port values.

   You can perform one of the following actions to change port settings after installation:

   - Use the **updatePorts** tool to change port settings.
   - Edit the *profile_root*/config/cells/*cell_name*/nodes/*node_name*/serverindex.xml file to change the port settings, or use scripting to change the values.

Port number settings in WebSphere Application Server versions

You should be able to identify the default port numbers used in the various versions of WebSphere Application Server so that you can avoid port conflicts if you plan for an earlier version to coexist or interoperate with Version 6.1.

When you configure WebSphere Application Server resources or assign port numbers to other applications, you must avoid conflicts with other assigned ports. In addition, when you configure a firewall, you must explicitly enable access to particular port numbers.

Version 6.1 port numbers

Table 1. Port definitions for WebSphere Application Server Version 6.1

Port Name	Default Value	Files
HTTP_Transport Port (WC_defaulthost)	9080	serverindex.xml and virtualhosts.xml
Administrative Console Port (WC_adminhost)	9060	
HTTPS Transport Port (WC_defaulthost_secure)	9443	
Administrative Console Secure Port (WC_adminhost_secure)	9043	

Table 1. Port definitions for WebSphere Application Server Version 6.1 (continued)

Port Name	Default Value	Files
Bootstrap Port (BOOTSTRAP_ADDRESS)	2809	serverindex.xml
SOAP Connector Port (SOAP_CONNECTOR_ADDRESS)	8880	
SAS_SSL_SERVERAUTH_LISTENER_ADDRESS	9401	
CSIV2 Server Authentication Port (CSIV2_SSL_SERVERAUTH_LISTENER_ADDRESS)	9403	
CSIV2 Multi-authentication Listener Port (CSIV2_SSL_MUTUALAUTH_LISTENER_ADDRESS)	9402	
ORB Listener Port (ORB_LISTENER_ADDRESS)	9100	
High Availability Manager Communication Port (DCS_UNICAST_ADDRESS)	9353	
Service Integration Port (SIB_ENDPOINT_ADDRESS)	7276	
Service Integration Port Secure (SIB_ENDPOINT_SECURE_ADDRESS)	7286	
MQ Transport (SIB_MQ_ENDPOINT_ADDRESS)	5558	
MQ Transport secure (SIB_MQ_ENDPOINT_SECURE_ADDRESS)	5578	
SIP Container Port (SIP_DEFAULTHOST)	5060	
SIP Container Secure Port (SIP_DEFAULTHOST_SECURE)	5061	
Internal JMS Server (JMSSERVER_SECURITY_PORT)	5557	
DRS_CLIENT_ADDRESS **Deprecation:** This port is deprecated and is no longer used in the current version of WebSphere Application Server.	7873	
IBM HTTP Server Port	80	virtualhosts.xml, plugin-cfg.xml, and *web_server_root*/conf/ httpd.conf
IBM HTTPS Server Administration Port	8008	*web_server_root*/conf/ admin.conf

Version 6.0.x port numbers

Table 2. Port definitions for WebSphere Application Server Version 6.0.x

Port Name	Default Value	Files
HTTP_TRANSPORT	9080	serverindex.xml and virtualhosts.xml
HTTP Admin Console Port (HTTP_TRANSPORT_ADMIN)	9060	
HTTPS Transport Port (HTTPS_TRANSPORT)	9443	
HTTPS Admin Console Secure Port (HTTPS_TRANSPORT_ADMIN)	9043	
BOOTSTRAP_ADDRESS	2809	serverindex.xml
SOAP_CONNECTOR_ADDRESS	8880	
SAS_SSL_SERVERAUTH_LISTENER_ADDRESS	9401	
CSIV2_SSL_SERVERAUTH_LISTENER_ADDRESS	9403	
CSIV2_SSL_MUTUALAUTH_LISTENER_ADDRESS	9402	
ORB_LISTENER_ADDRESS	9100	
DCS_UNICAST_ADDRESS	9353	
SIB_ENDPOINT_ADDRESS	7276	
SIB_ENDPOINT_SECURE_ADDRESS	7286	
SIB_MQ_ENDPOINT_ADDRESS	5558	
SIB_MQ_ENDPOINT_SECURE_ADDRESS	5578	
Internal JMS Server (JMSSERVER_SECURITY_PORT)	5557	
DRS_CLIENT_ADDRESS **Deprecation:** This port is deprecated and is no longer used in the current version of WebSphere Application Server.	7873	
IBM HTTP Server Port	80	virtualhosts.xml, plugin-cfg.xml, and *web_server_root*/conf/ httpd.conf
IBM HTTPS Server Administration Port	8008	*web_server_root*/conf/ admin.conf
NODE_MULTICAST_IPV6_DISCOVERY_ADDRESS	5001	serverindex.xml

Version 5.1 port numbers

Table 3. Port definitions for WebSphere Application Server Version 5.1

Port Name	Default Value	Files
HTTP_TRANSPORT	9080	server.xml and virtualhosts.xml
HTTPS_TRANSPORT	9443	
HTTP_TRANSPORT_ADMIN	9090	
HTTPS_TRANSPORT_ADMIN	9043	
JMSSERVER_SECURITY_PORT	5557	server.xml
JMSSERVER_QUEUED_ADDRESS	5558	serverindex.xml
JMSSERVER_DIRECT_ADDRESS	5559	
BOOTSTRAP_ADDRESS	2809	
SOAP_CONNECTOR_ADDRESS	8880	
DRS_CLIENT_ADDRESS	7873	
SAS_SSL_SERVERAUTH_LISTENER_ADDRESS	0	
CSIV2_SSL_SERVERAUTH_LISTENER_ADDRESS	0	
CSIV2_SSL_MUTUALAUTH_LISTENER_ADDRESS	0	
ORB_LISTENER_ADDRESS	0	
IBM HTTP Server Port	80	virtualhosts.xml, plugin-cfg.xml, and *web_server_root*/conf/ httpd.conf
IBM HTTPS Server Administration Port	8008	*web_server_root*/conf/ admin.conf

Chapter 4. Communicating with Web servers

The WebSphere Application Server 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 WebSphere Application Server. The Web server plug-in uses the XML configuration file to determine whether a request is from the Web server or the Application Server.

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

  If you want to use an IBM HTTP Server, see the *Installing your application serving environment* PDF. Otherwise, see the installation information provided with your Web server.
- Ensure that your Web server is configured to perform operations required by Web applications, such as GET and POST. Typically, this involves setting a directive in the Web server configuration file (such as the httpd.conf file for an IBM HTTP Server). Refer to the Web server documentation for instructions. If an operation is not enabled when a servlet or JSP file requiring the operation is accessed, an error message displays, such as this one from the IBM HTTP Server:

  ```
  HTTP method POST is not supported by this URL.
  ```
- Make sure the appropriate plug-in file has been installed on your Web server and the configureWeb_server_name script has been run to create and configure the Web server definition for this Web server.

  If you are using a distributed platform Web server, use the Plug-in Installation wizard to install the appropriate plug-in file to your Web server. Then run the configureWeb_server_name script created by the wizard to create and configure the Web server definition in the WebSphere configuration repository.

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

1. A Web server definition is created.

   You can also use either the administrative console or use the ConfigureWebServerDefintion.jacl script to create a Web server definition. If you use the administrative console:

   a. Select the node that was created in the preceding step, and in the Server name field, enter the local name of the Web server for which you are creating a Web server definition.

   b. Use the wizard to complete the Web server definition.
2. An application or modules are mapped to a Web server. If an application that you want to use with this Web server is already installed, the application is automatically mapped to the Web server. If the application is not installed, select this Web server during the Map modules to servers step of the application installation process.
3. The master repository is updated and saved.

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

Generating or regenerating the configuration file might take a while to complete. After it finishes, all objects in the administrative cell use their newest settings, which the Web server can access. If the Application Server is on the same physical machine as the Web server, the regeneration usually takes about 30 to 60 seconds to complete. The regeneration takes longer if they are not both on the same machine.

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

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

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

a. Select **Servers > Web Servers >** *webserver* **> plug-in properties**.

b. Select **Automatically generate plug-in configuration file** or click on one or more of the following topics to manually configure the plugin-cfg.xml file:
   - Caching
   - Request and response
   - Request routing
   - Service

   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. You might need 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. **Optional:** Edit the plug-in configuration file. You should not have to edit the configuration file. If you do edit this file remember that:
   - The file is in ASCII format (ISO-98859-1).
   - Any manual changes you make to the file are overwritten the next time the file is regenerated.
3. If you want to use Secure-socket layer (SSL) with this configuration, use the plug-in's installation wizard to install the appropriate GSKIT installation image file on your workstation.
4. If you want to enable the Application Server to use the private headers that the Web server plug-in sends, make sure the transport you are using is configured for SSL and is trusted. If a trust file definition is not included, the private headers will be ignored, and the application server might not locate the requested application.

   If you are using an HTTP transport, make sure the transport is configured for SSL and the TrustedProxy custom property for the transport is set to true.

   After you enable the use of private headers, this transport trusts all inbound private headers it receives. Therefore, you must ensure that all inbound paths to this transport are trusted.
5. Tune your Web server. See the *Tuning guide* PDF for more information.
6. Propagate the plug-in configuration. The plug-in configuration file (plugin-cfg.xml) is automatically propagated to the Web server if the Web server plug-in configuration service is enabled, and one of the following is true:
   - The Web server is a local Web server. (It are located on the same machine as an application server.)
   - The Web server is a remote IBM HTTP Server Version 6.0 that has a running IBM HTTP Server Administrative server.

   If neither of these conditions is true, the plugin-cfg.xml file must be manually copied to the remote Web server's installation location.

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

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

```
chmod 644 plugin-cfg.xml
```

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

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** > **Application Servers** > *server_name* > **Administration Services** >**Web server plug-in configuration service** and then unselect the Enable automated Web server configuration processing option.

Tip: 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

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

Before using this topic to install the IBM HTTP Server, see the Information Center for IBM HTTP Server.

To use a Web server other than IBM HTTP Server Version 6.x, 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.

The Plug-ins installation wizard configures supported Web servers. You can also manually configure supported Web servers for WebSphere Application Server, Version 6, as described in “Editing Web server configuration files” on page 102.

See the information center for IBM HTTP Server Version 6 for more information.

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.

This topic describes installing IBM HTTP Server.

1. Prepare your operating platform for installing IBM HTTP Server as you would for installing any of the installable components on the product disc.
2. Insert the product disc and mount the disc if necessary.
3. Start the installation with the launchpad.sh command on Linux and UNIX platforms or the launchpad.bat on Windows platforms. You can also start the installation using the /IHS/install command, where IHS is the installable component directory on the product disc:

   - Linux /IHS/install
   - Windows \IHS\install

   If you have a problem launching the launchpad, use the troubleshooting information in *Installing your application serving environment* PDF to correct the problem.

   When using the launchpad, launch the Installation wizard for IBM HTTP Server:

   After launching the Installation wizard from the launchpad or from the command line, the ISMP wizard initializes and then displays the Welcome panel.

   Separate installation procedures for the WebSphere Application Server product, the IBM HTTP Server product, and the Web server plug-ins let you install only what you need on a particular machine.
4. Click **Next** to display the License agreement panel.
5. Accept the license agreement and click **Next** to display the installation root directory panel.
6. Specify the root directory information and click **Next** to display the feature type selection panel.

   The panel lets you bypass features selection by accepting typical features. Selecting Custom lets you select features in the Features selection panel.
7. Click **Custom** to select features and click **Next** to display the Features selection panel.
8. Windows Select features to install and click **Next** to display the Windows service authorization panel, when installing IBM HTTP Server on a Windows platform.
9. Windows If you are planning to administer IBM HTTP Server using the WebSphere Application Server V6.x administrative console, select **Run IBM HTTP Administration as a Windows Service with Log on as Local System account**. User name and password is not required for this selection. Click **Next**. If you will not administer IBM HTTP Server using the WebSphere Application Server V6.x administrative console, select **Run IBM HTTP Administration as a Windows Service with Log on as a specified user account** and specify your user ID and password information. Click **Next**.
10. Review the confirmation panel to verify your selections. Click **Back** to change any of your specifications. Click **Next** to begin installing IBM HTTP Server.
11. Review the confirmation panel to verify your selections. Click **Back** to change any of your specifications. Click **Next** to begin installing IBM HTTP Server.

    After displaying installation status, the wizard displays the Completion status panel that indicates a successful installation.
12. Click **Next** to display the Web server plug-ins prompt panel.
13. Click **Next** to launch the Plug-ins installation wizard. See “Installing Web server plug-ins” on page 53 to continue the installation.

    If the plugin directory does not exist at the same level as the IHS directory, the prompt panel for selecting the plug-ins installer does not display and the installation is finished. In that case, launch the Plug-ins installation wizard using the launchpad.

You can install the IBM HTTP Server product.

Refer to the Information Center for IBM HTTP Server at http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.ihs.doc/info/welcome_ihs.html for a description of configuring the Web server for SSL, the Fast Response Cache Accelerator, or Apache directives.

See the *Installing your application serving environment* PDF for more information to continue.

Installing Web server plug-ins

It is possible to configure your Web server plug-in to route requests to WebSphere Application Server V4.x, V5.x, and V6.x releases. This topic describes configuring Web server plug-ins to route requests to WebSphere Application Server V6.x releases.

Go to http://www-1.ibm.com/support/docview.wss?rs=180&context=SSEQTP&uid=swg21160581 for information about how to verify what V4.0, V5.0, V5.1, V6.0, and V6.1 plug-in versions are installed on local or remote Web servers, and how to determine if the installation complies with supported configurations.

You must install a supported Web server before you can install a plug-in for the Web server. If the Web server is not already installed, you cannot install the plug-in for it. If the WebSphere Application Server product is not installed, you can install the plug-in. To create a Web server configuration for unmanaged nodes, WebSphere Application Server must be installed on your system.

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.

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

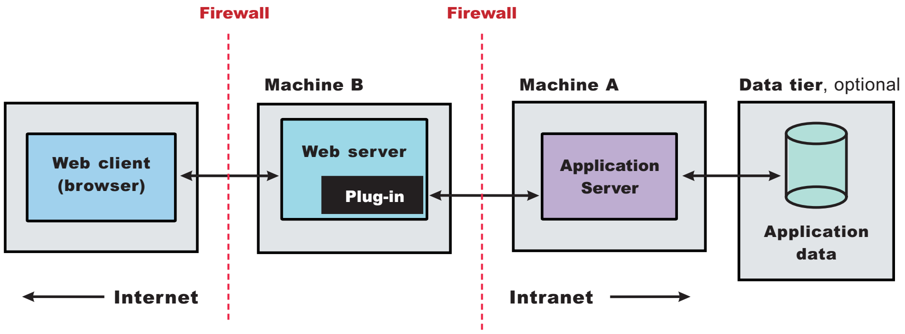
This topic describes installing a Web server plug-in for WebSphere Application Server. WebSphere Application Server products supply a unique binary plug-in module for each supported Web server. The plug-in configuration file, which the WebSphere Application Server products create and maintain, interacts with the binary module to provide information about the application server configuration to the Web server. The Web server uses the information to determine how to communicate with the application server, but to locate specific applications on the application server.

The Plug-ins installation wizard installs required files and configures the Web server and the application server to allow communication between the servers.

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

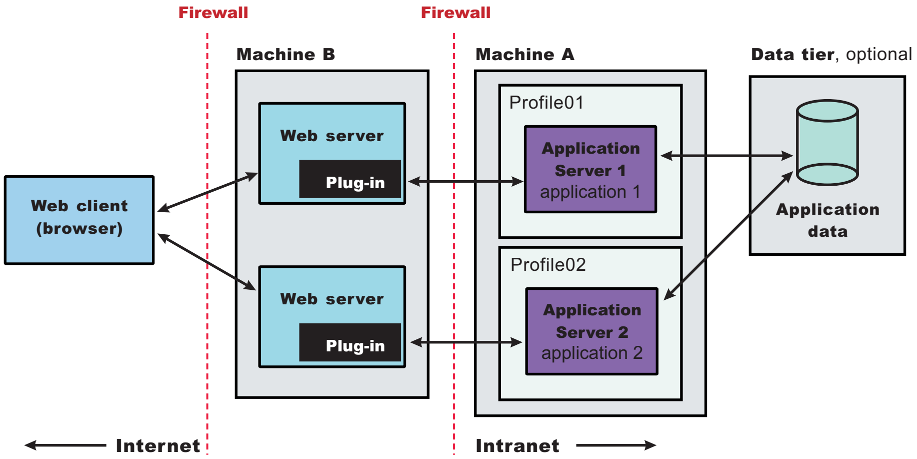
When multiple profiles exist, the plug-ins installer configures only the default profile. See "Plug-ins configuration" on page 58 for a description of the flow of logic that determines how the installer selects the profile to configure.

- **Scenario 1: Remote** The application server and the Web server are on separate machines or logical partitions.



See “Configuring a Web server and an application server on separate machines (remote)” on page 67 for the procedure that explains how to create this Web server topology.

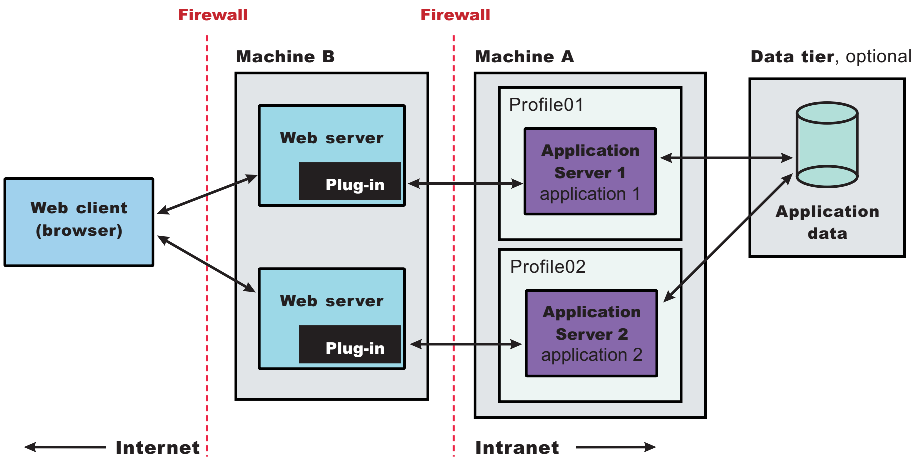
- **Scenario 2: Remote** Multiple stand-alone application servers are on one machine, and each application server has a dedicated Web server on a separate machine or logical partition.



See “Configuring multiple Web servers and remote stand-alone application servers” on page 74 for the procedure that explains how to create this Web server topology.

- **Scenario 3: Local Application Server profile** The application server and the Web server are on a single machine or logical partition.

  See “Configuring a Web server and an application server profile on the same machine” on page 82 for the procedure that explains how to create this Web server topology for an application server profile.
- **Scenario 6: Non-default profile** Creating a Web server definition for a profile that is not the default profile.



You can install a Web server and the Web server plug-ins for various stand-alone application server topologies by following the procedures described in this topic.

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

See “Web server configuration” on page 63 for more information about the files involved in configuring a Web server.

See “Editing Web server configuration files” on page 102 for information about how the Plug-ins installation wizard configures supported Web servers.

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.

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.

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:

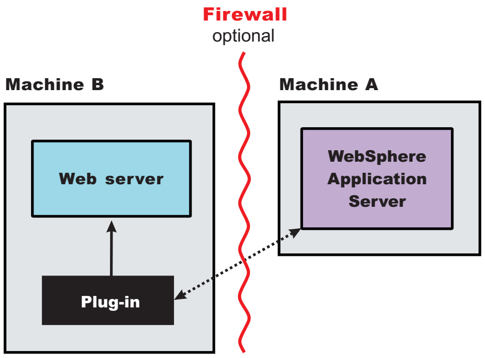
Detects the default profile on a local application server machine and creates the Web server definition for it directly. Install the Web server and its plug-in on the same machine with the application server. This configuration is for development and test environments.

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

- **Set up a remote Web server installation.**

  The remote Web server configuration is recommended for production environments.

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



Remote installation scenario

Table 4. Installation and configuration

Step	Machine	Task
1	A	Install your WebSphere Application Server product. See Installing the product and additional software.
2	B	Install IBM HTTP Server or another supported Web server..
3	B	Install the binary plug-in module using the Plug-ins installation wizard. See “Configuring a Web server and an application server on separate machines (remote)” on page 67. The script for creating and configuring the Web server is created under the *plugins_root*/ bin directory.
4	B	Copy the configure*web_server_name* script to Machine A. If one machine is running under Linux or UNIX and the other machine is running under Windows, copy the script from the *plugins_root*/bin/crossPlatformScripts directory. See “Configuring a Web server and an application server on separate machines (remote)” on page 67 for information about cross-platform scripts and file encoding differences.
5	A	Paste the configure*web_server_name* script from Machine B to the *app_server_root*/bin directory on Machine A.
6	A	Start the application server, then run the script from a command line.
7	A	Verify that the application server is running. Open the administrative console and save the changed configuration.
8	B	Linux Source the *plugins_root*/setupPluginCfg.sh script for a Domino Web Server before starting a Domino Web server. Start the Web server.

Table 4. Installation and configuration (continued)

Step	Machine	Task
9	B	Run the snoop servlet. To verify with your own application, regenerate and propagate the plugin-cfg.xml file after installing the application.

Regeneration of the plugin-cfg.xml file

During the installation of the plug-ins, the temporary plugin-cfg.xml file is installed on Machine B in the *plugins_root*/ config/ web_server_name directory.

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

To use the actual plugin-cfg.xml file from the application server, propagate the plugin-cfg.xml file as described in the next section.

Propagation of the plugin-cfg.xml file

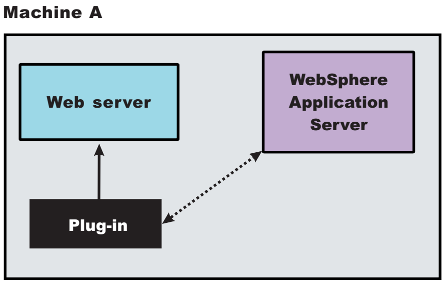
The Web server plug-in configuration service propagates the plugin-cfg.xml file automatically for IBM HTTP Server 6.0 only.

For all other Web servers, propagate the plug-in configuration file manually. Copy the plugin-cfg.xml file from the *profile_root*/ config/cells/*cell_name*/nodes/*web_server_name*_node/servers/*web_server_name* directory on Machine A. Paste the file into the *plugins_root*/config/*web_server_name* directory on Machine B.

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

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

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



Local installation scenario

Table 5. Installation and configuration

Step	Machine	Task
1	A	Install your WebSphere Application Server product. See Installing the product and additional software.
2	A	Install IBM HTTP Server or another supported Web server.
3	A	Install the binary plug-in module using the Plug-ins installation wizard. See “Configuring a Web server and an application server profile on the same machine” on page 82. The Web server definition is automatically created and configured during the installation of the plug-ins.
4	A	Verify that the application server is running. Open the administrative console and save the changed configuration.
5	B	Linux Run the *plugins_root*/setupPluginCfg.sh script for a Domino Web Server before starting a Domino Web server.Otherwise, start the Web server.

Table 5. Installation and configuration (continued)

Step	Machine	Task
6	B	Run the snoop servlet. To verify with your own application, regenerate and propagate the plugin-cfg.xml file after installing the application.

Regeneration of the plugin-cfg.xml file

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

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

Propagation of the plugin-cfg.xml file

The local file does not require propagation.

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

See "Web server configuration" on page 63 for more information about the files involved in configuring a Web server.

See "Plug-ins configuration" for information about the logic behind the processing scenarios for the Plug-ins installation wizard.

See "Editing Web server configuration files" on page 102 for information about how the Plug-ins installation wizard configures supported Web servers.

See "Installing Web server plug-ins" on page 53 for information about other installation scenarios for installing Web server plug-ins.

Plug-ins configuration

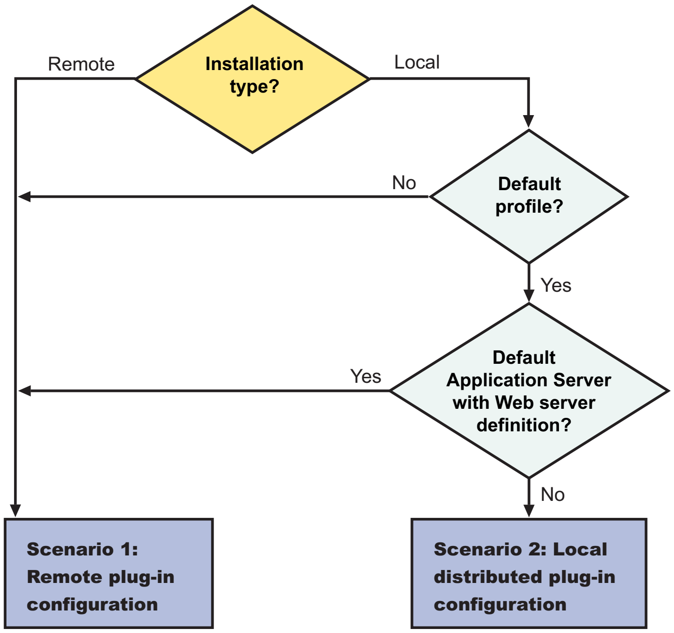
The Plug-ins installation wizard installs a binary plug-in module and a plug-in configuration file for the Web server. The wizard then configures the supported Web server for the Application Server and creates a Web server definition in the configuration of the application server. This overview shows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


Installation type?
Remote
Local
Default profile?
No
Yes
Default Application Server with Web server definition?
Yes
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:

   - Linux ./*IHS_root*/bin/apachectl start
   - Windows *IHS_root*\bin\apache

2. Start the application server on the remote machine.

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

   - Linux `./profile_root/bin/startServer.sh server1`
   - Windows `profile_root\bin\startServer server1`

3. Point your browser to http://localhost:9080/snoop to test the internal HTTP transport provided by the application server. Point your browser to http://*Host_name_of_Web_server_machine*/snoop to test the Web server plug-in.
4. Verify that both Web addresses display the Snoop Servlet - Request/Client Information page.

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

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

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

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

   b. Open a command window and run the script:

      - Linux ./configure*web_server_name*.sh
      - Windows configure*web_server_name*.bat

   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.

   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:

   - Linux ./*IHS_root*/bin/apachectl start
   - Windows *IHS_root*\bin\apache

2. Start the application server.

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

   - Linux `./profile_root/bin/startServer.sh server1`
   - Windows `profile_root\bin\startServer server1`

   Open the administrative console and save the changed configuration.

3. Point your browser to http://localhost:9080/snoop to test the internal HTTP transport provided by the Application Server. Point your browser to http://*Host_name_of_Web_server_machine*/snoop to test the Web server plug-in.
4. Verify that both Web addresses display the Snoop Servlet - Request/Client Information page.

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 6. 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.

After installing a supported Web server, you must install a binary plug-in module for the Web server. The plug-in module lets the Web server communicate with the application server. The Plug-ins installation wizard installs the Web server plug-in. The wizard configures the Web server. The wizard also creates a Web server definition in the configuration of the application server. The Plug-ins installation wizard 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 Plug-ins installation Wizard 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 **default (temporary) plug-in configuration file, plugin-cfg.xml**, on the Web server machine.
- The **configure*web_server_name* script** that you copy from the Web server machine to the application server machine.

See the following descriptions of each file.

Web server configuration file

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

The wizard must reconfigure the configuration file for a supported 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 133 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.

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 settings” on page 126 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.

The stand-alone application server regenerates the plugin-cfg.xml file in the *profile_root*/config/cells/*cell_name*/nodes/*web_server_name*_node/servers/*web_server_name* directory. Regeneration occurs whenever a change occurs in the application server configuration that affects deployed applications.

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

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

See “Web server plug-in configuration service property settings” on page 126 for more information.

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

The Plug-ins installation wizard creates the temporary plugin-cfg.xml file in the *plugins_root*/config/*web_server_name* directory. The wizard creates the file for every remote installation scenario. The wizard creates the file at the same time that it installs the binary plug-in module for the Web server.

The default file is a placeholder that you must replace with the plugin-cfg.xml file from the Web server definition on the application server. The default file is a replica of the file that the application server creates for a default stand-alone application server that has the samples installed.

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

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

The configure*web_server_name* script for the Web server definition

The Plug-ins installation wizard creates the configure*web_server_name* script on the Web server machine in the *plugins_root*/bin directory. If one machine in a remote scenario is running under Linux or UNIX and the other machine is running under Windows, use the script created in the *plugins_root*/bin/crossPlatformScripts directory. The script is created for remote installation scenarios only.

Copy the script from the Web server machine to the *app_server_root*/bin directory on a remote application server machine. You do not have to copy the script on a local installation. Run the script to create a Web server definition in the configuration of the application server.

When using the IBM HTTP Server, configure the IBM HTTP Administration Server also. The IBM HTTP Administration Server works with the administrative console to manage Web server definitions. Also, use the administrative console to update your Web server definition with remote Web server management options. Click **Servers > Web servers > *web_server_name*** to see configuration options. For example, click **Remote Web server management** to change such properties as:

- Host name
- Administrative port
- User ID
- Password

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

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

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

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

Reset the variable in either of two ways:

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

  Windows

  1. Open a command prompt window.
  2. Change directories to the *app_server_root*\bin directory.
  3. Issue the setupCmdLine.bat command.
  4. Use the same command prompt window to start the update installer, as described in the appropriate procedure.

  Linux

  1. Open a command shell window.

2. Change directories to the *app_server_root*/bin directory.
3. Issue the . ./setupCmdLine.sh command. Notice the space between the periods. The special format for this command sources the command to make the setting active for all processes started from the command shell.
4. Use the same command shell window to start the update installer, as described in the appropriate procedure.

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
```

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.

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

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

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

- **Virtual host name**

  Default value: default_host

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

  To access applications on the new virtual host, propagate the real plugin-cfg.xml file. Propagation is copying the updated file from the application server machine to the Web server machine.
- **HTTP transport port**

Default value: 9080

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

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

- **Web server listening port**

  Default value: 80

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

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

- **HTTPS transport port**

  Default value: 9443

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

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

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

  All of the default servlets and applications are included in the default file, including the WSsamples application and the SamplesGallery application.

  The default file lists all of the default applications and samples. The list can be inaccurate. If you performed a custom installation and did not install the samples, for example, the list is inaccurate.

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

Configuring a Web server and an application server on separate machines (remote)

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

When multiple profiles exist, the plug-ins installer configures only the default profile. See “Plug-ins configuration” on page 58 for a description of the flow of logic that determines how the installer selects the profile to configure.

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

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

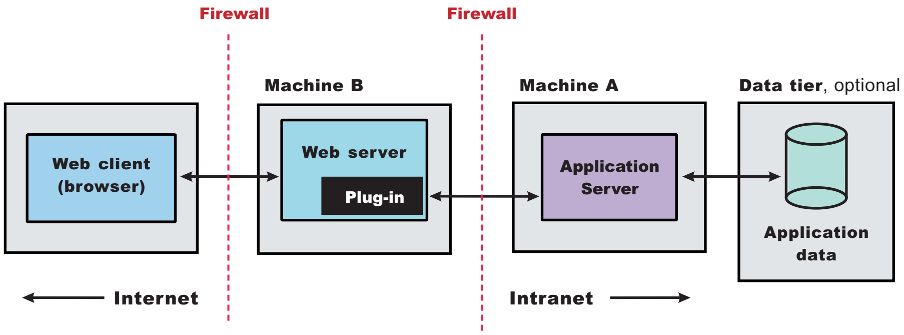
Suppose that you create a new profile. Suppose also that you want to use a Web server. You must install a new Web server for the new profile and use the Plug-ins installation wizard to install the binary plug-in module and to configure both the Web server and the application server.

If the Web server is not already installed, you can still install the plug-ins for future use. If the WebSphere Application Server product is not installed, you can still install the plug-ins. However, it is recommended that you install the Web server and the WebSphere Application Server product before installing the plug-ins for the supported Web server.

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.

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

This topic describes how to create the following topology:



This topic describes the installation of a Web server on one machine and the application server on a separate machine. In this situation, the Plug-ins installation wizard on one machine cannot create the Web server definition in the application server configuration on the other machine.

In such a case, the Plug-ins installation wizard creates a script on the Web server machine that you can copy to the application server machine. Run the script on the application server machine to create the Web server configuration definition within the application server configuration.

Perform the following procedure to install the plug-in and configure both the Web server and the application server.

1. Log on to the operating system.

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

   ```
   umask
   ```

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

   ```
   umask 022
   ```

   Windows When installing on a Windows system, you can create a Windows service to autostart the application server. To do so, your installer user account must have the following advanced user rights:

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

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

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

2. Install WebSphere Application Server - Express on Machine A.

   See Installing the product and additional software.

3. Install the IBM HTTP Server or another supported Web server on Machine B.

4. Launch the Plug-ins installation wizard on the machine with the Web server.

   Select the Plug-ins installation wizard from the launchpad or change directories to the plugin directory on the product disk or in the downloaded installation image and issue the install command.

5. Clear the check box for the roadmap or select the check box to view the roadmap, then click **Next**.

   If you are unsure of which installation scenario to follow, display the roadmap instead. Print and keep the roadmap as a handy overview of the installation steps.

   Press **Ctrl-P** to print the roadmap if the Web browser navigation controls and the menu bar are not present on the browser window that displays the Plug-ins roadmap. Press **Ctrl-W** to close the browser window if the navigation controls and the menu bar do not display. Or close the browser window with the window control in the title bar.

6. Read the license agreement and accept the agreement it if you agree to its terms. Click **Next** when you are finished.

7. If your system does not pass the prerequisites check, stop the installation, correct any problems, and restart the installation. If your system passes the prerequisites check, click **Next**.

   Look for the appropriate log file for information about missing prerequisites:

   - If you stop the installation, see the temporaryPluginInstallLog.txt file in the temporary directory of the user who installed the plug-ins. For example, the /tmp/temporaryPluginInstallLog.txt file might exist if the root user installed the plug-ins on a Linux or UNIX system.
   - If you continue the installation in spite of warnings about missing prerequisites, see the *plugins_root*/logs/install/log.txt file after the installation is complete.

   See Troubleshooting installation for more information about log files.

8. Select the type of Web server that you are configuring and click **Next**.

   The Plug-ins installation wizard panel prompts you to identify the *Web servers* to configure. Actually you can select only one Web server each time you run the Plug-ins installation wizard.

   Stop any Web server while you are configuring it. A step later in the procedure directs you to start the Web server as you begin the snoop servlet test.

   If you select the Web server identification option labeled **None**, the Web server installs the binary plug-ins but does not configure the Web server.

9. Select **Web server machine (remote)** and click **Next**.

10. Accept the default location for the installation root directory for the plug-ins. Click **Next**.

    You can type another new directory or click **Browse** to select an empty directory. The fully qualified path identifies the plug-ins installation root directory.

    The default location is shown in “Directory conventions,” on page 305.

    A possibility exists that the Web server might run on a platform that WebSphere Application Server does not support.

11. Click **Browse** to select the configuration file for your Web server, verify that the Web server port is correct, and then click **Next** when you are finished.

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

    The following list shows configuration files for supported Web servers:

    Apache HTTP Server
        apache_root/config/httpd.conf

Domino Web Server

names.nsf and Notes.jar

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

The Plug-ins installation wizard verifies that the files exist but the wizard does not validate either file.

IBM HTTP Server

IHS_root/conf/httpd.conf

Microsoft Internet Information Services (IIS)

The Plug-ins installation wizard can determine the correct files to edit.

Sun ONE Web Server 6.0 or Sun Java System Web Server, Version 6.1

obj.conf and magnus.conf

The wizard displays a naming panel for the nickname of the Web server definition.

12. Specify a nickname for the Web server. Click **Next** when you are finished.

    The wizard uses the value to name configuration folders in the plug-ins installation root directory. The wizard also uses the name in the configuration script for the application server to name the Web server definition.

    If the application server profile already has a Web server definition, delete the Web server definition before continuing. Use the following commands to delete the Web server definition:

    ```
    $AdminTask deleteServer { -serverName webserver1 -nodeName webserver1_node }
    $AdminTask removeUnmanagedNode { -nodeName webserver1_node }
    $AdminConfig save
    ```

13. Accept the default location for the plugin-cfg.xml file that the wizard creates on the Web server machine, then click **Next**.

    You can type a change to the value or click **Browse** to select a file in another location. If you do not accept the default location, the plugin-cfg.xml file must exist.

14. Identify the host name or IP address of Machine A, which is the application server machine, then click **Next**.

15. Examine the summary panel. Click **Next** when you are finished.

    The panel notifies you that you have manual steps to perform to complete the installation and configuration. The type of Web server, the nickname of the Web server, and the location of the plugin-cfg.xml file displays on the panel.

    The Plug-ins installation wizard creates the configure*web_server_name* script in the *plugins_root*/bin/ directory on Machine B.

    The Plug-ins installation wizard also creates the plugin-cfg.xml file in the *plugins_root*/config/ *web_server_name* directory.

    The Web server reads the plugin-cfg.xml file to determine the applications that the application server on Machine A can serve to the Web server on Machine B. Whenever the configuration changes, the application server regenerates the file. When regeneration occurs, propagate, or copy the actual plugin-cfg.xml file from the application server machine to the Web server machine. You can automatically propagate the file to the IBM HTTP Server product.

16. Click **Next** on the pre-installation summary panel to begin the installation or click **Back** to change any characteristics of the installation.

    The panel specifies the plug-ins installation root directory, the Web server plug-ins feature, and the disk size of the code that installs when you click **Next**.

17. After the wizard installs the code and creates the uninstaller program, examine the post-installation summary panel. Click **Next** when you are finished to display the Plug-ins installation roadmap.

    The Plug-ins installation wizard installs the binary plug-in module. On a Linux system, for example, the installation creates the *plugins_root* directory. The *plugins_root*/config/*web_server_name* directory contains the plugin-cfg.xml file.

The wizard displays the name and location of the configuration script and the plugin-cfg.xml file. The wizard also displays the type of Web server that is configured and the nickname of the Web server.

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

18. Close the road map and click **Finish** to exit the wizard.

    Log files from the installation are in the *plugins_root*/logs/install directory.

19. Copy the configure*web_server_name*.sh script on Linux and UNIX systems, the configure*web_server_name*.bat script on Windows systems, or the configure*web_server_name* script on i5/OS from Machine B to the *app_server_root* /bin directory on Machine A.

    On a Linux or UNIX system, the file is configure*web_server_name*.sh. On a Windows system, the file is configure*web_server_name*.bat. For example, on a Linux system with an IBM HTTP Server named webserver1 in the default location, copy *plugins_root*/bin/configurewebserver1.sh from Machine B to the *app_server_root*/bin directory on Machine A.

    If one platform is a Linux or UNIX system and the other is a Windows platform, copy the script from the crossPlatformScripts directory. For example:

    - Linux *plugins_root*/`bin/configure`*`Web_server_name`*`.sh`
    - Windows *plugins_root*/`bin/crossPlatformScripts/configure`*`Web_server_name`*`.bat`

20. Compensate for file encoding differences to prevent script failure.

    The content of the configure*web_server_name*.bat script or the configure*web_server_name*.sh script can be corrupt if the default file encoding of the two machines differs. This scenario is possible when one machine is set up for a double-byte character set (DBCS) locale and the other machine is not.

    Determine the file encoding and use one of the following procedures to circumvent the failure. To determine the default file encoding, run the appropriate command.

    - Windows Run the following command on a Windows machine:

      ```
      CHCP
      ```

    - Linux Run the following command on a Linux or UNIX machine:

      ```
      locale
      ```

    Use the result of the command on each machine as the value of the *web_server_machine_encoding* variable and the *application_server_machine_encoding* variable in one of the following procedures.

    Procedures for compensating for encoding differences

    Suppose that the Web server is running on a Linux machine and Network Deployment is running on a Windows machine.

    Web server running on a Linux or UNIX system

    Run the following command on the Linux or UNIX system to encode the script file that configures the Web server definition, before you FTP the file to the Windows machine in binary mode:

    ```
    iconv -f web_server_machine_encoding \
       -t application_server_machine_encoding \
       configureweb_server_name.bat
    ```

    Omit the Linux and UNIX continuation characters (\) if you enter the command on one line.

    Important: The name of the Web server (nick name) is used in the name of the script file. The name cannot contain characters from a double-byte character set (DBCS) if you intend to set up IBM HTTP Server for automatic propagation.

    Suppose that the Web server is running on a Windows machine and Network Deployment is running on a Linux or UNIX machine.

    Web server running on a Windows machine

    Run the following command on the Linux or UNIX system to encode the script file that configures the Web server definition, after you FTP the file in binary mode:

```
iconv -f web_server_machine_encoding \
   -t application_server_machine_encoding \
   configureweb_server_name.sh
```

Omit the Linux and UNIX continuation characters (\) if you enter the command on one line.

If the conversion mapping is not supported by the iconv command on your system, copy the contents of the Web server configuration script to a clip board and paste it onto the machine where the application server is running.

21. Start the application server on Machine A.

    Use the startServer command, for example:

    - Linux `profile_root/bin/startServer.sh server1`
    - Windows `profile_root\bin\startServer server1`

22. Open a command window and change to the profile directory where the Web server should be assigned. Run the script that you copied to Machine A. You will need the following parameters:

    ```
    Profile Name
    (Optional) Admin user ID
    (Optional) Admin user password
    ```

    For example: *configurewebserver1.sh Dmgr01 myUserID myPassword*

    The webserver will be configured via wsadmin.

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

    Linux To configure Domino, you must set the WAS_PLUGIN_CONFIG_FILE environment variable. On Linux and UNIX-based platforms, sourcing a script to the parent shell allows child processes to inherit the exported variables. On Windows systems, run the script as you would run any other command. Sourcing is automatic on Windows systems.

    a. Open a command window.

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

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

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

    The script is also in the *lotus_root*/notesdata directory on Linux and UNIX systems.

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

24. Start the Snoop servlet to verify the ability of the Web server to retrieve an application from the Application Server.

    Test your environment by starting your Application Server, your Web server, and using the snoop servlet with an IP address.

    a. Start the Application Server.

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

       - Linux ./startServer.sh server1
       - Windows startServer server1

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

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

       To start the IBM HTTP Server from the command line:

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

       - Linux ./apachectl start
       - Windows apache

c. Point your browser to http://localhost:9080/snoop to test the internal HTTP transport provided by the Application Server. Point your browser to http://*Host_name_of_Web_server_machine*/snoop to test the Web server plug-in.

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

d. Verify that snoop is running.

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

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

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

   1) Create a user=adminUser, password=adminPassword in the *IHS_root* /conf/admin.passwd file. For example: c:\ws\ihs60\bin\htpasswd -cb c:\ws\ihs60\conf\admin.passwd adminUser adminPassword
   2) Use the administrative console of the Application Server to enter the User ID and password information that you created for the administrative user of IBM HTTP Server. Go to **Servers > Web server >** ***Web_server_definition*** **> Remote Web server administration**. Set the following values: admin Port=8008, User Id=adminUser, Password=adminPassword.
   3) Set the correct read/write permissions for the httpd.conf file and the plugin-cfg.xml file. See the *IHS_root* /logs/admin_error.log file for more information.

   Automatic propagation of the plug-in configuration file requires the IBM HTTP administrative server to be up and running. If you are managing an IBM HTTP Server using the WebSphere Application Server administrative console, the following error might display:

   ```
   "Could not connect to IHS Administration server error"
   ```

   Perform the following procedure to correct the error:

   1) Verify that the IBM HTTP Server administration server is running.
   2) Verify that the Web server host name and the port that is defined in the WebSphere Application Server administrative console matches the IBM HTTP Server administration host name and port.
   3) Verify that the fire wall is not preventing you from accessing the IBM HTTP Server administration server from the WebSphere Application Server administrative console.
   4) Verify that the user ID and password that is specified in the WebSphere Application Server administrative console under remote managed, is created in the admin.passwd file, using the htpasswd command.
   5) If you are trying to connect securely, verify that you export the IBM HTTP Server administration server keydb personal certificate into the WebSphere Application Server key database as a signer certificate. This key database is specified by the com.ibm.ssl.trustStore directive in the sas.client.props file in the profile where your administrative console is running. This consideration is primarily for self-signed certificates.
   6) If you still have problems, check the IBM HTTP Server admin_error.log file and the WebSphere Application Server logs (trace.log file) to determine the cause of the problem.

25. Regenerate the plugin-cfg.xml file on Machine A using the administrative console. Click **Servers > Web server**. Select the Web server, then click **Generate Plug-in**.

    During the installation of the plug-ins, the default plugin-cfg.xml file is installed on Machine B in the *plugins_root*/config/*web_server_name* directory. The Web server plug-in configuration service regenerates the plugin-cfg.xml file automatically. To use the current plugin-cfg.xml file from the application server, propagate the plugin-cfg.xml file as described in the next step.

This step shows you how to regenerate the plugin-cfg.xml file. WebSphere Application Server products are configured to automatically regenerate the file each time a significant event occurs. Such events include installing applications on the application server and the Web server, for example. Creating a new virtual host is another such event.

26. Propagate the plugin-cfg.xml file from the application server to the Web server using the administrative console. Click **Servers > Web server**. Select the Web server, then click **Propagate Plug-in**. Web servers other than IBM HTTP Server require manual propagation.

    The Web server plug-in configuration service propagates the plugin-cfg.xml file automatically for IBM HTTP Server 6.0 only. For all other Web servers, propagate the plug-in configuration file by manually copying the plugin-cfg.xml file from the *profile_root*/config/cells/*cell_name*/nodes/*node_name*/servers/*web_server_name* directory on Machine A to the *plugins_root*/config/*web_server_name* directory on Machine B.

This procedure results in the installation of the Web server plug-ins for WebSphere Application Server on a Web server machine. The Plug-ins installation wizard also configures the Web server to support an application server on a separate machine.

The installation of the binary plug-in modules results in the creation of the Plugins directory and several subdirectories. The following directories are among those created on a Linux system, for example:

- *plugins_root*/uninstPlugin/_jvmForPlugin contains the WebSphere Application Server SDK, Java technology edition used to uninstall the product
- *plugins_root*/uninstPlugin contains the uninstaller program
- *plugins_root*/bin contains the binary plug-ins for all supported Web servers
- *plugins_root*/logs contains log files
- *plugins_root*/properties contains version information
- *plugins_root*/roadmap contains the roadmap for the Plug-ins installation wizard

See “Selecting a Web server topology diagram and roadmap” on page 55 for an overview of the installation procedure.

See “Web server configuration” on page 63 for more information about the files involved in configuring a Web server.

See “Plug-ins configuration” on page 58 for information about the location of the plug-in configuration file.

See “Editing Web server configuration files” on page 102 for information about how the Plug-ins installation wizard configures supported Web servers.

Configuring multiple Web servers and remote stand-alone application servers

This topic describes installing a Web server plug-in that WebSphere Application Server provides to communicate with a particular brand of Web server. This procedure describes installing multiple Web servers and their Web server plug-ins for WebSphere Application Server on one machine and multiple application servers on another machine.

When multiple profiles exist, the plug-ins installer configures only the default profile. See “Plug-ins configuration” on page 58 for a description of the flow of logic that determines how the installer selects the profile to configure.

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

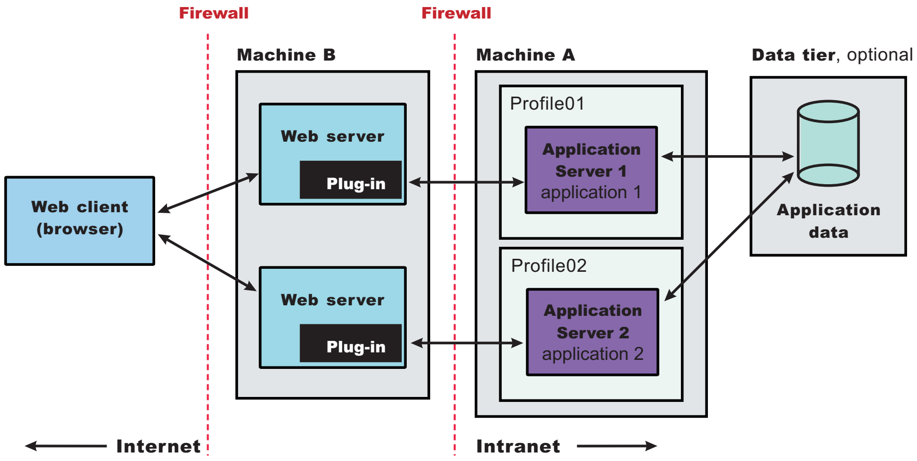
If the WebSphere Application Server product family does not provide a binary plug-in for a particular brand of Web server, then the Web server is not supported. The purpose of the binary plug-in is to provide the communication protocol between the Web server and the application server.

Suppose that you create a new profile. Suppose also that you want to use a Web server. You must install a new Web server for the new profile and use the Plug-ins installation wizard to install the binary plug-in module and to configure both the Web server and the application server.

If the Web server is not already installed, you can still install the plug-ins for future use. If the WebSphere Application Server product is not installed, you can still install the plug-ins. However, it is recommended that you install the Web server and the WebSphere Application Server product before installing the plug-ins for the supported Web server.

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.

This topic describes how to create the following topology:



Perform the following procedure to install the plug-ins and configure both Web servers and both application servers.

This topology lets each profile have unique applications, configuration settings, data, and log files, while sharing the same set of system files. Creating multiple profiles creates multiple application server environments that you can then dedicate to different purposes.

For example, each application server on a Web site can serve a different application. In another example, each application server can be a separate test environment that you assign to a programmer or a development team.

1. Log on to the operating system.

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

   ```
   umask
   ```

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

   ```
   umask 022
   ```

Windows When installing on a Windows system, you can create a Windows service to autostart the application server. To do so, your installer user account must have the following advanced user rights:

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

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

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

2. Install WebSphere Application Server - Express on Machine A.

   See Installing the product and additional software.

3. Install the IBM HTTP Server or another supported Web server on Machine B.
4. Launch the Plug-ins installation wizard on the machine with the Web server.

   Select the Plug-ins installation wizard from the launchpad or change directories to the plugin directory on the product disk or in the downloaded installation image and issue the install command.

5. Clear the check box for the roadmap or select the check box to view the roadmap, then click **Next**.

   If you are unsure of which installation scenario to follow, display the roadmap instead. Print and keep the roadmap as a handy overview of the installation steps.

   Press **Ctrl-P** to print the roadmap if the Web browser navigation controls and the menu bar are not present on the browser window that displays the Plug-ins roadmap. Press **Ctrl-W** to close the browser window if the navigation controls and the menu bar do not display. Or close the browser window with the window control in the title bar.

6. Read the license agreement and accept the agreement it if you agree to its terms. Click **Next** when you are finished.
7. If your system does not pass the prerequisites check, stop the installation, correct any problems, and restart the installation. If your system passes the prerequisites check, click **Next**.

   Look for the appropriate log file for information about missing prerequisites:

   - If you stop the installation, see the temporaryPluginInstallLog.txt file in the temporary directory of the user who installed the plug-ins. For example, the /tmp/temporaryPluginInstallLog.txt file might exist if the root user installed the plug-ins on a Linux or UNIX system.
   - If you continue the installation in spite of warnings about missing prerequisites, see the *plugins_root*/logs/install/log.txt file after the installation is complete.

   See Troubleshooting installation for more information about log files.

8. Select the type of Web server that you are configuring and click **Next**.

   The Plug-ins installation wizard panel prompts you to identify the *Web servers* to configure. Actually you can select only one Web server each time you run the Plug-ins installation wizard.

   Stop any Web server while you are configuring it. A step later in the procedure directs you to start the Web server as you begin the snoop servlet test.

   If you select the Web server identification option labeled **None**, the Web server installs the binary plug-ins but does not configure the Web server.

9. Select **Web server machine (remote)** and click **Next**.
10. Accept the default location for the installation root directory for the plug-ins. Click **Next**.

    You can type another new directory or click **Browse** to select an empty directory. The fully qualified path identifies the plug-ins installation root directory.

    The default location is shown in “Directory conventions,” on page 305.

    A possibility exists that the Web server might run on a platform that WebSphere Application Server does not support.

11. Click **Browse** to select the configuration file for your Web server, verify that the Web server port is correct, and then click **Next** when you are finished.

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

    The following list shows configuration files for supported Web servers:

    Apache HTTP Server
    apache_root/config/httpd.conf

    Domino Web Server
    names.nsf and Notes.jar

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

    The Plug-ins installation wizard verifies that the files exist but the wizard does not validate either file.

    IBM HTTP Server
    IHS_root/conf/httpd.conf

    Microsoft Internet Information Services (IIS)
    The Plug-ins installation wizard can determine the correct files to edit.

    Sun ONE Web Server 6.0 or Sun Java System Web Server, Version 6.1
    obj.conf and magnus.conf

    The wizard displays a naming panel for the nickname of the Web server definition.

12. Specify a nickname for the Web server. Click **Next** when you are finished.

    The wizard uses the value to name configuration folders in the plug-ins installation root directory. The wizard also uses the name in the configuration script for the application server to name the Web server definition.

    If the application server profile already has a Web server definition, delete the Web server definition before continuing. Use the following commands to delete the Web server definition:

    ```
    $AdminTask deleteServer { -serverName webserver1 -nodeName webserver1_node }
    $AdminTask removeUnmanagedNode { -nodeName webserver1_node }
    $AdminConfig save
    ```

13. Accept the default location for the plugin-cfg.xml file that the wizard creates on the Web server machine, then click **Next**.

    You can type a change to the value or click **Browse** to select a file in another location. If you do not accept the default location, the plugin-cfg.xml file must exist.

14. Identify the host name or IP address of Machine A, which is the application server machine, then click **Next**.

15. Examine the summary panel. Click **Next** when you are finished.

    The panel notifies you that you have manual steps to perform to complete the installation and configuration. The type of Web server, the nickname of the Web server, and the location of the plugin-cfg.xml file displays on the panel.

    The Plug-ins installation wizard creates the configure*web_server_name* script in the *plugins_root*/bin/ directory on Machine B.

    The Plug-ins installation wizard also creates the plugin-cfg.xml file in the *plugins_root*/config/ *web_server_name* directory.

    The Web server reads the plugin-cfg.xml file to determine the applications that the application server on Machine A can serve to the Web server on Machine B. Whenever the configuration changes, the application server regenerates the file. When regeneration occurs, propagate, or copy the actual plugin-cfg.xml file from the application server machine to the Web server machine. You can automatically propagate the file to the IBM HTTP Server product.

16. Click **Next** on the pre-installation summary panel to begin the installation or click **Back** to change any characteristics of the installation.

The panel specifies the plug-ins installation root directory, the Web server plug-ins feature, and the disk size of the code that installs when you click **Next**.

17. After the wizard installs the code and creates the uninstaller program, examine the post-installation summary panel. Click **Next** when you are finished to display the Plug-ins installation roadmap.

    The Plug-ins installation wizard installs the binary plug-in module. On a Linux system, for example, the installation creates the *plugins_root* directory. The *plugins_root*/config/*web_server_name* directory contains the plugin-cfg.xml file.

    The wizard displays the name and location of the configuration script and the plugin-cfg.xml file. The wizard also displays the type of Web server that is configured and the nickname of the Web server.

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

18. Close the road map and click **Finish** to exit the wizard.

    Log files from the installation are in the *plugins_root*/logs/install directory.

19. Copy the configure*web_server_name*.sh script on Linux and UNIX systems, the configure*web_server_name*.bat script on Windows systems, or the configure*web_server_name* script on i5/OS from Machine B to the *app_server_root* /bin directory on Machine A.

    On a Linux or UNIX system, the file is configure*web_server_name*.sh. On a Windows system, the file is configure*web_server_name*.bat. For example, on a Linux system with an IBM HTTP Server named webserver1 in the default location, copy *plugins_root*/bin/configurewebserver1.sh from Machine B to the *app_server_root*/bin directory on Machine A.

    If one platform is a Linux or UNIX system and the other is a Windows platform, copy the script from the crossPlatformScripts directory. For example:

    - Linux *plugins_root*/`bin/configure`*`Web_server_name`*`.sh`
    - Windows *plugins_root*/`bin/crossPlatformScripts/configure`*`Web_server_name`*`.bat`

20. Compensate for file encoding differences to prevent script failure.

    The content of the configure*web_server_name*.bat script or the configure*web_server_name*.sh script can be corrupt if the default file encoding of the two machines differs. This scenario is possible when one machine is set up for a double-byte character set (DBCS) locale and the other machine is not.

    Determine the file encoding and use one of the following procedures to circumvent the failure. To determine the default file encoding, run the appropriate command.

    - Windows Run the following command on a Windows machine:

      ```
      CHCP
      ```

    - Linux Run the following command on a Linux or UNIX machine:

      ```
      locale
      ```

    Use the result of the command on each machine as the value of the *web_server_machine_encoding* variable and the *application_server_machine_encoding* variable in one of the following procedures.

    Procedures for compensating for encoding differences

    Suppose that the Web server is running on a Linux machine and Network Deployment is running on a Windows machine.

    Web server running on a Linux or UNIX system

    Run the following command on the Linux or UNIX system to encode the script file that configures the Web server definition, before you FTP the file to the Windows machine in binary mode:

    ```
    iconv -f web_server_machine_encoding \
       -t application_server_machine_encoding \
       configureweb_server_name.bat
    ```

    Omit the Linux and UNIX continuation characters (\) if you enter the command on one line.

Important: The name of the Web server (nick name) is used in the name of the script file. The name cannot contain characters from a double-byte character set (DBCS) if you intend to set up IBM HTTP Server for automatic propagation.

Suppose that the Web server is running on a Windows machine and Network Deployment is running on a Linux or UNIX machine.

Web server running on a Windows machine

Run the following command on the Linux or UNIX system to encode the script file that configures the Web server definition, after you FTP the file in binary mode:

```
iconv -f web_server_machine_encoding \
   -t application_server_machine_encoding \
   configureweb_server_name.sh
```

Omit the Linux and UNIX continuation characters (\) if you enter the command on one line.

If the conversion mapping is not supported by the iconv command on your system, copy the contents of the Web server configuration script to a clip board and paste it onto the machine where the application server is running.

21. Start the application server on Machine A.

    Use the startServer command, for example:

    - Linux `profile_root/bin/startServer.sh server1`
    - Windows `profile_root\bin\startServer server1`

22. Open a command window and change to the profile directory where the Web server should be assigned. Run the script that you copied to Machine A. You will need the following parameters:

    ```
    Profile Name
    (Optional) Admin user ID
    (Optional) Admin user password
    ```

    For example: *configurewebserver1.sh Dmgr01 myUserID myPassword*

    The webserver will be configured via wsadmin.

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

    Linux To configure Domino, you must set the WAS_PLUGIN_CONFIG_FILE environment variable. On Linux and UNIX-based platforms, sourcing a script to the parent shell allows child processes to inherit the exported variables. On Windows systems, run the script as you would run any other command. Sourcing is automatic on Windows systems.

    a. Open a command window.

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

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

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

    The script is also in the *lotus_root*/notesdata directory on Linux and UNIX systems.

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

24. Start the Snoop servlet to verify the ability of the Web server to retrieve an application from the Application Server.

    Test your environment by starting your Application Server, your Web server, and using the snoop servlet with an IP address.

    a. Start the Application Server.

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

       - Linux ./startServer.sh server1
       - Windows startServer server1

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

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

To start the IBM HTTP Server from the command line:

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

- Linux ./apachectl start
- Windows apache

c. Point your browser to http://localhost:9080/snoop to test the internal HTTP transport provided by the Application Server. Point your browser to http://*Host_name_of_Web_server_machine*/snoop to test the Web server plug-in.

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

d. Verify that snoop is running.

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

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

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

1) Create a user=adminUser, password=adminPassword in the *IHS_root* /conf/admin.passwd file. For example: c:\ws\ihs60\bin\htpasswd -cb c:\ws\ihs60\conf\admin.passwd adminUser adminPassword
2) Use the administrative console of the Application Server to enter the User ID and password information that you created for the administrative user of IBM HTTP Server. Go to **Servers > Web server > *Web_server_definition* > Remote Web server administration**. Set the following values: admin Port=8008, User Id=adminUser, Password=adminPassword.
3) Set the correct read/write permissions for the httpd.conf file and the plugin-cfg.xml file. See the *IHS_root* /logs/admin_error.log file for more information.

Automatic propagation of the plug-in configuration file requires the IBM HTTP administrative server to be up and running. If you are managing an IBM HTTP Server using the WebSphere Application Server administrative console, the following error might display:

```
"Could not connect to IHS Administration server error"
```

Perform the following procedure to correct the error:

1) Verify that the IBM HTTP Server administration server is running.
2) Verify that the Web server host name and the port that is defined in the WebSphere Application Server administrative console matches the IBM HTTP Server administration host name and port.
3) Verify that the fire wall is not preventing you from accessing the IBM HTTP Server administration server from the WebSphere Application Server administrative console.
4) Verify that the user ID and password that is specified in the WebSphere Application Server administrative console under remote managed, is created in the admin.passwd file, using the htpasswd command.
5) If you are trying to connect securely, verify that you export the IBM HTTP Server administration server keydb personal certificate into the WebSphere Application Server key database as a signer certificate. This key database is specified by the com.ibm.ssl.trustStore directive in the sas.client.props file in the profile where your administrative console is running. This consideration is primarily for self-signed certificates.
6) If you still have problems, check the IBM HTTP Server admin_error.log file and the WebSphere Application Server logs (trace.log file) to determine the cause of the problem.

25. Regenerate the plugin-cfg.xml file on Machine A using the administrative console. Click **Servers > Web server**. Select the Web server, then click **Generate Plug-in**.

    During the installation of the plug-ins, the default plugin-cfg.xml file is installed on Machine B in the *plugins_root*/config/*web_server_name* directory. The Web server plug-in configuration service regenerates the plugin-cfg.xml file automatically. To use the current plugin-cfg.xml file from the application server, propagate the plugin-cfg.xml file as described in the next step.

    This step shows you how to regenerate the plugin-cfg.xml file. WebSphere Application Server products are configured to automatically regenerate the file each time a significant event occurs. Such events include installing applications on the application server and the Web server, for example. Creating a new virtual host is another such event.
26. Propagate the plugin-cfg.xml file from the application server to the Web server using the administrative console. Click **Servers > Web server**. Select the Web server, then click **Propagate Plug-in**. Web servers other than IBM HTTP Server require manual propagation.

    The Web server plug-in configuration service propagates the plugin-cfg.xml file automatically for IBM HTTP Server 6.0 only. For all other Web servers, propagate the plug-in configuration file by manually copying the plugin-cfg.xml file from the *profile_root*/config/cells/*cell_name*/nodes/*node_name*/servers/*web_server_name* directory on Machine A to the *plugins_root*/config/*web_server_name* directory on Machine B.
27. Create the second application server profile using the Profile Management tool on Machine A. Make the profile the default profile during the profile creation by selecting the check box on the appropriate panel.

    The script that the Plug-ins installation wizard creates only works on the default profile. So, this script can create only a Web server definition on the profile that is the default profile at the time that the script runs.
28. Install a second IBM HTTP Server or another supported Web server on Machine B.
29. On Machine B, install the Web server plug-ins to configure the second Web server using the Plug-ins installation wizard. Both Web servers share a single installation of the plug-in binaries but must be configured individually.
30. The Plug-ins installation wizard creates a script named configure*web_server_name* for the second Web server. The script is in the *plugins_root*/bin directory on Machine B. Copy the script to the *app_server_root*/bin directory on Machine A.
31. Start the second application server.
32. Run the configure*web_server_name* script to create a Web server definition in the administrative console. You can then use the administrative console to manage the Web server.
33. Propagate the plugin-cfg.xml file from the second application server to the Web server using the administrative console. Click **Servers > Web server > Propagate Plug-in**. Web servers other than IBM HTTP Server require manual propagation.
34. Run the snoop servlet on the second Web server to verify that it is operational.

This procedure results in installing two or more application servers on one machine and installing dedicated Web servers on another machine. This procedure installs the Web server plug-ins for both Web servers and configures both Web servers and both application servers.

See “Selecting a Web server topology diagram and roadmap” on page 55 for an overview of the installation procedure.

See “Editing Web server configuration files” on page 102 for information about how the Plug-ins installation wizard configures supported Web servers.

See “Web server configuration” on page 63 for more information about the files involved in configuring a Web server.

For IHS Web servers, you can stop and start the Web server and propagate the plugin-cfg.xml file from the WebSphere Application Server machine to the Web server machine. For all other Web servers, you can not start/stop or propagate the plugin-cfg.xml file in the admin console. You will need to propagate the plugin-cfg.xml file manually. The following three steps describes how to perform manual propagation:

1. After completion of configuration with Web servers other than IHS 6.x, verify that the plugin-cfg.xml file exists at <WAS_HOME>/profiles/<PROFILE_HOME>/config/cells/<CELL_NAME>/nodes/<SERVER_NAME>/servers/<WEBSERVER_DEFINITION>
2. Transfer the above plugin-cfg.xml to replace <PLUGIN_HOME>/config/<WEBSERVER_DEFINITION>/plugin-xfg.xml
3. Restart the Web server and corresponding profile.

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

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

When multiple profiles exist, the plug-ins installer configures only the default profile. See “Plug-ins configuration” on page 58 for a description of the flow of logic that determines how the installer selects the profile to configure.

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

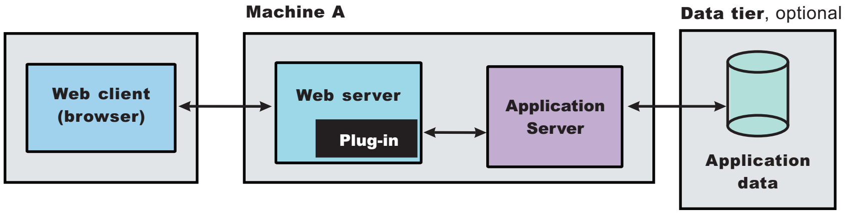
If the WebSphere Application Server product family does not provide a binary plug-in for a particular brand of Web server, then the Web server is not supported. The purpose of the binary plug-in is to provide the communication protocol between the Web server and the application server.

Suppose that you create a new profile. Suppose also that you want to use a Web server. You must install a new Web server for the new profile and use the Plug-ins installation wizard to install the binary plug-in module and to configure both the Web server and the application server.

If the Web server is not already installed, you can still install the plug-ins for future use. If the WebSphere Application Server product is not installed, you can still install the plug-ins. However, it is recommended that you install the Web server and the WebSphere Application Server product before installing the plug-ins for the supported Web server.

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

This topic describes how to create the following topology:



The wizard performs three steps to properly configure a Web server for Version 6. The wizard performs the steps in the following order:

1. The wizard installs the unique binary plug-in module for the supported Web server after collecting the following information:
   - The type of Web server
   - The location of the configuration file for the Web server that the wizard configures
   - The plug-ins installation root directory for the Web server plug-in modules that the wizard installs
   - The installation root directory of the WebSphere Application Server product, where the wizard creates a Web server definition

   If administrative security is enabled, the Plug-ins installation wizard prompts for the administrative user ID and password for the profile.
2. The wizard prompts you for the location of the configuration file or files for the Web server. You must browse for and select the correct file.

   The wizard edits the configuration file or files for a Web server by creating directives that point to the location of the binary plug-in module and the plug-in configuration file.

   The name of the binary plug-in module varies per Web server type. The plug-in configuration file is always the plugin-cfg.xml file.
3. The wizard creates a Web server definition in the configuration of the application server unless one already exists.

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

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

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

1. Log on to the operating system.

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

   ```
   umask
   ```

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

   ```
   umask 022
   ```

   Windows When installing on a Windows system, you can create a Windows service to autostart the application server. To do so, your installer user account must have the following advanced user rights:
   - Act as part of the operating system
   - Log on as a service

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

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

2. Install WebSphere Application Server - Express on Machine A.

   See Installing the product and additional software.

3. Install the IBM HTTP Server or another supported Web server on Machine B.

4. Launch the Plug-ins installation wizard on the machine with the Web server.

   Select the Plug-ins installation wizard from the launchpad or change directories to the plugin directory on the product disk or in the downloaded installation image and issue the install command.

5. Stop the stand-alone application server before installing the Web server plug-ins. For example, assuming that the profile name is default, use one of the following commands.

   - AIX /usr/ IBM/ WebSphere/ AppServer/ profiles/ *default*/ bin/ stopServer.sh server1
   - Linux /opt/ IBM/ WebSphere/ AppServer/ profiles/ *default*/ bin/ stopServer.sh server1
   - Windows C:\ Program Files\ IBM\ WebSphere\ AppServer\ profiles\ *default*\ bin\ stopServer.sh server1

6. Clear the check box for the roadmap or select the check box to view the roadmap, then click **Next**.

   If you are unsure of which installation scenario to follow, display the roadmap instead. Print and keep the roadmap as a handy overview of the installation steps.

   Press **Ctrl-P** to print the roadmap if the Web browser navigation controls and the menu bar are not present on the browser window that displays the Plug-ins roadmap. Press **Ctrl-W** to close the browser window if the navigation controls and the menu bar do not display. Or close the browser window with the window control in the title bar.

7. Read the license agreement and accept the agreement it if you agree to its terms. Click **Next** when you are finished.

8. If your system does not pass the prerequisites check, stop the installation, correct any problems, and restart the installation. If your system passes the prerequisites check, click **Next**.

   Look for the appropriate log file for information about missing prerequisites:

   - If you stop the installation, see the temporaryPluginInstallLog.txt file in the temporary directory of the user who installed the plug-ins. For example, the /tmp/temporaryPluginInstallLog.txt file might exist if the root user installed the plug-ins on a Linux or UNIX system.
   - If you continue the installation in spite of warnings about missing prerequisites, see the *plugins_root*/logs/install/log.txt file after the installation is complete.

   See Troubleshooting installation for more information about log files.

9. Select the type of Web server that you are configuring and click **Next**.

   The Plug-ins installation wizard panel prompts you to identify the *Web servers* to configure. Actually you can select only one Web server each time you run the Plug-ins installation wizard.

   Stop any Web server while you are configuring it. A step later in the procedure directs you to start the Web server as you begin the snoop servlet test.

   If you select the Web server identification option labeled **None**, the Web server installs the binary plug-ins but does not configure the Web server.

10. Select **Application Server machine (local)** and click **Next**.

11. Accept the default location for the installation root directory for the plug-ins. Click **Next**.

    You can type another new directory or click **Browse** to select an empty directory. The fully qualified path identifies the plug-ins installation root directory.

    The default location is shown in "Directory conventions," on page 305.

    A possibility exists that the Web server might run on a platform that WebSphere Application Server does not support.

12. Click **Browse** on the Application Server Installation Location panel to browse for the location of the application server profile if necessary. Click **Next** when the installation root directory is correct.

    The fully qualified path identifies the installation root directory for the WebSphere Application Server product, which is referred to as the app_server_root throughout the information center.

13. Enter an administrative user ID and password if administrative security is enabled on the application server.

14. Click **Browse** to select the configuration file for your Web server, verify that the Web server port is correct, and then click **Next** when you are finished.

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

    The following list shows configuration files for supported Web servers:

    Apache HTTP Server

    apache_root/config/httpd.conf

    Domino Web Server

    names.nsf and Notes.jar

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

    The Plug-ins installation wizard verifies that the files exist but the wizard does not validate either file.

    IBM HTTP Server

    IHS_root/conf/httpd.conf

    Microsoft Internet Information Services (IIS)

    The Plug-ins installation wizard can determine the correct files to edit.

    Sun ONE Web Server 6.0 or Sun Java System Web Server, Version 6.1

    obj.conf and magnus.conf

    The wizard displays a naming panel for the nickname of the Web server definition.

15. Specify a nickname for the Web server. Click **Next** when you are finished.

    The wizard uses the value to name configuration folders in the plug-ins installation root directory. The wizard also uses the name in the configuration script for the application server to name the Web server definition.

    If the application server profile already has a Web server definition, delete the Web server definition before continuing. Use the following commands to delete the Web server definition:

    ```
    $AdminTask deleteServer { -serverName webserver1 -nodeName webserver1_node }
    $AdminTask removeUnmanagedNode { -nodeName webserver1_node }
    $AdminConfig save
    ```

16. Specify the location for the plugin-cfg.xml file and click **Next**.

    This is a critical selection.

    See “Plug-ins configuration” on page 58 for a description of the logic that determines what path is configured by default. The following possibilities exist for the default location of the plug-in configuration file. The wizard determines the characteristics of the application server to determine the best path for the file:

    - An application server that has an existing Web server definition has the following path:

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

    - A stand-alone application server that does not have a Web server definition has the following path:

      ```
      profile_root
      /config/cells/cell_name/nodes/
      web_server_name_node/servers/
      web_server_name/plugin-cfg.xml
      ```

    You can accept the default value if the application server does not have a Web server definition.

    Using an existing Web server definition

    If the application server has a Web server definition, the wizard cannot create a new Web server definition within the application server configuration. However, the wizard can reconfigure the Web server. Click **Browse** and select the existing plugin-cfg.xml file in the application server configuration.

To find the plug-in configuration file in a stand-alone application server, follow this file path:

```
profile_root
   /config/cells/cell_name/nodes/
   web_server_name_node/servers/
   web_server_name/plugin-cfg.xml
```

If the existing *web_server_name* is different than the nickname that you gave the Web server in the wizard, click **Back** to return to the naming panel for the Web server and change the name to match the existing Web server definition name.

If you cannot find an existing plugin-cfg.xml file after all, you must install the temporary plugin-cfg.xml file. In such a case, type the path to the plug-ins installation root directory so that the wizard can install the temporary plug-in configuration file:

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

17. Click **Next** after verifying the characteristics of the plug-ins installation or click **Back** to make changes.

    Once created, a Web server definition on a stand-alone application server node cannot be removed except through scripting. (See “Uninstalling the Web server plug-ins for WebSphere Application Server” on page 115 for the procedure.)

    You can, however, reuse the same definition for a different type of Web server. Run the Plug-ins installation wizard to configure a new Web server in that situation. The Plug-ins installation wizard configures the new Web server to use the existing plugin-cfg.xml file.

18. Click **Next** on the pre-installation summary panel to begin the installation or click **Back** to change any characteristics of the installation. The wizard begins installing the plug-ins and configuring the Web server and the application server.

    The wizard shows an installation status panel as it installs the plug-ins.

    The wizard displays the Installation summary panel at the completion of the installation.

19. Verify the success of the installation on the Installation summary panel and click **Finish** to exit the wizard.

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

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

    ▶ Linux To configure Domino, you must set the WAS_PLUGIN_CONFIG_FILE environment variable. On Linux and UNIX-based platforms, sourcing a script to the parent shell allows child processes to inherit the exported variables. On Windows systems, run the script as you would run any other command. Sourcing is automatic on Windows systems.

    a. Open a command window.

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

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

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

    The script is also in the *lotus_root*/notesdata directory on Linux and UNIX systems.

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

21. Start the Snoop servlet to verify the ability of the Web server to retrieve an application from the Application Server.

    Test your environment by starting your Application Server, your Web server, and using the snoop servlet with an IP address.

    a. Start the Application Server.

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

       - ▶ Linux ./startServer.sh server1
       - ▶ Windows startServer server1

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

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

   To start the IBM HTTP Server from the command line:

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

   - Linux ./apachectl start
   - Windows apache

c. Point your browser to http://localhost:9080/snoop to test the internal HTTP transport provided by the Application Server. Point your browser to http://*Host_name_of_Web_server_machine*/snoop to test the Web server plug-in.

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

d. Verify that snoop is running.

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

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

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

   1) Create a user=adminUser, password=adminPassword in the *IHS_root* /conf/admin.passwd file. For example: c:\ws\ihs60\bin\htpasswd -cb c:\ws\ihs60\conf\admin.passwd adminUser adminPassword
   2) Use the administrative console of the Application Server to enter the User ID and password information that you created for the administrative user of IBM HTTP Server. Go to **Servers > Web server > *Web_server_definition* > Remote Web server administration**. Set the following values: admin Port=8008, User Id=adminUser, Password=adminPassword.
   3) Set the correct read/write permissions for the httpd.conf file and the plugin-cfg.xml file. See the *IHS_root* /logs/admin_error.log file for more information.

   Automatic propagation of the plug-in configuration file requires the IBM HTTP administrative server to be up and running. If you are managing an IBM HTTP Server using the WebSphere Application Server administrative console, the following error might display:

   ```
   "Could not connect to IHS Administration server error"
   ```

   Perform the following procedure to correct the error:

   1) Verify that the IBM HTTP Server administration server is running.
   2) Verify that the Web server host name and the port that is defined in the WebSphere Application Server administrative console matches the IBM HTTP Server administration host name and port.
   3) Verify that the fire wall is not preventing you from accessing the IBM HTTP Server administration server from the WebSphere Application Server administrative console.
   4) Verify that the user ID and password that is specified in the WebSphere Application Server administrative console under remote managed, is created in the admin.passwd file, using the htpasswd command.
   5) If you are trying to connect securely, verify that you export the IBM HTTP Server administration server keydb personal certificate into the WebSphere Application Server key database as a signer certificate. This key database is specified by the com.ibm.ssl.trustStore directive in the sas.client.props file in the profile where your administrative console is running. This consideration is primarily for self-signed certificates.
   6) If you still have problems, check the IBM HTTP Server admin_error.log file and the WebSphere Application Server logs (trace.log file) to determine the cause of the problem.

The installation of the binary plug-in modules results in the creation of the Plugins directory and several subdirectories. The following directories are among those created on a Linux system, for example:

- *plugins_root*/uninstPlugin/_jvmForPlugin contains the WebSphere Application Server SDK, Java technology edition used to uninstall the product
- *plugins_root*/uninstPlugin contains the uninstaller program
- *plugins_root*/bin contains the binary plug-ins for all supported Web servers
- *plugins_root*/logs contains log files
- *plugins_root*/properties contains version information
- *plugins_root*/roadmap contains the roadmap for the Plug-ins installation wizard

The Plug-ins installation wizard creates a Web server definition within the application server profile unless one already exists.

The Plug-ins installation wizard configures the Web server to use the *profile_root*/plugin-cfg.xml file.

The application server regenerates the Web server plug-in configuration file, plugin-cfg.xml whenever an event occurs that affects the file. Such events include the addition or removal of an application, server, or virtual host. The stand-alone application server regenerates the file in the following location:

```
profile_root
   /config/cells/cell_name/nodes/
   web_server_name_node/servers/
   web_server_name/plugin-cfg.xml
```

You can start a stand-alone application server and the Web server immediately after installing of the binary plug-in for the local Web server. Open the administrative console of the application server after you start the server and save the changed configuration.

See “Selecting a Web server topology diagram and roadmap” on page 55 for an overview of the installation procedure.

See “Plug-ins configuration” on page 58 for information about the location of the plug-in configuration file.

See “Web server configuration” on page 63 for information about the files involved in configuring a Web server.

See “Editing Web server configuration files” on page 102 for information about how the Plug-ins installation wizard configures supported Web servers.

See “Installing Web server plug-ins” on page 53 for information about other installation scenarios for installing Web server plug-ins.

Configuring a Web server for a non-default profile

This topic describes installing a Web server plug-in and configuring an application server that is not the default profile.

When multiple profiles exist, the plug-ins installer configures only the default profile. See “Plug-ins configuration” on page 58 for a description of the flow of logic that determines how the installer selects the profile to configure.

If you intend to create a Web server definition for an application server that is not the default profile, you must force the Plug-ins installation wizard to configure a profile that is not the default profile.

This procedure assumes that you have installed the Web server on one machine and the WebSphere Application Server product on a separate machine. This procedure also works when the Web server and the application server profile are on the same machine. In such a case, select the Remote installation type when installing the plug-ins so that the Plug-ins installation wizard creates the configuration script for the application server.

This procedure describes how to create a Web server definition for an application server that is not the default profile. This procedure describes selecting a remote installation scenario and issuing a command for the configuration script that points the command to a profile that you select.

The ordinary behavior of the configuration script is to work on the default profile on the machine where the script runs. This procedure describes how to redirect the script to configure a profile that is not the default.

1. Log on to the operating system.

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

   ```
   umask
   ```

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

   ```
   umask 022
   ```

   Windows When installing on a Windows system, you can create a Windows service to autostart the application server. To do so, your installer user account must have the following advanced user rights:

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

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

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

2. Launch the Plug-ins installation wizard on the machine with the Web server.

   Select the Plug-ins installation wizard from the launchpad or change directories to the plugin directory on the product disk or in the downloaded installation image and issue the install command.

3. Clear the check box for the roadmap or select the check box to view the roadmap, then click **Next**.

   If you are unsure of which installation scenario to follow, display the roadmap instead. Print and keep the roadmap as a handy overview of the installation steps.

   Press **Ctrl-P** to print the roadmap if the Web browser navigation controls and the menu bar are not present on the browser window that displays the Plug-ins roadmap. Press **Ctrl-W** to close the browser window if the navigation controls and the menu bar do not display. Or close the browser window with the window control in the title bar.

4. Read the license agreement and accept the agreement it if you agree to its terms. Click **Next** when you are finished.

5. If your system does not pass the prerequisites check, stop the installation, correct any problems, and restart the installation. If your system passes the prerequisites check, click **Next**.

   Look for the appropriate log file for information about missing prerequisites:

   - If you stop the installation, see the temporaryPluginInstallLog.txt file in the temporary directory of the user who installed the plug-ins. For example, the /tmp/temporaryPluginInstallLog.txt file might exist if the root user installed the plug-ins on a Linux or UNIX system.
   - If you continue the installation in spite of warnings about missing prerequisites, see the *plugins_root*/logs/install/log.txt file after the installation is complete.

See Troubleshooting installation for more information about log files.

6. Select the type of Web server that you are configuring and click **Next**.

   The Plug-ins installation wizard panel prompts you to identify the *Web servers* to configure. Actually you can select only one Web server each time you run the Plug-ins installation wizard.

   Stop any Web server while you are configuring it. A step later in the procedure directs you to start the Web server as you begin the snoop servlet test.

   If you select the Web server identification option labeled **None**, the Web server installs the binary plug-ins but does not configure the Web server.

7. Select **Web server machine (remote)** and click **Next**.

8. Accept the default location for the installation root directory for the plug-ins. Click **Next**.

   You can type another new directory or click **Browse** to select an empty directory. The fully qualified path identifies the plug-ins installation root directory.

   The default location is shown in "Directory conventions," on page 305.

   A possibility exists that the Web server might run on a platform that WebSphere Application Server does not support.

9. Click **Browse** to select the configuration file for your Web server, verify that the Web server port is correct, and then click **Next** when you are finished.

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

   The following list shows configuration files for supported Web servers:

   Apache HTTP Server
      apache_root/config/httpd.conf

   Domino Web Server
      names.nsf and Notes.jar

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

      The Plug-ins installation wizard verifies that the files exist but the wizard does not validate either file.

   IBM HTTP Server
      IHS_root/conf/httpd.conf

   Microsoft Internet Information Services (IIS)
      The Plug-ins installation wizard can determine the correct files to edit.

   Sun ONE Web Server 6.0 or Sun Java System Web Server, Version 6.1
      obj.conf and magnus.conf

   The wizard displays a naming panel for the nickname of the Web server definition.

10. Specify a nickname for the Web server. Click **Next** when you are finished.

    The wizard uses the value to name configuration folders in the plug-ins installation root directory. The wizard also uses the name in the configuration script for the application server to name the Web server definition.

    If the application server profile already has a Web server definition, delete the Web server definition before continuing. Use the following commands to delete the Web server definition:

    ```
    $AdminTask deleteServer { -serverName webserver1 -nodeName webserver1_node }
    $AdminTask removeUnmanagedNode { -nodeName webserver1_node }
    $AdminConfig save
    ```

11. Type a value for the location of the plugin-cfg.xml file.

    For a stand-alone application server node, the application server creates the file in the *profile_root* /config/cells/ *cell_name* /nodes/ *web_server_name*_node/servers/ *web_server_name* /plugin-cfg.xml file path.

Specify the location in the field so that the Web server can access the file when the Web server and the application server are on the same machine. Accept the default value if the Web server and the application server are on separate machines.

The location of the plugin-cfg.xml file is used in the Web server configuration file. If you cannot enter the location correctly at this point, you can edit the Web server configuration file manually to correct the location so that the location points to the plugin-cfg.xml file in the non-default application server profile. On a remote installation, the default location is within the plug-ins installation root directory. Propagation copies the current file from the application server machine to the Web server machine.

12. Identify the host name or IP address of the application server machine, then click **Next**.
13. Examine the summary panel. Click **Next** when you are finished.

    The panel notifies you that you have manual steps to perform to complete the installation and configuration. The type of Web server, the nickname of the Web server, and the location of the plugin-cfg.xml file displays on the panel.

    The Plug-ins installation wizard creates the configure*web_server_name* script in the *plugins_root*/bin/ directory.

    The Plug-ins installation wizard also creates a default plugin-cfg.xml file in the *plugins_root*/config/*web_server_name* directory.
14. Click **Next** on the pre-installation summary panel to begin the installation or click **Back** to change any characteristics of the installation.

    The panel specifies the plug-ins installation root directory, the Web server plug-ins feature, and the disk size of the code that installs when you click **Next**.
15. After the wizard installs the code and creates the uninstaller program, examine the post-installation summary panel. Click **Next** when you are finished to display the Plug-ins installation roadmap.

    The Plug-ins installation wizard installs the binary plug-in module. On a Linux system, for example, the installation creates the *plugins_root* directory. The *plugins_root*/config/*web_server_name* directory contains the plugin-cfg.xml file.

    The wizard displays the name and location of the configuration script and the plugin-cfg.xml file. The wizard also displays the type of Web server that is configured and the nickname of the Web server.

    If a problem occurs and the installation is unsuccessful, examine the logs in the *plugins_root*/logs directory. Correct any problems and reinstall.
16. Close the road map and click **Finish** to exit the wizard.

    Log files from the installation are in the *plugins_root*/logs/install directory.
17. Copy the configure*web_server_name*.sh script on Linux and UNIX systems, the configure*web_server_name* script on i5/OS, or the configure*web_server_name*.bat script on Windows systems to the *app_server_root*/bin directory on the application server machine.

    For example, on a Linux system with an IBM HTTP Server named webserver1 in the default location, copy /opt/IBM/WebServer/Plugins/bin/configurewebserver1.sh to the /opt/IBM/WebSphere/AppServer/bin directory on the application server machine.

    If one platform is a Linux or UNIX system and the other is a Windows platform, copy the script from the crossPlatformScripts directory. For example:

    - Linux *plugins_root*/`bin/configureWeb_server_name.sh`
    - Windows *plugins_root*/`bin/`**`crossPlatformScripts`**`/configureWeb_server_name.bat`
18. Compensate for file encoding differences to prevent script failure.

    The content of the configure*web_server_name*.bat script or the configure*web_server_name*.sh script can be corrupt if the default file encoding of the two machines differs. This scenario is possible when one machine is set up for a double-byte character set (DBCS) locale and the other machine is not.

    Determine the file encoding and use one of the following procedures to circumvent the failure. To determine the default file encoding, run the appropriate command.

    - Windows Run the following command on a Windows machine:

```
CHCP
```

- Linux Run the following command on a Linux or UNIX machine:

  ```
  locale
  ```

Use the result of the command on each machine as the value of the *web_server_machine_encoding* variable and the *application_server_machine_encoding* variable in one of the following procedures.

Procedures for compensating for encoding differences

Suppose that the Web server is running on a Linux machine and Network Deployment is running on a Windows machine.

Web server running on a Linux or UNIX system

Run the following command on the Linux or UNIX system to encode the script file that configures the Web server definition, before you FTP the file to the Windows machine in binary mode:

```
iconv -f web_server_machine_encoding \
   -t application_server_machine_encoding \
   configureweb_server_name.bat
```

Omit the Linux and UNIX continuation characters (\) if you enter the command on one line.

Important: The name of the Web server (nick name) is used in the name of the script file. The name cannot contain characters from a double-byte character set (DBCS) if you intend to set up IBM HTTP Server for automatic propagation.

Suppose that the Web server is running on a Windows machine and Network Deployment is running on a Linux or UNIX machine.

Web server running on a Windows machine

Run the following command on the Linux or UNIX system to encode the script file that configures the Web server definition, after you FTP the file in binary mode:

```
iconv -f web_server_machine_encoding \
   -t application_server_machine_encoding \
   configureweb_server_name.sh
```

Omit the Linux and UNIX continuation characters (\) if you enter the command on one line.

If the conversion mapping is not supported by the iconv command on your system, copy the contents of the Web server configuration script to a clip board and paste it onto the machine where the application server is running.

19. Copy the configure*web_server_name*.sh script on Linux and UNIX systems, the configure*web_server_name*.bat script on Windows systems, or the configure*web_server_name* script on i5/OS from Machine B to the *app_server_root* /bin directory on Machine A.

    Use the startServer command to start the application server, for example:

    - Linux *profile_root*/bin/startServer.sh server1
    - Windows *profile_root*\bin\startServer server1
20. Stop the Web server if it is running.
21. Run the script on the application server machine to create a Web server definition for the non-default profile. The profile name should be passed as a parameter when the script is executed.

    Open a command window to run the script that you copied to Machine A.

    As soon as the Web server definition is created, the application server creates a plugin-cfg.xml file for the Web server. For example, the file on a Linux system might have the following file path: *profile_root*/config/cells/*cellname*/nodes/webserver1_node/servers/webserver1/plugin-cfg.xml
22. Linux **Domino Web Server only:** Set the WAS_PLUGIN_CONFIG_FILE environment variable.

    Linux To configure Domino, you must set the WAS_PLUGIN_CONFIG_FILE environment variable. On Linux and UNIX-based platforms, sourcing a script to the parent shell allows child processes to inherit the exported variables. On Windows systems, run the script as you would run any other command. Sourcing is automatic on Windows systems.

a. Open a command window.

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

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

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

The script is also in the *lotus_root*/notesdata directory on Linux and UNIX systems.

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

23. Start the Snoop servlet to verify the ability of the Web server to retrieve an application from the Application Server.

Test your environment by starting your Application Server, your Web server, and using the snoop servlet with an IP address.

a. Start the Application Server.

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

- Linux ./startServer.sh server1
- Windows startServer server1

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

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

To start the IBM HTTP Server from the command line:

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

- Linux ./apachectl start
- Windows apache

c. Point your browser to http://localhost:9080/snoop to test the internal HTTP transport provided by the Application Server. Point your browser to http://*Host_name_of_Web_server_machine*/snoop to test the Web server plug-in.

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

d. Verify that snoop is running.

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

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

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

1) Create a user=adminUser, password=adminPassword in the *IHS_root* /conf/admin.passwd file. For example: c:\ws\ihs60\bin\htpasswd -cb c:\ws\ihs60\conf\admin.passwd adminUser adminPassword

2) Use the administrative console of the Application Server to enter the User ID and password information that you created for the administrative user of IBM HTTP Server. Go to **Servers > Web server >** ***Web_server_definition*** **> Remote Web server administration**. Set the following values: admin Port=8008, User Id=adminUser, Password=adminPassword.

3) Set the correct read/write permissions for the httpd.conf file and the plugin-cfg.xml file. See the *IHS_root* /logs/admin_error.log file for more information.

Automatic propagation of the plug-in configuration file requires the IBM HTTP administrative server to be up and running. If you are managing an IBM HTTP Server using the WebSphere Application Server administrative console, the following error might display:

```
"Could not connect to IHS Administration server error"
```

Perform the following procedure to correct the error:

1) Verify that the IBM HTTP Server administration server is running.
2) Verify that the Web server host name and the port that is defined in the WebSphere Application Server administrative console matches the IBM HTTP Server administration host name and port.
3) Verify that the fire wall is not preventing you from accessing the IBM HTTP Server administration server from the WebSphere Application Server administrative console.
4) Verify that the user ID and password that is specified in the WebSphere Application Server administrative console under remote managed, is created in the admin.passwd file, using the htpasswd command.
5) If you are trying to connect securely, verify that you export the IBM HTTP Server administration server keydb personal certificate into the WebSphere Application Server key database as a signer certificate. This key database is specified by the com.ibm.ssl.trustStore directive in the sas.client.props file in the profile where your administrative console is running. This consideration is primarily for self-signed certificates.
6) If you still have problems, check the IBM HTTP Server admin_error.log file and the WebSphere Application Server logs (trace.log file) to determine the cause of the problem.

This procedure results in the installation of the Web server plug-ins for WebSphere Application Server on a Web server machine. After modification, the resulting script creates a Web server definition for an application server that is not the default profile. The Plug-ins installation wizard also configures the Web server to support the application server.

If you can see the Snoop servlet through the Web server, you have successfully configured the Web server and the application server.

The installation of the binary plug-in modules results in the creation of the Plugins directory and several subdirectories. The following directories are among those created on a Linux system, for example:

- *plugins_root*/uninstPlugin/_jvmForPlugin contains the WebSphere Application Server SDK, Java technology edition used to uninstall the product
- *plugins_root*/uninstPlugin contains the uninstaller program
- *plugins_root*/bin contains the binary plug-ins for all supported Web servers
- *plugins_root*/logs contains log files
- *plugins_root*/properties contains version information
- *plugins_root*/roadmap contains the roadmap for the Plug-ins installation wizard

After configuring a Web server and creating a Web server definition, you can deploy applications and serve them through the Web server. See Fast paths for WebSphere Application Server to get started deploying applications.

responsefile.txt

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. Use the following command to use a copy of the options file named myresponsefile.txt for a silent installation:

```
install -options "myresponsefile.txt" -silent
```

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" -silent
```

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:

   Linux

   ```
   install -options /tmp/plugins/myresponsefile.txt -silent
   ```

   Windows On non-Windows operating systems, silent installations run in a synchronous process. The process does not return until the silent installation finishes. For a silent installation to run synchronously on Windows, issue the first of the following commands:

   - **Synchronous processing:** START /WAIT install.exe -options ″C:\temp\myresponsefile.txt″ -silent
   - **Asynchronous processing:** install -options ″C:\temp\plugins\myresponsefile.txt″ -silent
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 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.

▶ Solaris The GSKit 4 installation record is written after the GSKIT 4: 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.

Information that ISMP can log when it cannot start the Plug-ins installation wizard

Certain events can prevent the installer from starting the Installation wizard. Such an event is not enough disk space to launch the Installation wizard, for example. If your installation fails and there is no information in the installation logs, use the -log parameter to record entries for events that cause the installer program to fail to start the installation wizard. The syntax of the install command for logging such events is:

```
install  -options fully_qualified_options_response_file_name
         -silent
         -log # !fully_qualified_log_file_name  @ALL
```

- AIX

  ```
  install -options "/usr/IBM/WebSphere/silentFiles/myresponsefile.txt"
          -silent -log # !/usr/IBM/WebSphere/myOptionFiles/log.txt  @ALL
  ```

- Linux HP-UX Solaris

  ```
  install -options "/opt/IBM/WebSphere/silentFiles/myresponsefile.txt"
          -silent -log # !/opt/IBM/WebSphere/myOptionFiles/log.txt  @ALL
  ```

- Windows

  ```
  install.exe -options "C:\IBM\WebSphere\silentFiles\myresponsefile.txt"
              -silent -log # !C:\IBM\WebSphere\silentFiles\log.txt  @ALL
  ```

Verify or troubleshoot the installation if the *plugins_root*/logs/install/log.txt file does not contain a record of any problems, but problems exist.

If the error happens early in the installation, look for the logs in the system temporary directory. The installation program copies the logs from the system temporary directory to the logs directory at the end of the installation.

See Troubleshooting installation and Installation component troubleshooting tips for more information.

Usage notes

- The file is not a read-only file.
- Edit this file directly with your flat file editor of choice, such as WordPad on a Windows platform.
- 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.

Example responsefile.txt file

Edit the version of the file that ships with your WebSphere Application Server product. The following example is not guaranteed to be an accurate representation of the file that ships with the product.

```
################################################################################
#
# A common use of an options file is to run the wizard in silent mode. This lets
# the options file author specify wizard settings without having to run the
# wizard in graphical or console mode. To use this options file for silent mode
# execution, use the following command line arguments when running the wizard:
#
#   install -options "responsefile.txt" -silent
#
################################################################################
# Note: Synchronous Silent Installs
# --------------------------------
# On non-Windows operating systems, silent installs will execute in a synchronous process.
# The process will not return until the silent install has completed.
#
# For the silent install to run synchronously on Windows, you must run the
# executable in the following way:
#
#    START /WAIT  install.exe -options "response_local.txt" -silent
#
#
################################################################################

################################################################################
#
# TROUBLE SHOOTING TIP
#
# If no signs of an install are visible, reference the log file (niflogs/log.txt)
# in the user's temporary directory for signs of cause.
#

################################################################################
# Note: Default Web server Plugins install locations
# ---------------------------------------
#
#  Below are the default install locations for each supported operating system.
#
#  AIX:  "/usr/IBM/WebSphere/Plugins"
#
#  HP-UX, Linux, and Solaris: "/opt/IBM/WebSphere/Plugins"
#
#  Windows: "C:\Program Files\IBM\WebSphere\Plugins"
#
#  Note: On unix operating systems, spaces in the install location is not supported.
################################################################################

#################################################################################
#
# Allow non-root/non-admin install
#
# Valid Options : true   Allows install for non-root/non-admin users.
#    false  Disallows install for non-root/non-admin users. Install will not occur.
#
# If no install occurs, this will be logged to a log file (niflogs/log.txt)
# in the user's temporary directory.
#
# By changing this value in this response file to "true", you will allow the
# install to continue
# for non-root/non-admin users.
#
#

-OPT allowNonRootSilentInstall="false"
```

```
###################################################################################
#
# License Acceptance
#
# Valid Options : true   Accepts the license. Will install the product.
#    false  Declines the license. Install will not occur.
#
# If no install occurs, this will be logged to a temporary log file (
# temporaryPluginInstallLog.txt) in the user's temporary directory.
#
# By changing this value in this response file to "true", you agree that you
# have reviewed and agree to the terms of the IBM International Program License
# Agreement accompanying this program, which is located at CD_ROOT\plugins\lafiles.
# If you do not agree to these terms, do not change the value or otherwise download,
# install, copy, access, or use the program and promptly return the program and proof
# of entitlement to the party from whom you acquired it to obtain a refund of the
# amount you paid.
#
#

-OPT silentInstallLicenseAcceptance="false"

################################################################################
# Operating System Prerequisite Checking
#
# If you want to disable operating system prerequisite checking, uncomment
# the following line. This will notify the installer that you are disabling
# operating system prerequisite checking.
#
-OPT disableOSPrereqChecking="true"

################################################################################
#
# install Type
#
# Valid Options : local   WebSphere Application Server and Web server on same machine
#                 remote  WebSphere Applicaiton Server and Web server on seperate machines.
#

-OPT installType="local"


################################################################################
#
# pluginDestination
#
# Web server Plugins for IBM WebSphere Application Server V6.1 install location
#

-OPT installLocation="C:\Program Files\IBM\WebSphere\Plugins"


################################################################################
#
# WAS V6.1 Existing Location
#
# Valid Options : Existing WebSphere Application Server Version 6.1 install home directory.
#
# Note  : This option is valid for local install type.
#         The install will use the directory entered below.
#

-OPT wasExistingLocation="C:\Program Files\IBM\WebSphere\AppServer"
```

```
################################################################################
#
# Web server to configure
#
# Valid options
#  : none          Install binaries only.  No Web server configuration.
#  : ihs           IBM HTTP Server V6 or V6.1
#  : apache        Apache Web Server V2
#  : domino5       Lotus Domino Web Server V5 (supported on HP-UX)
#  : domino6       Lotus Domino Web Server V6 or V6.5 (not supported on HP-UX)
#  : sunone        Sun ONE Web Server 6.0 or Sun Java System Web Server V6.1
#  : iis5          Microsoft Internet Information Services V5 (supported on Windows)
#  : iis6          Microsoft Internet Information Services V6 (supported on Windows)
#
# Note  : Specify only one Web server to configure.
#

-OPT webServerSelected="domino6"


################################################################################
#
# IHS-specific Administrator settings
#
################################################################################
#
# HTTP Administration Port number
#
# Specify the HTTP Administration Port number (only specify value when configuring IHS)
#

-OPT ihsAdminPort=""


################################################################################
#
# IHS Administrator User ID for IHS Administration server
#
# Specify the IHS Administrator server userid. (only specify value when configuring IHS)
#

-OPT ihsAdminUserID=""



################################################################################
#
# Web server Configuration File 1
#
# Valid options for Web server configuration file 1
#
#  ihs   : httpd.conf
#  apache          : httpd.conf
#  domino5  : Notes.jar
#  domino6  : Notes.jar
#  sunone   : obj.conf
#
#  Note : File must exist
#

-OPT webServerConfigFile1="C:\Program Files\IBM HTTP Server\conf\httpd.conf"


################################################################################
#
# Web server Configuration File 2
#
```

```
# Valid options for Web server configuration file 2
#
#  domino5 : names.nsf
#  domino6 : names.nsf
#  sunone  : magnus.conf
#
# Note : File must exist
#

-OPT webServerConfigFile2=""


################################################################################
#
# Web server port number
#
# Specify the Web server port for the Web server selected to be configured.
#

-OPT webServerPortNumber="80"


################################################################################
#
# Domino 6 User ID
#
# Specify the Domino 6 User ID. (only specify value when configuring Domino 6 on
# Linux, Solaris, and AIX platforms)
#
# Note : Domino 6 is supported on Windows, but this value is not required on Windows.
#

-OPT domino6UserID="notes"


################################################################################
#
# Web server Definition Name
#
# A web server definition allows for Web server administration through the WebSphere
# admin console.
#
#  Note : No spaces are allowed in the Web server definition name.
#

-OPT webServerDefinition="webserver1"


################################################################################
#
# plugin-cfg.xml File Location
#
# This file will be generated by the plugin installer.
#
# Valid options:
# ""      : leaving the string empty will result in installer generating the plugin-cfg.xml
#       file location at install time and configuring Web server to use this location.
#       This is the recommended option.
#
# "<file_location>" : User may enter an existing file location.  Web server will be
#       configured to use this existing plugin-cfg.xml file location.
#       If file is specified, it must exist, otherwise install will not proceed.
#

-OPT pluginCfgXmlLocation=""
```

```
################################################################################
#
# WebSphere Application Server Machine HostName
#
# remote install type  : enter the hostname of the WebSphere Application Server machine.
# local install type  : hostname of machine being installed to will be used.
#

-OPT wasMachineHostName=""


################################################################################
#
#  Advanced User Options available in silent installs only
#
#  Map all the existing deployed applications to the Web server.
#
#  Valid Options
#   true  :  Web server Definition is mapped as a target to the existing deployed applications
#               such as snoop and hitcount (Recommended)
#   false :  No applications are mapped to the Web server definition.
#
#  Note : If not set to a valid option of true or false, the installer will set to true
# and continue install.
#

-OPT mapWebserverToApplications="true"


################################################################################
#
#  Web server Hostname
#
#  In advanced scenarios where a user has multiple Web server hostnames on a machine,
#  set the entry below to the Web server hostname used to configure.
#
#  Valid Options :
#     ""    :  Install will resolve to hostname detected on machine (Recommended)
#   "<HOSTNAME>" :  Where <HOSTNAME> is a Web server hostname on the machine.
#

-OPT webServerHostName=""


################################################################################
#
# WAS Profile Name
#
# Specify the name of the WAS Profile to be configured. This option is only valid
# in local install scenarios.
#
# Valid options:
# "<WAS_profile_name>"  : User must enter the name of an existing WAS profile
#                              leaving the string empty will result in the installer
#                              using the default profile
#                              example.  profileName="AppSrv01"
#

-OPT profileName=""
```

Editing Web server configuration files

The Plug-ins installation wizard automatically configures supported Web servers during the installation of the binary plug-in modules. Use this topic to understand how the wizard configures the Web server configuration files.

If you must change a configuration for some reason, you can run the Plug-ins installation wizard again to reconfigure the Web server, or you can edit the files. The recommended approach is to always use the Plug-ins installation wizard to configure the Web server configuration file. However, sometimes you must edit the files. An example is when you are installing a Web server definition for a non-default profile.

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

This task shows you how to edit the Web server configuration files. Select one of the following links that identifies the Web server that you must reconfigure.

- Configure Apache HTTP Server 2.0. See “Configuring Apache HTTP Server V2.0.”
- Configure Domino Web Server Version 5 and Version 6.x. See “Configuring Lotus Domino” on page 105.
- Configure IBM HTTP Server Version 6.0. See “Configuring IBM HTTP Server Version” on page 109.
- Configure IBM HTTP Server Version 2.0. See “Configuring IBM HTTP Server powered by Apache 2.0” on page 107.
- Configure Microsoft Internet Information Services (IIS). See “Configuring Microsoft Internet Information Services (IIS)” on page 110.
- Configure Sun Java System Web Server (formerly Sun ONE and iPlanet). See “Configuring the Sun Java System Web Server” on page 113.

You can use the Plug-ins installation wizard to automatically configure supported Web servers. You can also configure a Web servers by editing its configuration.

To continue preparing your e-business network, return to “Installing Web server plug-ins” on page 53.

Configuring Apache HTTP Server V2.0

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

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 2.0” on page 107.

When you install the Web server plug-ins for WebSphere Application Server, as described in “Installing Web server plug-ins” on page 53, 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” on page 102 describe configuring other supported Web servers.

Important:

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

Testing verified that the WebSphere Application Server plug-in for Apache HTTP Server, Version 2.0 works with Apache HTTP Server Version 2.0.48, the latest version of Apache 2.0. The plug-in was tested with the threaded worker multi-processing module (MPM) on all platforms except Windows. The plug-in was tested with the default threaded MPM on Windows.

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

Compatibility Statement The plug-in works with versions of the Apache HTTP Server that claim full binary compatibility with Apache 2.0.42 and later, which are built with compilers and compiler options that are compatible with those used to build the plug-in.

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.

- AIX Configure entries in the httpd.conf file.

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

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

  Example:

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

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

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

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

  Example:

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

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

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

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

  Example:

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

- Windows Configure entries in the httpd.conf file.

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

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

Example:

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

This procedure results in reconfiguring the Apache 2.0 Web server.

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.

When you install the Web server plug-ins for WebSphere Application Server on a non-iSeries system, as described in "Installing Web server plug-ins" on page 53, the Plug-ins installation wizard configures the Web server.

This topic describes how to manually configure the Domino V6 Web server.

Other procedures in "Editing Web server configuration files" on page 102 describe configuring other supported Web servers.

Use the following procedure to enable the Web server plug-in to work with Lotus Domino.

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 WebSphere Application Server, Version 6.
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/QWAS61.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**

Set the WAS_PLUGIN_CONFIG_FILE environment variable to the location of the plug-in configuration file using one of the following paths:

If the type of installation is:	Then use this command to set the environment variable:
Remote	WAS_PLUGIN_CONFIG_FILE=/opt/IBM/WebSphere/Plugins/config/webserver1/plugin-cfg.xml
Local stand-alone	WAS_PLUGIN_CONFIG_FILE=*profile_root*/config/cells/sa_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. **Setting the path to the plug-in configuration file on a Windows system**

Add the appropriate statement to your *lotus_domino_root*\notes.ini file:

Example:

```
WebSpherePluginCfg="C:\Program Files\IBM\WebSphere\Plugins\config\webserver1\plugin-cfg.xml"
```

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
```

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

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 tips

Lotus Domino Server is one of the Web servers that WebSphere Application Server supports. This topic describes configuration file locations, a Solaris configuration tip, and how to avoid a DSAPI filter-loading error when starting the Lotus Domino Web server as a non-root user on Linux and UNIX operating systems.

Lotus Domino Server file locations

Lotus Domino server file locations are:

- AIX
  - /usr/lotus/notes/50091/ibmpow/Notes.jar
  - /usr/notesdata/names.nsf
- Solaris
  - /opt/lotus/notes/5080/sunspa/Notes.jar
  - /opt/notesdata/names.nsf

- Windows
  - c:\Program Files\lotus\notes\Notes.jar
  - c:\Program Files\lotus\notes\data\names.nsf

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

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

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

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

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

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

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

On a Linux or UNIX operating system, if the Lotus Domino Web server starts using a non-root user, you are likely to generate a DSAPI filter-loading error when the Lotus Domino Server starts:

```
Error loading DSAPI filter.

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

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

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

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

Configuring IBM HTTP Server powered by Apache 2.0

This topic describes how to change configuration settings for IBM HTTP Server powered by Apache 2.0.

When you install the Web server plug-ins for WebSphere Application Server, as described in “Installing Web server plug-ins” on page 53, the Plug-ins installation wizard configures the Web server on distributed platforms. This topic describes how to configure the IBM HTTP Server, V2.x. Other procedures in “Editing Web server configuration files” on page 102 describe configuring other supported Web servers.

Perform the step that configures IBM HTTP Server 6.0 version 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 Web server definition name in the following examples is webserver1.

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

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

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

  Example:

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

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

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

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

  Example:

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

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

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

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

  Example:

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

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

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

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

  Example:

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

This procedure results in editing and reconfiguring IBM HTTP Server powered by Apache 2.0.

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 IBM HTTP Server Version

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

When you install the Web server plug-ins for WebSphere Application Server, as described in “Installing Web server plug-ins” on page 53, the Plug-ins installation wizard configures the Web server. This topic describes how to configure IBM HTTP Server, V6.x. Other procedures in “Editing Web server configuration files” on page 102 describe configuring other supported Web servers.

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

- AIX Configure entries in the httpd.conf file.

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

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

  Example:

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

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

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

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

  Example:

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

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

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

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

Example:

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

- Windows Configure entries in the httpd.conf file.

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

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

  Example:

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

This procedure results in editing and reconfiguring IBM HTTP Server.

If the IBM HTTP Server V1.3.x directive, LoadModule ibm_app_server_http_module, is present in an IBM HTTP Server V6.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 Microsoft Internet Information Services (IIS)

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

When you install the Web server plug-ins for WebSphere Application Server, as described in “Installing Web server plug-ins” on page 53, the Plug-ins installation wizard configures the Web server. This topic describes how to configure the Internet Information Services (IIS) Web server. Other procedures in “Editing Web server configuration files” on page 102 describe configuring other supported Web servers.

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

Use the following procedure to manually reproduce how the Installation wizard configures the Microsoft Internet Information Services Web server.

- Configure IIS V5.0.
  1. Start the IIS application and create a new virtual directory for the Web site instance that you intend to work with WebSphere Application Server. These instructions assume that you are using the Default Web Site.
  2. Expand the tree on the left until you see Default Web Site.

     Right-click **Default Web Site**, then click **New > Virtual Directory** to create the directory with a default installation.
  3. Type **sePlugins** in the Alias to be used to Access Virtual Directory field.
  4. Browse to the *plugins_root*\bin directory in the Enter the physical path of the directory containing the content you want to publish field.
  5. Select the appropriate **Execute** check box (such as ISAPI applications or CGI) in the What access permissions do you want to set for this directory field.

6. Click **Next** to add the sePlugins virtual directory to your default Web site.
7. Click **Finish**.
8. Right-click **Default Web Site** in the navigation tree and click **Properties**.

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

   In the Properties dialog, perform the following steps:

   a. Click the **Internet Information Services** tab.

   b. Click **WWW Service** in the Master properties window.

   c. Click **Edit** to open the WWW Service master properties window.

   d. Click **ISAPI Filters > Add** to open the Filter properties window.

   e. Type **iisWASPlugin** in the Filter Name field.

   f. Click **Browse** in the Executable field.

   g. Browse to the *plugins_root*\bin directory.

   h. Click the **iisWASPlugin_http.dll** file.

   i. Click **OK** until all the open windows close.

- Configure IIS V6.0.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Example:

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

12. Configure the Web server to run WebSphere Application Server extensions:

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

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

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

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

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

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

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

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

To enable worker process in isolation mode:

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

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

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

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

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

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

4. Assign an application pool for each Web site.

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

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

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

d. Click **OK**.

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

5. Start the IIS service and start each Web site.

This procedure results in reconfiguring the Internet Information Services (IIS) Web server.

You can now install applications on the configured Web server. See the Applications section of the information center for more information.

Configuring the Sun Java System Web Server

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

When you install the Web server plug-ins for WebSphere Application Server, as described in “Installing Web server plug-ins” on page 53, the Plug-ins installation wizard configures the Web server. This topic describes how to configure the Sun Java System Web Server if you must change something in the existing configuration. Other procedures in “Editing Web server configuration files” on page 102 describe configuring other supported Web servers.

Configure the Sun ONE Web Server 6.0 or Sun Java System Web Server, V6.1 and later, on a Linux, UNIX, or Windows platform.

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

1. Configure entries in the obj.conf configuration file and in the magnus.conf configuration file for Version 6.0 and later of Sun Java System Web Server.
   a. Add two directives to the obj.conf file after the <Object name=default> tag:

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

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

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

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

      - ▶ Linux

        Example:

        ```
        Init fn="load-modules"
             funcs="as_init,as_handler,as_term"
             shlib="/opt/IBM/WebSphere/Plugins/bin/libns41_http.so"
        Init fn="as_init"
             bootstrap.properties="/opt/IBM/WebSphere/Plugins/config/webserver1/plugin-cfg.xml"
        ```

      - ▶ Windows

        Example:

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

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

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

3. Disable the feature of Sun Java System Web Server V6.1 that supports servlets and JavaServer Pages files by default. Disable this feature so that the WebSphere Application Server plug-in can handle the requests.

   Perform the following steps to disable the feature:

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

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

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

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

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

This procedure results in editing and reconfiguring the Sun Java System Web Server.

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

Installing Web server plug-in maintenance

Use the Update Installer program to install maintenance packages for Web server plug-ins. Stop the Web server before installing the maintenance package to allow the Update Installer to update the configuration files in the Web server.

Before installing a maintenance package for a Web server plug-in, you must first install the Web server plug-in from the WebSphere Application Server installation image. See “Installing Web server plug-ins” on page 53 for more information.

This topic describes installing a maintenance package for a Web server plug-in for WebSphere Application Server. WebSphere Application Server products supply a unique binary plug-in module for each supported Web server, which might require service updates that are included in a maintenance package. The plug-in configuration file that the WebSphere Application Server products supply is associated with the binary module and might also require maintenance. The Update Installer wizard installs any maintenance packages that you download from the Support site.

1. Go to the Recommended Updates for WebSphere Application Server Web page to determine which maintenance packages are available for the Web server plug-ins. This site is located at http://www-1.ibm.com/support/docview.wss?rs=180&uid=swg27004980

   Install maintenance packages for Web server plug-ins in this order:

   a. Refresh packs

   b. Fix packs

   c. Interim fixes

2. Download the appropriate maintenance packages to a temporary location.

   For example, click the ftp link for the download package labeled, Intel Plug-ins, to download the maintenance package.

3. Follow the directions for using the Update Installer wizard to install the maintenance packages.

   Verify that you stop the Web server process before installing a maintenance package for the Web server plug-ins.

You can install a maintenance package for a Web server plug-in by following the procedures described in this topic.

After installing maintenance packages, test the applications that the Web server provides to verify that the Web server is functioning correctly.

Uninstalling the Web server plug-ins for WebSphere Application Server

You can uninstall Web server plug-ins for WebSphere Application Server without uninstalling the supported Web server or the application server.

Use the following instructions to uninstall a web server plug-in.

This topic describes uninstalling the Web server plug-ins for WebSphere Application Server.

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.

After uninstalling the Web server plug-ins for WebSphere Application Server, you can reinstall them using any of the installation procedures described in “Installing Web server plug-ins” on page 53

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.
4. Issue the uninstall command.

   The Uninstall wizard displays a welcome panel.

   The next few steps in this procedure describe using the wizard interactively. You can also issue the uninstall command with a silent parameter to use the wizard without the graphical user interface.

   ```
   uninstall -silent
   ```
5. Click **Next** on the Welcome panel.

   The Uninstaller wizard displays a confirmation panel for you to confirm what is to be uninstalled. Each feature that displays in the panel is a separate Web server installation. Each Web server installation is configured to use one of the binary plug-in modules that is being deleted.
6. Click **Next** on the confirmation panel to begin uninstalling the plug-ins.

   The Uninstaller wizard displays a summary panel that provides status.
7. Click **Finish** to close the Uninstaller wizard.

   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.
8. ▶ Linux Uninstall the IBM Global Services Kit (GSKit) on Linux and UNIX systems.

   The Plug-ins installation wizard does not uninstall GSKit 7 because there is no registry describing products that are enrolled to use the GSKit. Without this critical information, uninstalling the GSKit product might affect other products that use GSKit.

   You can uninstall GSKit once it is no longer in use on your systems.

   ▶ Solaris On Solaris systems, you must also uninstall GSKit 4 in addition to GSKit 7.
9. ▶ Windows Uninstall GSKit on Windows systems.

   The Plug-ins uninstaller program unregisters the GSKit. The registry key is **HKEY_LOCAL_MACHINE > SOFTWARE > IBM > GSK7 > REGAPPS > WASPlugins60_*unique_key***.

If the registry key WASPlugins60_*unique_key* is the last remaining key in the GSKit registry entry, the wizard also uninstalls the GSK7 product. If another product, such as IBM HTTP Server, is registered to use GSKit, the wizard does not uninstall the GSKit. The wizard always unregisters the registry key for the Web server plug-ins for WebSphere Application Server, which is WASPlugins60_*unique_key*.

The GSKit uninstall log is the *plugins_root*/log/uninstall/uninstallGSKit.log file.

Possible problem after using the Update Installer to install a maintenance package

When you apply a WebSphere Application Server plug-in refresh pack, the plug-in installer incorrectly adds an additional registry key that prevents the GSKit from uninstalling.

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

10. Delete files from the system temporary directory.

    Delete the following files:

    - Install temporary log : temporaryPluginInstallLog.txt
    - Uninstall temporary log : temporaryPluginUnInstallLog.txt

11. Delete the Web server definition in a stand-alone 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
    ```

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.

Important: 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.

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 305.

See “Installing Web server plug-ins” on page 53 for information about installation scenarios for reinstalling Web server plug-ins.

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 use this manual procedure for uninstalling Web server plug-ins for WebSphere Application Server when the uninstaller program is not available for some reason.

This topic describes uninstalling the Web server plug-ins for WebSphere Application Server.

After uninstalling the Web server plug-ins for WebSphere Application Server, you can reinstall them using any of the installation procedures that are described in “Installing Web server plug-ins” on page 53

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. Manually delete GSKit.

   Use the following procedure to verify that no other products are registered in GSKit before running the isuninst command.

   a. Click **Start > Run** and run the regedit command to edit the registry.

   b. Change directories to HKEY_LOCAL_MACHINE\SOFTWARE\IBM\GSK*x*\CurrentVersion\REGAPP, where *x* is the version of GSKit, such as GSK7.

   c. Look for other products that are registered within GSKit.

      Look for entries other than the default entry and for products other than IBM HTTP Server and Web Server plug-ins for WebSphere Application Server. If the following entries are the only ones present, you can delete GSKit as described in the following step:

      - (Default)
      - WASPLUGIN60
      - IHS60

   d. Run the following command to invoke the GSKit Uninstaller program:

      ```
      isuninst -f"gskit_root\gsk7BUI.isu"
      ```

      This command removes GSKit, regardless of whether or not other applications are registered as using GSKit.
4. Reconfigure any Web servers that were configured to use the binary plug-ins that you deleted.

   See “Editing Web server configuration files” on page 102.
5. Delete the following registry keys using the platform-specific steps described in Uninstalling manually.

Operating system	Registry keys
AIX, Linux and Windows	WSPAA60, WSPAA60DefineglobalconstantsComponent, WSPAA60DefinelocalvariablesComponent, WSPAA60LicensingComponent, WSPAA60Webserverplugins, WSPAA60WebserverpluginsComponent, WSPAA60AddBytes, WSPAA60gskit, WSPAA60gskitComponent
HP-UX	WSPAA60, WSPAA60DGCC, WSPAA60DLVC, WSPAA60LC, WSPAA60WSPC, WSPAA60AddBytesHS, WSPAA60gskitHP, WSPAA60gskitHPC, WSPAA60jdkHP
Solaris	WSPAA60, WSPAA60AC, WSPAA60BC, WSPAA60CC, WSPAA60DC, WSPAA60FC, WSPAA60FB, WSPAA60GC, WSPAA60HC

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. Reinstall the Web server plug-ins for WebSphere Application Server and reconfigure the Web servers to restore their functionality.

See “Installing Web server plug-ins” on page 53 for information about other installation scenarios for reinstalling Web server plug-ins.

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.

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

The Plug-ins installation wizard creates a Web server definition on the Application Server system, either directly when they are on the same machine, or by a script for remote scenarios.

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

The plugin-cfg.xml file can be overwritten by the deployment manager sync operation, the GenPluginCfg script or any other method that regenerates the file. If you make changes to the plugin-cfg.xml file, and want to keep those changes, it is recommended that you create a copy of the file in a separate location. Make your manual updates each time the file is automatically refreshed by another process.

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.

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

   ```
   ./stopServer.sh server1
   ```

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

   ```
   Server server1 stop completed.
   ```

   To start the application server, issue the following command:

   ```
   ./startServer.sh server1
   ```

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

   ```
   Server server1 open for e-business; process id is 1719
   ```
4. Edit the plugin-cfg.xml file to include the following entries:

   ```
   <VirtualHostGroup Name="admin_host">
          <VirtualHost Name="*:9060"/>
          <VirtualHost Name="*:80"/>
          <VirtualHost Name="*:9043"/>
      </VirtualHostGroup>
      ...
      ...
      ...
      <ServerCluster Name="server1_SERVER1HOSTserver1_Cluster">
   ```

```
        <Server LoadBalanceWeight="1" Name="SERVER1HOSTserver1_dmgr">
            <Transport Hostname="SERVER1HOST" Port="9060" Protocol="http"/>
        </Server>

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

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

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

The *port* variable is your HTTP server port.

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

Web server plug-in properties settings

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 > 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 an IBM® HTTP Server V6.1 for your Web server, WebSphere® Application Server can automatically propagate the plug-in configuration file for you to remote machines provided there is a working HTTP transport mechanism to propagate the file.

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

Data type	String
Default	plugin-cfg.xml

Automatically generate plug-in configuration file

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

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

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

- The WebSphere Application Server administrator defines new Web server.
- An application is deployed to an Application Server.
- An application is uninstalled.

- A virtual host definition is updated and saved.

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

Important: When the plug-in configuration file is generated, it does not include admin_host on the list of virtual hosts. The Information Center article ″Allowing Web servers to access the administrative console″ describes how to add admin_host to the list of virtual hosts.

Automatically propagate plug-in configuration file

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

- This field must be checked.
- The plug-in configuration service must be enabled
- A WebSphere Application Server node agent must be on the node that hosts the Web server associated with the changed plug-in configuration file.

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 Web server and its administration server is running.

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

For distributed platforms, you can check the related messages in the deployment manager SystemOut.log file to verify that the automatic propagation successfully completed.

Plug-in key store file name

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

- 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.

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

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.

Data type	String
Default	Error

Web server plug-in request and response optimization properties settings

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 > 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 URI's for a sendRedirect.

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

Specify:

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

The default is `webserverPort`.

Web server plug-in caching properties settings

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

To view this administrative console page, click **Servers > 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 settings

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 > Web Servers >** *Web_server_name* **Plug-in Properties > Plug-in** *server_cluster_name* **Properties**.

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 settings

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 **Application Servers >** *server_name* > **Administration Services** > **Web server plug-in configuration service** to view this administrative console page.

If you are using the deployment manager, click **System Administration** > **Deployment manager > Administration Services > Web server plug-in configuration service**.

For i5/OS and distributed platforms, the deployment manager SystemOut log contains the status of the automatic plug-in generation and propagation.

Enable automated Web server configuration processing

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

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

The plug-in configuration file does not regenerate when:

- A cluster member is added to a cluster
- TCP channel settings are updated for an application server

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 **Application Servers >** *server_name*, and then under Additional Properties, 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

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. If `Set Timeout` is selected, 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 you select `No Timeout`, 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	No Timeout

Connect timeout

Specifies whether or not there is a limited amount of time the application server will maintain a connection with the Web server.

You can select either **No timeout** or **Set timeout**. If you select **Set timeout** you, must specify, in seconds, the length of time a connection with the Web server is to be maintained.

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	0

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.

You can select either **No limit** or **Set limit**. If you select **Set limit** 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	-1

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 7. Web server plug-in configuration properties

Administrative console panel	Field name	Configuration property name
In the administrative console, click **Servers** > **Web Servers** > *Web_server_name* > **Plug-in properties**	Refresh configuration interval	RefreshInterval
In the administrative console, click **Servers** > **Web Servers** > *Web_server_name* > **Plug-in properties**	Plug-in log file name	Log->name
In the administrative console, click **Servers** > **Web Servers** > *Web_server_name* > **Plug-in properties**	Plug-in logging	Log->LogLevel
In the administrative console, click **Servers** > **Web Servers** > *Web_server_name* > **Plug-in properties**	Ignore DNS failures during Web server startup	IgnoreDNSFailures
In the administrative console, click **Servers** > **Web Servers** > *Web_server_name* > **Plug-in properties**	KeyringLocation	Keyring
In the administrative console, click **Servers** > **Web Servers** > *Web_server_name* > **Plug-in properties**	StashfileLocation	Stashfile
In the administrative console, click **Servers** > **Web Servers** > *Web_server_name* > **Plug-in properties** > **Custom properties** > **New**	FIPSEnable	FIPSEnable

Table 7. Web server plug-in configuration properties (continued)

In the administrative console, click **Servers > Web Servers >** *Web_server_name* **> Plug-in properties > Request routing**	Load balancing option	LoadBalance
In the administrative console, click **Servers > Web Servers >** *Web_server_name* **> Plug-in properties > Request Routing**	Clone separator change	CloneSeparatorChange
In the administrative console, click **Servers > Web Servers >** *Web_server_name* **> Plug-in properties > Request Routing**	Retry interval	RetryInterval
In the administrative console, click **Servers >** *Web server_name* **> Plug-in properties > Request routing**	Maximum size of request content	PostSizeLimit
In the administrative console, click **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 > Web Servers >** *Web_server_name* **> Plug-in properties > Request routing**	Remove special headers	RemoveSpecialHeaders
In the administrative console, click **Application Servers >** *server_name* **> Web server plug-in properties**	Server role	PrimaryServers and BackupServers list
In the administrative console, click **Application Servers >** *server_name* **> Web server plug-in properties**	Connect timeout	Server ConnectTimeout
In the administrative console, click **Application Servers >** *server_name* **> Web server plug-in properties**	The read and write timeouts for all the connections to the application server	ServerIOTimeout
In the administrative console, click **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 **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 **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 > 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

Table 7. Web server plug-in configuration properties (continued)

In the administrative console, click **Servers** > **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** > **Web Servers** > *Web_server_name* > **Plug-in properties** > **Request and Response**	Virtual host matching	VHostMatchingCompat
In the administrative console, click **Servers** > **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** > **Web Servers** > *Web_server_name* > **Plug-in properties** > **Request and Response**	Accept content for all requests	AcceptAllContent
In the administrative console, click **Servers** > **Web Servers** > *Web_server_name* > **Plug-in properties** > **Request and Response**	Chunk HTTP response to the client	ChunkedResponse
In the administrative console, click **Servers** > **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** > **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** > **Web Servers** > *Web_server_name* > **Plug-in properties** > **Caching**	Maximum cache size	ESIMaxCacheSize
In the administrative console, click **Servers** > **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 HTTP inbound channel's **Maximum persistent requests** property.)
- 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.

Important: 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.

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.

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.

1. Change the directory to the installation root of the Web server. For example, this is /opt/IBM/HTTPServer on a Solaris machine.
2. Find the subdirectory that contains the executable. The executable is:
   - ▶ Windows apache.exe
   - ▶ Linux apachectl
3. Issue the command.

   ▶ Windows On a Windows platform, issue:

   ```
   apache.exe -V
   ```

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

```
Server version: IBM_HTTP_Server/2.0.47 Apache/2.0.47
Server built: November 26 2005 20:38:36
Server's Module Magic Number: 20020903:4.
```

In this example an IBM HTTP Server powered by Apache 2.0.43 is installed.

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.

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.

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.

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
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

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 the threading model instead of the process model, and only one process is started.

The IBM HTTP Server V6.1 follows the threading model. To prevent the IBM HTTP Server from starting more than one process, change the following properties in the Web server configuration file (httpd.conf) to the indicated values:

```
ServerLimit          1
ThreadLimit          4000
StartServers         1
MaxClients           1024
MinSpareThreads      1
MaxSpareThreads      1024
ThreadsPerChild      1024
MaxRequestsPerChild  0
```

Improving performance in a high stress environment ▶ Windows

If you use the default settings for a Microsoft Windows operating system, you might encounter Web server plug-in performance problems if you are running in a high stress environment. To avoid these problems, consider tuning the TCP/IP setting for this operating system. Two of the keys setting to tune are TcpTimedWaitDelay and MaxUserPort.

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

1. Locate in the Windows Registry:

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

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

To tune the MaxUserPort setting:

1. Locate in the Windows Registry:

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

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

See the Microsoft 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.

Starting with Version V6.0.1, the plug-in configuration file is generated in ASCII format (ISO-98859-1). (Previously the configuration file was generated in EBCDIC format.) If you need to edit this file, issue the following command to convert the file to EBCDIC format:

```
> iconv  -f ISO8859-1 -t IBM-1047 plugin-cfg.xml.ASCII > plugin-cfg.xml.EBCDIC
```

Edit the file, and then issue the following command to convert it back to ASCII format:

```
> iconv  -f IBM-1047 -t ISO8859-1  plugin-cfg.xml.EBCDIC > plugin-cfg.xml.ASCII
```

CAUTION:
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:

- webserverPort if the port number from the host header of the HTTP request coming in is to be used.
- hostHeader 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.

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 algorithm for this attribute decrements all weights within the server cluster until all weights reach zero. After the weight specified for a particular server reaches zero, no more requests are routed to that server until all servers in the cluster have a weight of zero. After all servers reach 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 2 minutes depending on the platform) and allows the plug-in to mark the server *unavailable*. A value of 0 causes the plug-in to perform a blocking connect. A value greater than 0 specifies the number of seconds you want the plug-in to wait for a successful connection. If a connection does not occur after that time interval, the plug-in marks the server *unavailable* and fails over to one of the other servers defined in the cluster.

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.)

This attribute is not necessary on the z/OS platform. The z/OS controller working in conjunction with WLM, handles new connections dynamically.

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.

Important: Solaris The ServerIOTimeout attribute is ignored for a plug-in running on a Solaris platform.

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.

Important: 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.

Setting up a local Web server

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

Important: Non-IBM HTTP Server Web servers must reside on a managed node to facilitate plug-in administration functions and generation and propagation of the `plugin-cfg.xml` file.

You can define a locally-installed Web server on an unmanaged or managed node. If the Web server is defined on an unmanaged node, the administrative functions are handled through the IBM HTTP Server administration server. If the Web server is defined on a managed node, the administrative functions of the Web server are handled through the WebSphere Application Server node agent, which is beneficial.

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

1. Install your WebSphere Application Server product.

   See the *Installing your application serving environment* PDF.
2. Install IBM HTTP Server or another supported Web server.

   See the *Installing your application serving environment* PDF.
3. Install the binary plug-in module using the Plug-ins installation wizard.

   The Web server definition is automatically created and configured during the installation of the plug-ins.

   See the *Installing your application serving environment* PDF.
4. Complete the setup by creating the Web server definition using the WebSphere Application Server administrative console, or run the plug-in configuration script. The creation of this object is exclusive of the Web server installation.

   Select one of the following options:

   - **Using the administrative console.** You must create a Web server definition on an existing application server or unmanaged node.
     a. Click **Servers > Web servers > New** and use the **Create new Web server entry** wizard to create the Web server definition.
     b. Select the appropriate node.
     c. Enter the Web server properties:
        - Type: The Web server vendor type
        - Port: The existing Web server port (default: 80)

- Installation path: The Web server installation path. This field is required for IBM HTTP Server only.
- Service name (Windows operating systems): The Windows operating system service name of the Web server. The default is `IBMHTTPServer6.0`.
- Use secure protocol: Use the HTTPS protocol to communicate with the Web server. The default is `HTTP`.
- Plug-in installation location: The directory path in which the plug-in is installed.

d. Select a template. Select a system template or a user-defined template for the Web server you want to create.

e. Confirmation of Web server creation.

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

  See the *Installing your application serving environment* PDF.

  Refer to “Configuring a Web server and an application server profile on the same machine” on page 82.

  If you install the plug-in, save the plug-in configuration script to run after you create a managed node, otherwise an error occurs. Wait until the script runs successfully and creates the Web server definition on the managed node and node synchronization occurs before starting the Web server.

  Adding the node starts the node agent process. If the node agent is not running, start the node. **Tip**: If you want the Web server to handle requests for an application for multiple managed nodes, install the application on each managed node and on the Web server definition. The script already contains all of the information that you must gather when using the administrative console option.

  See the *Using the administrative clients* PDF for more information.

You can configure non-IBM HTTP Server Web servers as a remote Web server on unmanaged nodes, or as a local Web server on managed nodes. For a non-IBM HTTP Server Web server on a managed node, the following functions are supported:

- Generation of the plug-in configuration, based on WebSphere Application Server repository changes.
- Propagation of the `plugin-cfg.xml` file, based on using node synchronization with the WebSphere Application Server node. Node synchronization is necessary in order to propagate configuration changes to the affected node or nodes.

  The `plugin-cfg.xml` file is propagated to the application server node repository tree from the deployment manager repository.

  Important: The `plugin-cfg.xml` file is propagated to the application server node repository tree. This is not the default `plugin-cfg.xml` file installation location. Changes may have to be made to non-IBM HTTP Server Web server configuration files to update the location of the `plugin-cfg.xml` file that is read by the plug-in module.

  For example, Internet Information Services (IIS) has a file name called `plugin-cfg.loc`, which is read by the IIS plug-in modules to determine the location of the `plugin-cfg.xml` file. The `plugin-cfg.loc` file has to be updated to reflect the `plugin-cfg.xml` file location in the application server node repository.

  Other non-IBM HTTP Server Web servers have different methods to specify the location of the `plugin-cfg.xml` file for the plug-in module. However, in order for propagation to work, update the location to reflect the location in the application server node repository.

The following functions are not supported on a managed node:

- Starting and stopping the Web server.
- Viewing and editing the configuration file.
- Viewing the Web server logs.

For a non-IBM HTTP Server Web Server on an unmanaged node, you can generate plug-in configuration, based on WebSphere Application server repository changes. 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.

Setting up a remote Web server

This topic describes how to 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 one machine and the application server is on another.

Important: Non-IBM HTTP Server Web servers must reside on a managed node to facilitate plug-in administration functions and generation and propagation of the `plugin-cfg.xml` file.

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 a firewall. You must create a remote Web server on an unmanaged node. Unmanaged nodes are nodes without node agents. Since 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. In this case, 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. The following steps will create a Web server definition in the default profile.

1. Install your WebSphere Application Server product.

   See the *Installing your application serving environment* PDF.
2. Install IBM HTTP Server or another supported Web server.

   See the *Installing your application serving environment* PDF.
3. Install the binary plug-in module using the Plug-ins installation wizard.

   The Web server definition is automatically created and configured during the installation of the plug-ins.

   See the *Installing your application serving environment* PDF.
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 **System Administration > Nodes > Add Node** to create an unmanaged node in which to define a Web server in the topology.

        See the *Using the administrative clients* PDF for more information.

     b. Click **Servers > 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) Select appropriate node
        2) 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 `IBMHTTPServer6.0`.
           - **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.
  3) 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`.
  4) Select a Web server template. Select a system template or a user-defined template for the Web server you want to create.
  5) Confirmation of Web server creation.
- **Running the Plug-in configuration script.**

  See the *Installing your application serving environment* PDF.

5. Run the `setupadm` script on Linux and UNIX platforms. 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. Run the `htpasswd` script on Linux, UNIX and Windows platforms. 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 Linux and UNIX 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 on Windows operating systems or Starting the IBM HTTP administration server on Linux and UNIX platforms for instructions.

You can configure non-IBM HTTP Server Web servers as a remote Web server on unmanaged nodes, or as a local Web server on managed nodes. For a non-IBM HTTP Server Web server on a managed node, the following functions are supported:

- Generation of the plug-in configuration, based on WebSphere Application Server repository changes.
- Propagation of the `plugin-cfg.xml` file, based on using node synchronization with the WebSphere Application Server node. Node synchronization is necessary in order to propagate configuration changes to the affected node or nodes.

  The `plugin-cfg.xml` file is propagated to the application server node repository tree from the deployment manager repository.

  Important: The `plugin-cfg.xml` file is propagated to the application server node repository tree. This is not the default `plugin-cfg.xml` file installaion location. Changes may have to be made to non-IBM HTTP Server Web server configuration files to update the location of the `plugin-cfg.xml` file that is read by the plug-in module.

  For example, Internet Information Services (IIS) has a file name called `plugin-cfg.loc`, which is read by the IIS plug-in modules to determine the location of the `plugin-cfg.xml` file. The `plugin-cfg.loc` file has to be updated to reflect the `plugin-cfg.xml` file location in the application server node repository.

  Other non-IBM HTTP Server Web servers have different methods to specify the location of the `plugin-cfg.xml` file for the plug-in module. However, in order for propagation to work, update the location to reflect the location in the application server node repository.

The following functions are not supported on a managed node:

- Starting and stopping the Web server.
- Viewing and editing the configuration file.
- Viewing the Web server logs.

For a non-IBM HTTP Server Web Server on an unmanaged node, you can generate plug-in configuration, based on WebSphere Application server repository changes. 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 Web server properties, for example, installation root, port, configuration file paths, and 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.

You have 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 Express or Base 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 or dummy 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 and where a node agent is present. This node can exist in an Express, Base, or deployment manager environment. A dummy 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.

Administration of the Web server

The IBM HTTP Server Web server is administered and managed on the Web server collection panel in the WebSphere Application Server administrative console. You can start and stop IBM HTTP Server from the administrative console.

In WebSphere Application Server V6.1, you can create a Web server on a stand-alone node, enabling the Web server definition created on the stand-alone node to be included in the Deployment Manager's managed node when it is federated. Web servers that are defined on a stand-alone node are managed just as a Web server that is defined on an unmanaged node. An IBM HTTP Server Web server that is defined on a stand-alone node is managed by the IBM HTTP Server administration server. Non-IBM HTTP Server Web servers are not managed because no administrative agent exists to handle administration management.

WebSphere Application Server Express is limited to one Web server on a stand-alone or unmanaged node.

Starting and stopping IBM HTTP Server. Start and stop IBM HTTP Server as follows:

- On a managed node, start and stop IBM HTTP Server using the administrative console.
- On an unmanaged node, start and stop IBM HTTP Server using the IBM HTTP Server administration server.
- On a stand-alone node, which works like an unmanaged node, use the IBM HTTP Server administration server.

The IBM HTTP Server administration server that is running on the same machine and node as IBM HTTP Server must be started with a command line to handle IBM HTTP Server administration requests. The IBM HTTP Server administration server runs on a separate port from IBM HTTP Server and receives requests from the WebSphere Application Server administration server.

Viewing and editing the IBM HTTP Server configuration file. View and edit the IBM HTTP Server `httpd.conf` configuration file as follows:

- On a managed node use the administrative console. Click **Servers > Web servers >** *server_name* > **Configuration file**.
- On an unmanaged node, view and edit the configuration file using the IBM HTTP Server administration server.

Start the IBM HTTP Server administration server that runs on the same machine and node as the IBM HTTP Server Web server using a command line to manage IBM HTTP Server administration requests. The IBM HTTP Server administration server runs on a separate port from the IBM HTTP Server Web server and receives requests from the WebSphere Application Server administrative server.

Editing the Web server type

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

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** > **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**.

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** > **Web servers**.

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

Button	Resulting Action
Generate Plug-in	When the plug-in configuration service is enabled, a plug-in configuration file is automatically generated for a Web server whenever: • The WebSphere Application Server administrator defines new Web server. • An application is deployed to an Application Server. • An application is uninstalled. • A virtual host definition is updated and saved.

Button	Resulting Action
Propagate Plug-in	Choosing this action will copy the `plugin-cfg.xml` file from the local directory where the Application Server is installed to the remote machine. If you are using IBM HTTP Server V6 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 the name of the node on which the Web server is defined.

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

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

If IBM HTTP Server is defined on an Unmanaged node, you will need to start the IBM HTTP Server administration server before you can start and stop IBM HTTP Server.

Note that if the status is *Unavailable*, the node agent or IBM HTTP Server administration server is not running in that node, and you must start the node agent before you can start the Web server.

Web server configuration

Use this page to configure Web server properties.

Web servers

To view this administrative console page click **Servers** > **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 - Windows operating systems only

Specifies the 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 > 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** > **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 the properties of an IBM HTTP Server Web server that is created on an unmanaged node.

Create a new, unmanaged node by clicking **System administration** > **Nodes** > **Add Node**. Create a Web server using the newly-created, unmanaged node by clicking **Servers** > **Web Servers** > **New**. To view the administrative console page for Remote Web server management, click **Servers** > **Web Servers** > *Web_server_name* > **Remote Web server management**.

Web servers

Port	Indicates the port to access the administration server (default is 8008).
Use SSL	Specifies if the port is secure.
User ID	Specifies a user ID in the <*install_dir*>/conf/admin.passwd file. Create this with the htpasswd script file, located in the <*install_dir*>/bin directory.
Password	Specifies a password in the <*install_dir*>/conf/admin.passwd file. Create this with the htpasswd script file, located in the <*install_dir*>/bin directory.

Web server configuration file

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

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** > **Web Servers** > *Web_server_name* > **Configuration settings** > **Global directives**.

Security enabled

Specifies if security is enabled in your Web server.

Server name

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

Listen port

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

Document root

The directory where the Web server will serve files.

Keystore filename

Specifies the name you have assigned to your keystore.

Keystore directory

Specifies the target directory of your keystore on the machine where the Web server is installed.

SSL Version 2 timeout

Specifies the SSL Version 2 timeout.

SSL Version 3 timeout

Specifies the SSL Version 3 timeout.

Keystore certificate label

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

Virtual hosts collection

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

To view this administrative console page, click **Servers > 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**.

Virtual hosts detail

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

To view this administrative console page, click **Servers > 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 5. Creating and deleting profiles

This topic describes how to create and delete profiles. A profile is the set of files that define the runtime environment. At least one profile must exist to run WebSphere Application Server.

This task assumes a basic familiarity with the **manageprofiles** command, the Profile Management tool, and system commands.

You usually create a profile when you install the product. Depending on which WebSphere Application Server product you have, you might create additional profiles.

You can create profiles through the Profile Management tool or the **manageprofiles** command.

You cannot use the Profile Management tool to create profiles on
- 64-bit platforms
- ▶ Linux The zLinux platform

▶ Linux ▶ HP-UX ▶ Solaris ▶ AIX Non-root users can create their own profiles so that they can manage their own Application Servers. Typically, non-root users would manage Application Servers for development purposes.

You can delete profiles through the **manageprofiles** command or by other means if necessary. You might delete a profile if the configuration that you specified in the profile is not what you want.

Perform any of the following tasks to create or delete profiles.
- Create profiles through the Profile Management tool.

  This topic describes how to create profiles for WebSphere Application Server through the Profile Management tool.
- Create profiles through commands.

  This topic describes how to set up and use the profile environment through commands.
- Create profiles for non-root users.

  This topic discusses how to give a non-root user permissions for files and directories so that the non-root user can create a profile.
- Delete a profile.

  This topic discusses how to delete a profile with and without the **manageprofiles** command.

You created a profile to manage your WebSphere Application Server runtime environment and might have started an Application Server, depending on the tasks you followed. You might have deleted a profile.

Depending on the tasks that you completed, you can start WebSphere Application Server, or proceed to other tasks, such as deploying an application.

Profile concepts

The WebSphere Application Server profile defines the runtime environment. The profile includes all of the files that the server processes in the runtime environment and can change. This topic discusses the main terms and concepts that are associated with profiles.

You can create a runtime environment either through the **manageprofiles** command or the Profile Management tool graphical user interface.

You can use the wizard to enter most of the parameters that are described in this topic. Some parameters, however, require you to use the **manageprofiles** command. You must use the **manageprofiles** command to delete a profile, for instance because the Profile Management tool does not provide a deletion function.

However, the Profile Management tool also performs tasks that the **manageprofiles** command does not. For instance, the wizard creates the cell in a single step whereas the command version requires two separate invocations of the manageprofiles command.

Core product files

The core product files are the shared product binaries, which are shared by all profiles.

The directory structure for the product has two major divisions of files in the installation root directory for the product:

- The core product files are shared product binary files that do not change unless you install a refresh pack, a fix pack, or an interim fix. Some log information is also updated.

  The following list shows default installation locations for root users on supported platforms:

  – AIX /usr/IBM/WebSphere/AppServer

  – Linux HP-UX Solaris /opt/IBM/WebSphere/AppServer

  – Windows C:\Program Files\IBM\WebSphere\AppServer
- The *app_server_root*/profiles directory is the default directory for creating profiles.

When it is desirable to have binaries at different service levels, you must use a separate installation of WebSphere Application Server for each service level.

The configuration for every defined Application Server process is within the profiles directory unless you specify a new directory when you create a profile. These files change as often as you create a new profile, reconfigure an existing profile, or delete a profile.

All of the folders except for the profiles directory and a few others such as the logs directory and the properties directory do not change, unless you install service fixes. The profiles directory, however, changes each time you add, change, or delete a profile. The profiles directory is the default repository for profiles. However, you can put a profile anywhere on the machine or system, provided enough disk space is available.

If you put a profile in another existing folder in the installation root directory, a risk exists that the profile might be affected by the installation of a service fix that applies maintenance to the folder. Use a directory outside of the installation root directory when using a directory other than the profiles directory for creating profiles.

Why and when to create a profile

The **manageprofiles** command-line tool defines each Application Server profile for the product.

Run the wizard or the manageprofiles command each time that you want to create a stand-alone Application Server. A need for more than one stand-alone Application Server on a machine is common.

Administration is greatly enhanced when using profiles instead of multiple product installations. Not only is disk space saved, but updating the product is simplified when you maintain only a single set of product core files. Also, creating new profiles is faster and less prone to error than full product installations, allowing a developer to create separate profiles of the product for development and testing.

You can run the Profile Management tool or the command line tool to create a new Application Server environment on the same machine as an existing one. Define unique characteristics (such as profile name and node name) for the new profile. Each profile has its own administrative console and administrative scripting interface. Each Application Server process shares all run-time scripts, libraries, the Software Development Kit, and other core product files.

The installation program for WebSphere Application Server uses the manageprofiles command to create an Application Server profile named default.

Profile types

Templates for each profile are located in the `app_server_root/profileTemplates` directory.

Within this directory are various directories that correspond to different profile types and that vary with the type of product installed. The directories are the paths that you indicate while using the manageprofiles command with the -templatePath option. You can also specify profile templates that lie outside the installation root, if you happen to have any.

See the -templatePath parameter description in the Manageprofiles command topic in the *Using the administrative clients* PDF for more information.

Use the **manageprofiles** command to create stand-alone application server profiles. The basic function of the application server is to serve applications to the Internet or to an intranet. Each stand-alone application server has its own administrative console application, which you use to manage the application server. You can also use the wsadmin scripting facility to perform every function that is available in the administrative console application.

Security policy for Application Server profiles

In environments where you plan to have multiple stand-alone Application Servers, the security policy of each Application Server profile is independent of the others. Changes to the security policy in one Application Server profile is not synchronized with the other profiles.

Installed file set

You decide where to install the files that define a profile.

The default location is in the profiles directory in the installation root directory. You can change the location on the Profile Management tool or in a parameter when using the command line tool. For example, assume that you create two profiles on a Linux platform with host name devhost1. The profile directories resemble the following example if you do not relocate them:

```
/opt/IBM/WebSphere/AppServer/profiles/AppSrv01
/opt/IBM/WebSphere/AppServer/profiles/AppSrv02
```

If you specify a different directory, such as /opt/profiles for the profile directory, the profile directories resemble the directories shown in the following example:

```
/opt/profiles/AppSrv01
/opt/profiles/AppSrv02
```

The following directories exist within a typical profile. This example assumes that a profile named AppSrv01 exists:

- `/opt/IBM/WebSphere/AppServer/profiles/AppSrv01/bin`
- `/opt/IBM/WebSphere/AppServer/profiles/AppSrv01/config`
- `/opt/IBM/WebSphere/AppServer/profiles/AppSrv01/configuration`
- `/opt/IBM/WebSphere/AppServer/profiles/AppSrv01/etc`
- `/opt/IBM/WebSphere/AppServer/profiles/AppSrv01/firststeps`

- /opt/IBM/WebSphere/AppServer/profiles/AppSrv01/installableApps
- /opt/IBM/WebSphere/AppServer/profiles/AppSrv01/installedApps
- /opt/IBM/WebSphere/AppServer/profiles/AppSrv01/installedConnectors
- /opt/IBM/WebSphere/AppServer/profiles/AppSrv01/installedFilters
- /opt/IBM/WebSphere/AppServer/profiles/AppSrv01/logs
- /opt/IBM/WebSphere/AppServer/profiles/AppSrv01/properties
- /opt/IBM/WebSphere/AppServer/profiles/AppSrv01/samples
- /opt/IBM/WebSphere/AppServer/profiles/AppSrv01/temp
- /opt/IBM/WebSphere/AppServer/profiles/AppSrv01/wstemp

Profiles: required disk space

A minimum amount of space must be available in the directory where you create a profile.

An error can occur when you do not provide enough space to create a profile. Verify that you have, in addition to the minimum space required for a particular profile, an additional 40 MB of space. The 40 MB of space is used for log files and temporary files.

You must have 200 MB of available disk space in the directory where you create an Application Server profile.

Setting up and using the profile environment through commands

You can set up and use the profile environment through commands or the Profile Management tool. Use this example to set up and use the profile environment through commands.

This task assumes a basic familiarity with the **manageprofiles** command, other Application Server commands, and system commands.

Before you can create and use a profile, you must install WebSphere Application Server.

Perform the following steps to create and use the profile environment. This example deals with the profile environment of a stand-alone Application Server on the Windows platform.

1. Create the server profile from the original installation by using the `app_server_root\bin\manageprofiles.bat` script for the Windows platform. The script is *app_server_root*/bin/manageprofiles.sh on operating systems such as AIX or Linux.

   Assume that you create the profile by using the defaults. The following script is an example for creating an Application Server profile on the Windows platform:

   ```
   C:\Program Files\IBM\WebSphere\AppServer\bin\manageprofiles -create
   -templatePath C:\Program Files\IBM\WebSphere\AppServer\profileTemplates\default
   ```

   (The script is displayed on multiple lines for printing purposes.)
2. Change directories to the \bin directory of the new server profile.

   For example, issue the following command:

   ```
   cd Program Files\IBM\WebSphere\AppServer\profiles\AppSrv01\bin
   ```
3. Start server1 by changing directories to the `app_server_root\bin` directory of the original installation and issuing the **startServer** command.

   ```
   startServer server1 -profileName AppSrv01
   ```
4. Display a list of the ports assigned during profile creation.

   Open the `portdef.props` file in the `Program Files\IBM\WebSphere\AppServer\profiles\AppSrv01\properties` directory.

5. Open the administrative console for server1.

   The port for the administrative console is defined on the HTTP_TRANSPORT_ADMIN setting. If the value of the setting is `20003`, specify the following Web address in your browser:

   ```
   http://hostname_or_IP_address:20003/ibm/console/
   ```

You created an Application Server profile, started an Application Server, and accessed the administrative console through commands.

Deploy an application.

Creating profiles through the graphical user interface

This topic describes how to create runtime environments for WebSphere Application Server. Each runtime environment is created within a *profile*. A profile is the set of files that define the runtime environment. The Profile Management tool creates the profile for each runtime environment.

Before using the Profile Management tool, install the core product files.

The Profile Management tool is the graphical interface to the **manageprofiles** command. See the description of the manageprofiles command for more information.

You must provide enough system temporary space to create a profile. For information on the requirements, see the “Profiles: required disk space” on page 162 topic.

You cannot use the Profile Management tool to create profiles on

- 64-bit platforms
- Linux The zLinux platform

The installation procedure creates one profile named `default` for an application server named `server1`. You can use the Profile Management tool to create more application server processes. For example, a second profile can allow two different teams in a department to test independently of one another using the same machine.

1. Install the product to create the core product files.
2. Start the Profile Management tool to create a new runtime environment.

   Several ways exist to start the wizard:

   - Issue the command directly from a command line.

     For operating systems such as AIX or Linux, the command is in the *app_server_root*`/bin/ProfileManagement` directory. For the Windows platform, the command is in the *app_server_root*`\bin\ProfileManagement` directory. The name of the command varies per platform:

     – Linux HP-UX Solaris AIX `pmt.sh`

     – Windows `pmt.bat`
   - Select the Profile Management tool from the First steps console.
   - Windows Use the shortcut capability of the Windows operating system to access the Profile Management tool.
   - Linux Use the shortcut capability of the Linux operating system to access the Profile Management tool.

   This task selects the Profile Management tool from the First steps console:

   a. Open a command window.

   b. Change directories to the `firststeps` directory in the profile root directory:

      The profile root varies by platform:

- AIX HP-UX Linux Solaris *profile_root*/`firststeps`
- Windows *profile_root*\\`firststeps`

c. Issue the **firststeps** command to start the console:

- Linux HP-UX Solaris AIX `./firststeps.sh`
- Windows `firststeps.bat`

d. Select the Profile Management tool option on the console.

See the description of the “firststeps command” on page 17 for more information.

3. Create another stand-alone application server.

   See “Creating an application server profile.”

   The installation procedure creates a stand-alone application server during installation. However, you can use the Profile Management tool to create additional stand-alone application servers.

See the description of the manageprofiles command to learn more about the command-line alternative method of creating a profile, and to see examples of using the command.

See Chapter 1, “Planning the installation,” on page 1 for examples of configurations that you can create by creating profiles.

Creating an application server profile

The Profile Management tool can create an application server profile on any machine where the core product files exist.

Before using the Profile Management tool, install the core product files.

The Profile Management tool is the graphical interface to the **manageprofiles** command. See the description of the manageprofiles command for more information.

You must provide enough system temporary space to create a profile. For information on the requirements, see the “Profiles: required disk space” on page 162 topic.

The Installation wizard creates an application server profile with a server named server1. You can create additional profiles. Each additional profile is an application server named server1.

This procedure describes creating an application server profile using the graphical user interface that is provided by the Profile Management tool.

You can also use the **manageprofiles** command to create an application server profile. See the description of the manageprofiles command for more information.

1. Start the Profile Management tool to create a new runtime environment.

   Several ways exist to start the wizard:

   - Issue the command directly from a command line.

     For operating systems such as AIX or Linux, the command is in the *app_server_root*/`bin/ProfileManagement` directory. For the Windows platform, the command is in the *app_server_root*\\`bin\ProfileManagement` directory. The name of the command varies per platform:

     – Linux HP-UX Solaris AIX `pmt.sh`

     – Windows `pmt.bat`
   - Select the Profile Management tool from the First steps console.
   - Windows Use the shortcut capability of the Windows operating system to access the Profile Management tool.

- Linux Use the shortcut capability of the Linux operating system to access the Profile Management tool.

This task selects the Profile Management tool from the First steps console:

a. Open a command window.

b. Change directories to the `firststeps` directory in the profile root directory:

   The profile root varies by platform:

   - AIX HP-UX Linux Solaris *profile_root*/firststeps
   - Windows *profile_root*\firststeps

c. Issue the **firststeps** command to start the console:

   - Linux HP-UX Solaris AIX `./firststeps.sh`
   - Windows `firststeps.bat`

d. Select the Profile Management tool option on the console.

See the description of the "firststeps command" on page 17 for more information.

2. Click **Next** on the Welcome panel.

   The wizard displays the Profile type selection panel.

3. Click **Next**.

   The wizard displays the Profile name panel.

   Each profile that you create must have a name. The name is the name of the folder that contains all of the files that define the runtime environment for the profile. When you have more than one profile, you can tell them apart at their highest level by this name.

4. Specify a name for the profile and the directory path for the profile directory, or accept the defaults, then click **Next**.

   Profile naming guidelines: Double-byte characters are supported. The profile name can be any unique name with the following restrictions. Do not use any of the following characters when naming your profile:

   - Spaces
   - Illegal special characters that are not supported within the name of a directory on your operating system, such as *&?
   - Slashes (/) or (\)

   The default profile

   The first profile that you create on a machine is the default profile. The default profile is the default target for commands that are issued from the `bin` directory in the product installation root. When only one profile exists on a machine, every command works on the only server process in the configuration. To make another profile the default profile during subsequent profile creation, check **Make this profile the default** on the Profile name and location panel of the Advanced profile creation path.

   Addressing a profile in a multiprofile environment

   When two or more profiles exist on a machine, certain commands require that you specify the profile to which the command applies. These commands use the -profileName parameter to identify which profile to address. You might find it easier to use the commands that are in the `bin` directory of each profile.

   A command in the `profiles/`*`profile_name`*`/bin` directory has two lines. The first line sets the WAS_USER_SCRIPT environment variable for the command window. The variable sets up the command environment to address the profile. The second line calls the actual command in the *app_server_root*/`bin` directory.

   The actual command queries the command shell to determine the calling profile and to autonomically address the command to the calling profile.

Default profile information

The default profile name is *profileTypeProfileName*, where:

- *profileType* is a value of `AppSrv`, `DMgr`, or `Custom`.
- *ProfileName* is a sequential number that is used to create a unique profile name.

AIX HP-UX Linux Solaris The default profile directory is *app_server_root*/profiles, where:

- *app_server_root* is the WebSphere Application Server installation root.

Windows The default profile directory is *app_server_root*\profiles, where:

- *app_server_root* is the WebSphere Application Server installation root.

5. On the Node and host names panel, specify the characteristics for the application server; click **Next**.

   Use unique names for each application server that you create.

   Reserved names: Avoid using reserved folder names as field values. The use of reserved folder names can cause unpredictable results. The following words are reserved:

   - cells
   - nodes
   - servers
   - clusters
   - applications
   - deployments

Field name	Default value	Constraints	Description
Node name	*shortHostName* Node *NodeNumber* where: • *shortHostName* is the short host name. • *NodeNumber* is a sequential number starting at 01.	Avoid using the reserved words.	Pick any name you want. To help organize your installation, use a unique name if you plan to create more than one application server on the machine.
Host name	The long form of the domain name server (DNS) name	Addressable through your network.	Use the actual DNS name or IP address of your machine to enable communication with your machine. See additional information about the host name following this table.

Node name considerations:

Windows **Directory path considerations:** The installation directory path must be less than or equal to 60 characters.

Host name considerations:

The host name is the network name for the physical machine on which the node is installed. The host name must resolve to a physical network node on the server. When multiple network cards exist in the server, the host name or IP address must resolve to one of the network cards. Remote nodes use the host name to connect to and communicate with this node. Selecting a host name that other machines can reach within your network is extremely important. Do not use the generic identifier, localhost, for this value. Also, do not attempt to install WebSphere Application Server products on a machine with a host name that uses characters from the double-byte character set (DBCS). DBCS characters are not supported when used in the host name.

If you define coexisting nodes on the same computer with unique IP addresses, define each IP address in a domain name server (DNS) look-up table. Configuration files for stand-alone Application Servers do not provide domain name resolution for multiple IP addresses on a machine with a single network address.

The value that you specify for the host name is used as the value of the hostName property in configuration documents for the stand-alone Application Server. Specify the host name value in one of the following formats:

- Fully qualified domain name server (DNS) host name string, such as xmachine.manhattan.ibm.com
- The default short DNS host name string, such as xmachine
- Numeric IP address, such as 127.1.255.3

The fully qualified DNS host name has the advantages of being totally unambiguous and flexible. You have the flexibility of changing the actual IP address for the host system without having to change the Application Server configuration. This value for the host name is particularly useful if you plan to change the IP address frequently when using Dynamic Host Configuration Protocol (DHCP) to assign IP addresses. A format disadvantage is a dependency on DNS. If DNS is not available, then connectivity is compromised.

The short host name is also dynamically resolvable. A short name format has the added ability of being redefined in the local hosts file so that the system can run the Application Server, even when disconnected from the network. Define the short name to 127.0.0.1 (local loopback) in the hosts file to run disconnected. A format disadvantage is a dependency on DNS for remote access. If DNS is not available, then connectivity is compromised.

A numeric IP address has the advantage of not requiring name resolution through DNS. A remote node can connect to the node that you name with a numeric IP address without DNS being available. A format disadvantage is that the numeric IP address is fixed. You must change the setting of the hostName property in Express configuration documents whenever you change the machine IP address. Therefore, do not use a numeric IP address if you use DHCP, or if you change IP addresses regularly. Another format disadvantage is that you cannot use the node if the host is disconnected from the network.

After specifying application server characteristics, the wizard displays the Administrative security panel.

6. Optionally enable administrative security; click **Next**.

   You can enable administrative security now during profile creation, or later from the console. If you enable administrative security now, supply a user name and password to log onto the administrative console.

   If you installed the Samples and you chose to deploy them, the Samples require an account under which to run. Supply the Samples password for the account. You cannot change the user name of the account.

   After specifying security characteristics, the wizard displays the Port value assignment panel if you previously selected Advanced profile creation.

7. If you selected Typical profile creation at the beginning of these steps, go to the step that displays the Profile summary panel.
8. Verify that the ports specified for the stand-alone application server are unique; click **Next**.

   Port conflict resolution:

   Ports are recognized as being in use if

   - They are assigned to a profile created under an installation performed by the current user.
   - The port is currently in use.

   Validation of ports occurs when you access the Port value assignment panel. Conflicts can still occur between the Port value assignment panel and the Profile Creation Complete panel because ports are not assigned until profile creation completes.

   If you suspect a port conflict, determine the ports used during profile creation by examining the `*profile_root*/properties/portdef.props` file. Included in this file are the keys and values used in setting the ports. If you discover ports conflicts, you can reassign ports manually. To reassign ports, run the `updatePorts.ant` file through the ws_ant script.

Windows Linux The wizard displays the Windows service definition panel if you are installing on a Windows platform and the installing ID has the administrative group privilege, or the Linux service definition panel if you are installing on a supported Linux platform and the ID that runs the Profile Management tool is the root user.

9. Choose whether to run the Application Server as a Windows service on a Windows platform or as a Linux service on a Linux platform; click **Next**.

   - Windows

     The Windows service definition panel displays for the Windows platform only if the ID that installs the Windows service has the administrator group privilege. However, you can run the **WASService.exe** command to create the Windows service as long as the installer ID belongs to the administrator group. See “Automatically restarting server processes” on page 282 for more information.

     WebSphere Application Server attempts to start Windows services for application server processes that are started by a **startServer** command. For example, if you configure an application server as a Windows service and issue the **startServer** command, the **wasservice** command attempts to start the defined service.

     If you chose to install a local system service, you do not have to specify your user ID or password. If you create a specified user type of service, you must specify the user ID and the password for the user who runs the service. The user must have *Log on as a service* authority for the service to run properly. If the user does not have *Log on as a service* authority, the Profile Management tool automatically adds the authority.

     To perform this installation task, the user ID must not have spaces in its name. In addition to belonging to the administrator group, the ID must also have the advanced user right *Log on as a service*. The Installation wizard grants the user ID the advanced user right if it does not already have it, if the user ID belongs to the administrator group.

     You can also create other Windows services after the installation is complete, to start other server processes. See “Automatically restarting server processes” on page 282 for more information.

     You can remove the Windows service that is added during profile creation during profile deletion.

     IPv6 considerations

     Profiles created to run as a Windows service fail to start when using IPv6 if the service is configured to run as *Local System*. Create a user-specific environment variable to enable IPv6. Since this environment variable is a user variable instead of a *Local System* variable, only a Windows service that runs as that specific user can access this environment variable. By default, when a new profile is created and configured to run as a Windows service, the service is set to run as *Local System*. When the WebSphere Application Server Windows service tries to run, the service is unable to access the user environment variable that specifies IPv6, and thus tries to start as IPv4. The server does not start correctly in this case. To resolve the problem, when creating the profile, specify that the WebSphere Application Server Windows service runs as the same user ID under which the environment variable that specifies IPv6 is defined, instead of as *Local System*.

     Default Windows service information

     Windows The defaults for the Windows service definition panel are as follows:

     - The default is to run as a Windows service.
     - The service process is selected to run as a system account.
     - The user account is the current user name. User name requirements are the requirements that the Windows operating system imposes for a user ID.
     - The startup type is `automatic`. The values for the startup type are those values that the Windows operating system imposes.

   - Linux

The Linux service definition panel displays if the current operating system is a supported version of Linux and the current user has the appropriate permissions.

WebSphere Application Server attempts to start Linux services for application server processes that are started by a **startServer** command. For example, if you configure an application server as a Linux service and issue the **startServer** command, the **wasservice** command attempts to start the defined service.

By default, WebSphere Application Server is not selected to run as a Linux service.

To create the service, the user that runs the Profile Management tool must be the root user. If you run the Profile Management tool with a non-root user ID, the Linux service definition panel does not display, and no service is created.

You must specify a user name under which the service runs.

To delete a Linux service, the user must be the root user or have proper privileges for deleting the service. Otherwise, a removal script is created that the root user can run to delete the service on the user's behalf.

The next wizard panel displays the Web server definition panel if you previously selected Advanced profile creation.

10. For Advanced profile creation, if you choose to include a Web server definition in the profile now, specify the Web server characteristics on the panels, and click **Next** until you are through the Web server definition panels.

    If you use a Web server to route requests to WebSphere Application Server, you need to include a Web server definition. You can include the definition now, or define the Web server to WebSphere Application Server later. If you define the Web server definition during the creation of this profile, you can install the Web server and its plug-in after you create the profile. However, you must install both to the paths that you specify on the Web server definition panels. If you define the Web server to WebSphere Application Server after you create this profile, you must define the Web server in a separate profile.

    The wizard displays the Profile Creation Summary panel.

11. Click **Create** to create the application server or click **Back** to change the characteristics of the application server.

    The Profile creation progress panel, which shows the configuration commands that are running, is displayed.

    When the profile creation completes, the wizard displays the Profile Creation Complete panel.

12. Optionally, select **Launch the First steps console**, **Create another profile**, or both; click **Finish** to exit.

    With the First steps console, you can create additional profiles and start the application server. Use the Create another profile option to create additional profiles.

You created an application server profile. The node within the profile has an application server named server1.

Refer to the description of the manageprofiles command to learn about creating a profile using a command instead of a wizard.

Deploy an application to get started.

See Fast paths for WebSphere Application Server to get started deploying applications.

Creating profiles for non-root users

This topic discusses how to give a non-root user permissions for files and directories so that the non-root user can create a profile.

This task assumes a basic familiarity with the **manageprofiles** command, the Update Installer wizard, and system commands.

This task uses the following terms:

- `Root users` refers to root users on operating systems such as AIX or Linux, and to administrators on Windows platforms.
- `Non-root users` refers to non-root users on operating systems such as AIX or Linux, and to non-administrators on Windows platforms.
- `Installer` refers to a root user or a non-root user.

Remember: An ease-of-use limitation exists for non-root users who create profiles. Mechanisms within the Profile Management tool that suggest unique names and port values are disabled for non-root users. The non-root user must change the default field values in the Profile Management tool for the profile name, node name, and port assignments. Consider assigning non-root users a range of values for each of the fields. You can assign responsibility to the non-root users for adhering to their proper value ranges and for maintaining the integrity of their own definitions.

Non-root users might typically need these tasks completed so that they can start their own WebSphere Application Servers in development environments. For instance, an application developer might test an application on a WebSphere Application Server in a profile that he owns.

- Create a profile as an installer and assign ownership to a non-root user.

  This topic describes how the installer creates a profile and assigns ownership of the profile directory to a non-root user so that the non-root user can start WebSphere Application Server for a specific profile.
- Grant write permission of files and directories to a non-root user for profile creation.

  This topic describes how an installer authorizes a group to certain files and directories so that non-root users in the group can create profiles.
- Install maintenance as an installer and change the ownership of profile related files.

  This topic describes how to install WebSphere Application Server maintenance and change the ownership of new profile files to the non-root user that owns the profile. The installer changes ownership of the files so that the non-root user can then successfully start WebSphere Application Server.

Depending on the tasks that the installer completes, the installer has:

- Created a profile for a non-root user and assigned ownership of the profile directory to the non-root user
- Granted permission to the appropriate directories so that non-root users can create profiles
- After installing maintenance, changed ownership of new profile files in a directory that is owned by a non-root user so that the non-root user can successfully start WebSphere Application Server

Depending on the tasks that the installer completes, a non-root user can create a profile, start WebSphere Application Server, or do both.

Creating a profile as an installer and assigning ownership to a non-root user

The installer can create a profile and assign ownership of the profile directory to a non-root user so that the non-root user can start WebSphere Application Server for a specific profile. Use this example to accomplish the tasks through commands.

This task assumes a basic familiarity with the **manageprofiles** command and system commands.

This task uses the following terms:

- `Root users` refers to root users on operating systems such as AIX or Linux, and to administrators on Windows platforms.

- `Non-root users` refers to non-root users on operating systems such as AIX or Linux, and to non-administrators on Windows platforms.
- `Installer` refers to a root user or a non-root user.

Before you can create a profile, you must install WebSphere Application Server.

Have the installer perform the following step to create a profile and assign ownership for the profile directory and the logs directory. The ownership is assigned to a non-root user ID that is different from the installer ID. The non-root user needs access to these directories to start WebSphere Application Server.

1. Create the profile:

   Linux HP-UX Solaris AIX

   ```
   ./manageprofiles.sh -create -profileName profile01 -profilePath
   app_server_root/profiles/profile01 -templatePath
   app_server_root/profileTemplates/default
   ```

   Windows

   ```
   manageprofiles -create -profileName profile01 -profilePath C:\
   app_server_root\profiles\profile01 -templatePath C:\
   app_server_root\profileTemplates\default
   ```

   The commands are split on multiple lines for printing purposes.
2. Change ownership of the profile01 profile directory to the user1 non-root user:

   Linux HP-UX Solaris AIX For example, issue the following command:

   ```
   chown -R user1 app_server_root/profiles/profile01
   ```

   Windows Follow instructions in the Windows documentation to grant user1 access to the following directory:

   ```
   C:\app_server_root\profiles\profile01
   ```
3. Change the ownership of the logs directory for the profile01 profile to the user1 non-root user to prevent displaying log messages to the console:

   Linux HP-UX Solaris AIX Issue the following command:

   ```
   chown -R user1 app_server_root/logs/manageprofiles/profile01
   ```

   Windows Follow instructions in the Windows documentation to grant user1 access to the following directory:

   ```
   C:\app_server_root\logs\manageprofiles\profile01
   ```

The installer has created a profile and changed ownership of the profile directory and log directory to a non-root user.

The non-root user can start WebSphere Application Server associated with, in this example, profile01.

Granting write permission of files and directories to a non-root user for profile creation

The installer can grant write permission of the appropriate files and directories to a non-root user. The non-root user can then create the profile. The installer can create a group for users who are authorized to create profiles, or the installer can give individual users the ability to create profiles. The following example shows how to create a group that is authorized to create profiles.

This task assumes a basic familiarity with system commands.

This task uses the following terms:

- `Root users` refers to root users on operating systems such as AIX or Linux, and to administrators on Windows platforms.

- `Non-root users` refers to non-root users on operating systems such as AIX or Linux, and to non-administrators on Windows platforms.
- `Installer` refers to a root user or a non-root user.

Have the installer perform the following steps to create the profilers group and give the group proper permissions to create a profile.

1. Log on as the installer to the system that has WebSphere Application Server installed.
2. Create the profilers group that you can use to create profiles.
3. Create a user named user1 to create profiles.
4. Add the installer and user1 to the profilers group.
5. Linux HP-UX Solaris AIX Log off and back on as the installer to pick up the new group.
6. As the installer, use operating system tools to change directory and file permissions.

   Linux HP-UX Solaris AIX The following example assumes that the installation root directory is `/opt/IBM/WebSphere/AppServer` .

   ```
   chgrp  profilers /opt/IBM/WebSphere/AppServer/logs/manageprofiles
   chmod  g+wr  /opt/IBM/WebSphere/AppServer/logs/manageprofiles
   chgrp  profilers /opt/IBM/WebSphere/AppServer/properties
   chmod  g+wr  /opt/IBM/WebSphere/AppServer/properties
   chgrp  profilers /opt/IBM/WebSphere/AppServer/properties/fsdb
   chmod  g+wr  /opt/IBM/WebSphere/AppServer/properties/fsdb
   chgrp  profilers /opt/IBM/WebSphere/AppServer/properties/profileRegistry.xml
   chmod  g+wr  /opt/IBM/WebSphere/AppServer/properties/profileRegistry.xml
   chgrp -R profilers /opt/IBM/WebSphere/AppServer/profileTemplates
   ```

   HP-UX

   ```
   chmod  -R g+wr /opt/IBM/WebSphere/AppServer/profileTemplates/default/documents
   ```

   For the HP-UX operating system, the ownership of files is preserved when the files are copied to the profile directory during profile creation. You granted write permission to the profile directory so that files copied to the profile directory can be modified as part of the profile creation process. Files that are already in the `profileTemplate` directory structure prior to the start of profile creation are not modified during profile creation.

   Linux

   ```
   chgrp profilers /opt/IBM/WebSphere/AppServer/properties/Profiles.menu
   chmod  g+wr /opt/IBM/WebSphere/AppServer/properties/Profiles.menu
   ```

   Windows The following example assumes that the installation root directory is `C:\Program Files\IBM\WebSphere\AppServer` . Follow instructions in the Windows documentation to give the profilers group read and write permission to the following directories and their files:

   ```
   C:\Program Files\IBM\WebSphere\AppServer\logs\manageprofiles
   C:\Program Files\IBM\WebSphere\AppServer\properties
   C:\Program Files\IBM\WebSphere\AppServer\properties\fsdb
   C:\Program Files\IBM\WebSphere\AppServer\properties\profileRegistry.xml
   ```

   You might have to change the permissions on additional files if the non-root user encounters permission problems. For example, if you allow a non-root user to delete a profile, the user might have to delete the following file:

   Linux HP-UX Solaris AIX *app_server_root*/properties/profileRegistry.xml_LOCK

   Windows *app_server_root*\properties\profileRegistry.xml_LOCK

   - Give write access to the non-root user for the file to allow the user to delete the file. If the non-root user still cannot delete the profile, the installer can delete the profile.

The installer created the profilers group and gave the group proper permissions to certain directories and files to create a profile. These directories and files are the only ones in the installation root of WebSphere Application Server to which a non-root user needs to write to create a profile.

Have the non-root user that belongs to the profilers group create a profile in a directory that the non-root user owns and to which the non-root user has write permission, but not in the installation root directory of the product.

Installing maintenance as an installer and changing the ownership of profile related files

The installer can install maintenance packages for WebSphere Application Server that includes required service for existing profiles that are owned by a non- user. Installing a maintenance package that contains service for a non-root profile makes the installer owner of any new files that the maintenance package creates. The following example shows how to install the maintenance and change the ownership of the new files so that a non-root user can successfully start WebSphere Application Server.

This task assumes a basic familiarity with the Update Installer wizard and system commands.

This task uses the following terms:

- `Root users` refers to root users on operating systems such as AIX or Linux, and to administrators on Windows platforms.
- `Non-root users` refers to non-root users on operating systems such as AIX or Linux, and to non-administrators on Windows platforms.
- `Installer` refers to a root user or a non-root user.

Before you can update a profile, you must install WebSphere Application Server and create a profile.

This example assumes that the installer:

- Applies Refresh Pack 2-required service for the Java Database Connector (JDBC) resource provider templates and that the service creates new files in a profiles directory that the wsdemo non-root user owns.
- Changes ownership of new profile files to the wsdemo non-root user.

If the installer does not change ownership, when the non-root user starts WebSphere Application Server, the application server encounters an error and issues a message that is similar to the following example:

```
ADMR0104E:
The system is unable to read document
cells/express1Cell/nodes/express1/node-metadata.properties:
java.io.IOException: No such file or directory
```

1. Run the Update Installer wizard to install maintenance packages for WebSphere Application Server.

   When the installer installs Refresh Pack 2, the installer owns the following new JDBC-related files:

   ▶ Linux ▶ HP-UX ▶ Solaris ▶ AIX

   - *profile_root*`/logs/updateProfileJdbcTemplate.log`
   - *profile_root*`/config/templates/system/jdbc-resource-provider-only-templates.xml`
   - *profile_root*`/config/templates/system/jdbc-resource-provider-templates.xml`

   ▶ Windows

   - `C:\`*profile_root*`\logs\updateProfileJdbcTemplate.log`
   - `C:\`*profile_root*`\config\templates\system\jdbc-resource-provider-only-templates.xml`
   - `C:\`*profile_root*`\config\templates\system\jdbc-resource-provider-templates.xml`

2. Reassign ownership of the entire profile directory to the wsdemo non-root user.

   The *profile_root* variable in the following examples is the profile directory that the non-root user owns.

   ▶ Linux ▶ HP-UX ▶ Solaris ▶ AIX Issue the **chown** command.

   ```
   chown -R wsdemo profile_root
   ```

Windows Follow instructions in the Windows documentation to reassign ownership of the *profile_root* profile directory to the wsdemo non-root user.

The installer installed maintenance that creates new files in a non-root user profile directory and changes ownership of the new files to the non-root owner.

The non-root user can start WebSphere Application Server without receiving the ADMR0104E error message.

Deleting a profile

This topic describes how to manually delete a profile.

Before you delete a profile, stop its Application Server to ensure that the Application Server can be deleted.

Before using the manual procedure to remove a profile, try the **manageprofiles** command with the -delete option. For example, issue the following command for your operating system platform:

Linux HP-UX Solaris AIX

```
./manageprofiles.sh -delete
                -profileName profile_name
```

Windows

```
manageprofiles.bat -delete
                -profileName profile_name
```

If you delete a profile that has augmenting templates registered to it in the profile registry, then unaugment actions are attempted prior to the deletion.

If the command does not work, use this procedure to delete the profile.

This procedure describes how to manually delete a profile when the **manageprofiles -delete** command results in the following message:

```
INSTCONFFAILED: Cannot delete profile.
```

1. Issue operating system commands to delete the profile directory.
2. Issue the following command to remove references in the registry to deleted profiles:

   Linux HP-UX Solaris AIX

   ```
   ./manageprofiles.sh -validateAndUpdateRegistry
   ```

   Windows

   ```
   manageprofiles.bat -validateAndUpdateRegistry
   ```

   Editing of the registry is not recommended.

You have now deleted a profile.

See the description of the the manageprofiles command topic in the *Using the administrative clients* PDF to learn more about the command-line method of working with profiles.

See “Creating profiles through the graphical user interface” on page 163 for more information about creating profiles with the Profile Management tool.

Chapter 6. Setting up the administrative architecture

You can monitor and control incorporated nodes and the resources on those nodes by using these tasks with the administrative console or other administrative tools.

If your system uses administrative services, you can specify settings for the service.

Use the settings page for an administrative service to configure administrative services.

Administration service settings

Use this page to view and change the configuration for an administration service.

To view this administrative console page, click **Servers > Application Servers >** *server_name* **> Administration > Administration Services**

Preferred Connector

Specifies the preferred JMX Connector type. Available options, such as SOAPConnector or RMIConnector, are defined using the JMX Connectors page.

Data type	String
Default	SOAP

Extension MBean Providers collection

Use this page to view and change the configuration for JMX extension MBean providers.

You can configure JMX extension MBean providers to be used to extend the existing WebSphere managed resources in the core administrative system. Each MBean provider is a library containing an implementation of a JMX MBean and its MBean XML Descriptor file.

To view this administrative console page, click **Servers > Application Servers >** *server_name* **> Administration > Administration Services > Extension MBean Providers**

Name The name used to identify the Extension MBean provider library.

Description
An arbitrary descriptive text for the Extension MBean Provider configuration.

Classpath
The path to the Java archive (JAR) file that contains the Extension MBean provider library. This class path is automatically added to the Application Server class path.

Extension MBean Provider settings

Use this page to view and change the configuration for a JMX extension MBean provider.

You can configure a library containing an implementation of a JMX MBean, and its MBean XML Descriptor file, to be used to extend the existing WebSphere managed resources in the core administrative system

To view this administrative console page, click **Servers > Application Servers >** *server_name* **> Administration > Administration Services > Extension MBean Providers >** *provider_library_name*

Classpath:

The path to the Java archive (JAR) file that contains the Extension MBean provider library. This class path is automatically added to the Application Server class path. The class loader needs this information to load and parse the Extension MBean XML Descriptor file.

Data type	String

Description:

An arbitrary descriptive text for the Extension MBean Provider configuration. Use this field for any text that helps identify or differentiate the provider configuration.

Data type	String

Name:

The name used to identify the Extension MBean provider library.

Data type	String

Extension MBean collection

You can configure Java Management Extension (JMX) MBeans to extend the existing WebSphere Application Server managed resources in the administrative console. Use this page to register JMX MBeans. Any MBeans that are listed have already been registered.

To view this administrative console page, click **Servers** > **Application Servers** > *server name* > **Administration** > **Administration Services** > **Extension MBean Providers** > *provider library name*> **extensionMBeans**

DescriptorURI
Specifies the location, relative to the provider class path, where the MBean XML descriptor file is located.

Type Specifies the type to use for registering this MBean. The type must match the type that is declared in the MBean descriptor file.

Extension MBean settings

Use this page to view and configure Java Management Extension (JMX) MBeans.

To view this administrative console page, click **Servers** > **Application Servers** > *server name* > **Administration** > **Administration Services** > **Extension MBean Providers** > *provider library name* > **ExtensionMBeans** > *descriptorURI*

descriptorURI:

Specifies the location, relative to the provider class path, where the MBean XML descriptor file is located.

Data type	String

type:

Specifies the type to use for registering this MBean. The type must match the type that is declared in the MBean descriptor file.

Data type	String

Java Management Extensions connector properties

You can specify or set a property in the administrative console, the wsadmin tool, Application Server commands, the scripts that run from a command-line interface, or a custom Java administrative client program that you write. You can also set SOAP connector properties in the `soap.client.props` file.

A Java Management Extensions (JMX) connector can either be a Remote Method Invocation (RMI) connector or a Simple Object Access Protocol (SOAP) connector.

For specific information on how to code the JMX connector properties for the wsadmin tool, the Application Server commands, or scripts, see the particular tool or command. For specific information on how to code the JMX connector properties for a custom Java administrative client program, see the Java API documentation for Application Server.

For the administrative console, this topic specifies the coding of the particular setting or property. Coding of properties in the `soap.client.props` file that are specific to JMX connectors is specified. These properties begin with com.ibm.SOAP. Other properties in the `soap.client.props` file that contain information that can be set elsewhere in the Application Server are not documented here. The coding for the com.ibm.ssl.contextProvider property, which can be set only in the `soap.client.props` file, is specified.

Each profile has a property file at *`installation root`*`/profiles/`*`profile name`*`/properties/soap.client.props`. These property files allows you to set different properties, including security and timeout properties. These properties are the default for all the administrative connections that use the SOAP JMX connector between processes that run in a particular profile. For instance, the wsadmin program running under a particular profile uses the property values from that file for the SOAP connector behavior unless the properties are overridden by some other programmatic means.

To view the JMX connector custom properties administrative console panel that goes with this article, click **Servers > Application servers >***server name* **> Server Infrastructure > Administration > Administration Services > Additional properties > JMX Connectors>***connector type* **> Additional Properties > Custom properties**.

SOAP connector properties

This section discusses JMX connector properties that pertain to SOAP connectors.

SOAP request timeout

The value that you choose depends on a number of factors, such as the size and the number of the applications that are installed on the server, the speed of your machine, and the usage of your machine.

The program default value for the request timeout is `600 seconds`. However, other components that connect to the SOAP client can override the default. Components that use the `soap.client.props` file have a default value of `180 seconds`.

Set the property by using one of the following options:

- Scripts that run from a command-line interface.
- The `soap.client.props` file.

Property	com.ibm.SOAP.requestTimeout
Data type	Integer
Range in seconds	0 to n If the property is zero (0), the request never times out.
Default	180

- The administrative console. Specify the property and the value as a name-value pair on the JMX connector custom properties panel of the administrative console.

Property	requestTimeout
Data type	Integer
Range in seconds	0 to n If the property is zero (0), the request never times out.
Default	600

- A Java administrative client. The property is AdminClient.CONNECTOR_SOAP_REQUEST_TIMEOUT.

Configuration URL

Specify the configuration Universal Resource Locator (URL) property if you want a program to read SOAP properties from this file. You can set the property by using one of the following options:

- Scripts run from a command-line interface. Scripts can pass the Configuration URL property to the Application Server on the com.ibm.SOAP.ConfigURL system property.
- The administrative console. Specify the property and the value as a name-value pair on the JMX connector custom properties panel of the administrative console.

Property	ConfigURL
Data type	String
Valid Value	http://*Path*/soap.client.props
Default	None

- A Java administrative client. Use the AdminClient.CONNECTOR_SOAP_CONFIG property.

Security context provider

This property indicates the Secure Sockets Layer (SSL) implementation to use between the Application Server and the SOAP client. You can specify either IBM Java Secure Sockets Extension (IBMJSSE) or IBM Java Secure Sockets Extension that has undergone Federal Information Processing Standards certification (IBMJSSEFIPS). For information about IBMJSSEFIPS, see the *Using the administrative clients* PDF.

Set the property by using the `soap.client.props` file.

Property	com.ibm.ssl.contextProvider
Data type	String

Valid Values	`IBMJSSE`
	`IBMJSSEFIPS`
	`IBMJSSE2`

Default	IBMJSSE2

Secure Sockets Layer (SSL) security

Use this property to enable SSL security between Application Server and the SOAP client. Set the property by using one of the following options:

- Scripts that run from a command-`line interface.
- The `soap.client.props` file.

Property	com.ibm.SOAP.securityEnabled
Data type	Boolean
Default	False

- The administrative console. Specify the property and the value as a name-value pair on the JMX connector custom properties panel of the administrative console.

Property	securityEnabled
Data type	Boolean
Default	False

- A Java administrative client. Use the AdminClient.CONNECTOR_SECURITY_ENABLED property.

SOAP and RMI connector properties

This section discusses JMX connector properties that pertain to both SOAP connectors and RMI connectors.

Connector type

A connector type of SOAP or RMI, depends on whether Application Server connects to a SOAP server or an RMI server. You can set the property by using one of the following options:

- The wsadmin tool.
- Scripts that run from a command-line interface.
- The administrative console. Specify the property and the value as a name-value pair on the JMX connector custom properties panel of the administrative console.

Property	Type
Data type	String
Valid values	`SOAPConnector` `RMIConnector`
Default	SOAPConnector

- A Java administrative client. Use the AdminClient.CONNECTOR_TYPE property. Specify the connector type by using the AdminClient.CONNECTOR_TYPE_RMI or the AdminClient.CONNECTOR_TYPE_SOAP constants.

Host

The host name or the IP address of the server to which Application Server connects. The server can be a SOAP server or an RMI server. You can set the property by using one of the following options:

- The wsadmin tool.
- Scripts that run from a command-line interface.
- The administrative console. Specify the property and the value as a name-value pair on the JMX connector custom properties panel of the administrative console.

Property	host
Data type	String
Valid values	Host name or IP address
Default	None

- A Java administrative client. Use the AdminClient.CONNECTOR_HOST property.

Port

The port number of the server to which Application Server connects. The server can be a SOAP server or an RMI server. You can set the property by using one of the following options:

- The wsadmin tool.
- Scripts run from a command-line interface.

- The administrative console. Specify the property and the value as a name-value pair on the JMX connector custom properties panel of the administrative console.

Property	port
Data type	Integer
Valid value	Port number
Default	None

- A Java administrative client. Use the AdminClient.CONNECTOR_PORT property.

User name

The user name that Application Server uses to access the SOAP server or the RMI server. You can set the property by using one of the following options:

- The wsadmin tool.
- Scripts run from a command-line interface.
- The `soap.client.props` file.

Property	com.ibm.SOAP.loginUserid
Data type	String
Valid value	The value must match the global SSL settings for SOAP or RMI.
Default	None

- The administrative console. Specify the property and the value as a name-value pair on the JMX connector custom properties panel of the administrative console.

Property	username
Data type	String
Valid value	The value must match the global SSL settings for SOAP or RMI.
Default	None

- A Java administrative client. Use the AdminClient.USERNAME property.

Password

The password that Application Server uses to access the SOAP server or the RMI server. You can set the property by using one of the following options:

- The wsadmin tool.
- Scripts run from a command-line interface.
- The `soap.client.props` file.

Property	com.ibm.SOAP.loginPassword
Data type	String
Valid values	The value must match the global SSL settings for SOAP or RMI.
Default	None

- The administrative console. Specify the property and the value as a name-value pair on the JMX connector custom properties panel of the administrative console.

Property	password
Data type	String

Valid values	The value must match the global SSL settings for SOAP or RMI.
Default	None

- A Java administrative client. Use the AdminClient.PASSWORD property.

RMI connector properties

This section discusses JMX connector properties that pertain to RMI connectors.

Disabling the JSR 160 RMI connector

Support for JMX Remote application programming interface (JSR 160) is enabled by default so that you automatically receive specification-compliant JMX function. To disable the function for a particular server, set the property by using one of the following options:

- The wsadmin tool.
- The administrative console. Specify the property and the value as a name-value pair on the JMX connector custom properties panel of the administrative console.

Property	disableJDKJMXConnector
Data type	string
Value	true

Java Management Extensions connectors

Use this page to view and change the configuration for Java Management Extensions (JMX) connectors.

To view this administrative console page, click **Servers > Application Servers >** *server_name* **> Administration > Administration Services > JMX Connectors**

Java Management Extensions (JMX) connectors communicate with WebSphere Application Server when you invoke a scripting process. There is no default for the type and parameters of a connector. The `wsadmin.properties` file specifies the Simple Object Access Protocol (SOAP) connector and an appropriate port number. You can also use the Remote Method Invocation (RMI) connector.

Use one of the following methods to select the connector type and attributes:

- Specify properties in a properties file.
- Indicate options on the command line.

Type

Specifies the type of the JMX connector.

Data type	Enum
Default	SOAPConnector
Range	**SOAPConnector** For JMX connections using Simple Object Access Protocol (SOAP). **RMIConnector** For JMX connections using Remote Method Invocation (RMI).

JMX connector settings

Use this page to view the configuration for a Java Management Extensions (JMX) connector.

To view this administrative console page, click **Servers > Application Servers >** *server_name* **> Administration Services > JMX Connectors >** *connector_type*

Type:

Specifies the type of the JMX connector.

Data type	Enum
Default	SOAPConnector
Range	**SOAPConnector** For JMX connections using Simple Object Access Protocol (SOAP). **RMIConnector** For JMX connections using Remote Method Invocation (RMI).

Repository service settings

Use this page to view and change the configuration for an administrative service repository.

To view this administrative console page, click **Servers > Application Servers >** *server_name* **Administration > Administration Services > Repository Service**.

Audit Enabled

Specifies whether to audit repository updates in the log file. The default is to audit repository updates.

Data type	Boolean
Default	true

Administration services custom properties

This topic discusses the administration services custom properties that you can set on the administrative console.

To view the administration services custom properties administrative console page that goes with this topic, click: **Servers > Application Server >** *server_name* **> Administration > Administration Services > Custom Property**.

Specify a property and its value as a name-value pair on the Administration services custom properties page.

Disable routing

When a custom managed bean (MBean) is registered directly with the MBean server that runs in a WebSphere Application Server process, the MBean object name is enhanced by default to include the cell, node, and process names as key properties.

To turn off the default behavior, set the following custom property on the application server:

Property name	com.ibm.websphere.mbeans.disableRouting
Data type	string

Value

One or more MBean object names tagged with `<on>...</on>`. You can specify the object name of your MBean or a pattern that matches the names of several MBeans.

Example:

If you register a custom MBean with the `WebSphere:type=custom,name=custommbean1` object name and another custom MBean with the `WebSphere:type=custom,name=custommbean2` object name, each of the following values is valid:

- `<on>WebSphere:type=custom,name=custommbean1</on>`

  The value disables the MBean object name modification for this MBean.
- `<on>WebSphere:type=custom,*</on>`

  The value disables the MBean object name modification for this MBean.
- `<on>WebSphere:type=custom,name=custommbean1</on><on>WebSphere:type=custom,name=custommbean2</on>`

  The value disables the object name modification for both MBeans.

If this custom property is set, an administrative client needs to connect directly to the application server on which the MBean is registered to invoke methods. The MBean cannot participate in all the distributed functions of the administrative system.

Administrative audits

This topic discusses aspects of administrative audits, such as log files that contain the audit information, the administrative actions that are audited, and the types of audit messages that are logged.

Administrative audits use the same logging facility as the rest of the product. The audits are available in both the `activity.log` file and the `SystemOut.log` of the server that performs the action. You do not need to enable trace to produce the audits. However, through the Repository service console page, you can control whether configuration change auditing is done. This type of audit is done by default. Operational command auditing is always enabled. Information about which user performed the change is available only when security is enabled.

The following administrative actions are audited:

- All configuration changes, in terms of the configuration documents that are created, modified, or deleted.
- Certain operational changes, like starting and stopping servers and applications. These managed bean (MBean) operations provide administrative auditing:

Table 8.

MBean type	MBean operations
Server	stop, stopImmediate

Configuration change audits have ADMRxxxxI message IDs, where xxxx is the message number. Operational audits have ADMN10xxI message IDs, where 10xx is the message number.

Here are some examples from the Application Server `SystemOut.log` file:

```
[7/23/03 17:04:49:089 CDT] 39c26dad FileRepositor A ADMR0015I: Document
cells/ellingtonNetwork/security.xml was modified by user u1.
   [7/23/03 17:04:49:269 CDT] 3ea0edb5 FileRepositor A ADMR0016I: Document
cells/ellingtonNetwork/nodes/ellington/app.policy was created by user u1.
   ...
   [7/23/03 17:13:54:081 CDT] 39a572a1 AdminHelper   A ADMN1008I: Attempt
made to start the SamplesGallery application. (User ID = u1)
   ...
[7/23/03 17:39:59:360 CDT] 24865373 AdminHelper   A ADMN1020I: Attempt
made to stop the server1 server. (User ID = u1)
```

The message text is split for printing purposes.

Administrative agents: Resources for learning

Use the following links to find relevant supplemental information about WebSphere Application Server administrative agents and distributed administration. 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:

Administration

- IBM WebSphere Application Server Redbooks

  This site contains a listing of all WebSphere Application Server Redbooks.
- IBM WebSphere developerWorks

  This site is the home of technical information for developers working with WebSphere products. You can download WebSphere software, take a fast path to developerWorks zones, such as VisualAge Java or WebSphere Application Server, learn about WebSphere products through a newcomers page, tutorials, technology previews, training, and Redbooks, get answers to questions about WebSphere products, and join the WebSphere community, where you can keep up with the latest developments and technical papers.
- WebSphere Application Server Support page

  Take advantage of the Web-based Support and Service resources from IBM to quickly find answers to your technical questions. You can easily access this extensive Web-based support through the IBM Software Support portal at URL `http://www-3.ibm.com/software/support/` and search by product category, or by product name. For example, if you are experiencing problems specific to WebSphere Application Server, click **WebSphere Application Server** in the product list. The WebSphere Application Server Support page appears.

Chapter 7. Configuring the environment

Use the following links to find relevant supplemental information about configuring the environment. The information resides on IBM and non-IBM internet sites, whose sponsors control the technical accuracy of the information.

To assist in handling requests among Web applications, Web containers, and application servers, you can configure settings for virtual hosts, variables and shared libraries.

1. Configure virtual hosts.
2. Configure variables.
3. If your deployed applications use shared library files, define the shared library files needed.

   See “Managing shared libraries” on page 197.

Virtual hosts

When you configure WebSphere Application Server, you can associate a virtual host to one or more Web modules. Each Web module can be associated with one and only one virtual host.

A virtual host is a configuration entity that enables a single host machine to resemble multiple host machines. It maintains a list of Multipurpose Internet Mail Extensions (MIME) types that it processes. Resources associated with one virtual host cannot share data with resources associated with another virtual host, even if the virtual hosts share the same physical machine.

Each virtual host has a logical name and a list of one or more DNS aliases by which it is known. A DNS alias is the TCP/IP hostname and port number that is used to request the servlet, for example `yourHostName:80`. When no port number is specified, 80 is assumed.

A client request for a servlet, JavaServer Pages file, or related resource contains a DNS alias and a Uniform Resource Indicator (URI) that is unique to that resource. When a client request for a servlet, JavaServer Pages file, or related resource is received, the DNS alias is compared to the list of all known virtual host groups to locate the correct virtual host, and the URI is compared to the list of all known URI groups to locate the correct URI group. If the virtual host group and URI group are found, the request is sent to the corresponding server group for processing and a response is returned to browser. If a matching virtual host group or URI group is not found, an error is returned to the browser.

The first time that you start an application server, a default virtual host (named default_host) is configured. The DNS aliases for the default virtual host are configured as `*:80` and `*:9080`, where port 80 is the HTTP server port and port 9080 is the port for the default server’s HTTP transport. The default virtual host includes common aliases, such as the machine’s IP address, short host name, and fully qualified host name. One of these aliases comprises the first part of the path for accessing a resource such as a servlet. For example, the alias `localhost:80` is used in the request `http://localhost:80/myServlet.`

A virtual host is not associated with a particular machine. It is a configuration, rather than a live object, explaining why you can create it, but cannot start or stop it. For many users, creating virtual hosts is unnecessary because the default_host is provided.

Adding a localhost to the virtual hosts adds the host name and IP address of the localhost machine to the alias table. This allows a remote user to access the administrative console.

You can use the administrative console to add or change DNS aliases if you want to use ports other than the default ports. If you do make a change to a DNS alias, you must regenerate the Web server plug-in configuration. You can use the administrative console to initiate the plug-in regeneration.

Note: You might want to add additional aliases or change the default aliases if:

- The HTTP server instance is running on a port other than 80. Add the correct port number to each of the aliases. For example, change `yourhost` to `yourhost:8000`.
- You want to make HTTPS requests, which use Secure Sockets Layer (SSL). To make HTTPS requests you must add port 443 to each of the aliases. Port 443 is the default port for SSL requests.
- Your Web server instance is listening for SSL requests on a port other than 443. In this situation, you must add that port number to each of the aliases.
- You want to use a port other then default port (9080) for the application server.
- You want to use other aliases that are not listed.

Why you would use virtual hosting

Virtual hosts let you manage a single application server on a single machine as if the application server were multiple application servers each on their own host machine. Resources associated with one virtual host cannot share data with resources associated with another virtual host. This is true even though the virtual hosts share the same application server on the same physical machine.

Virtual hosts isolate and independently manage multiple sets of resources on the same physical machine.

Suppose an Internet service provider (ISP) has two customers with Internet sites hosted on the same machine. The ISP keeps the two sites isolated from one another, despite their sharing a machine, by using virtual hosts. The ISP associates the resources of the first company with VirtualHost1 and the resources of the second company with VirtualHost2. Both virtual hosts map to the same application server.

Further suppose that both company sites offer the same servlet. Each site has its own instance of the servlet, and is unaware of the same servlet on the other site. If the company whose site is organized on VirtualHost2 is past due in paying its account with the ISP, the ISP can refuse all servlet requests that are routed to VirtualHost2. Even though the same servlet is available on VirtualHost1, the requests directed at VirtualHost2 do not go to the other virtual host.

The servlets on one virtual host do not share their context with the servlets on the other virtual host. Requests for the servlet on VirtualHost1 can continue as usual. This is true even though VirtualHost2 is refusing to fill requests for the servlet with the same name.

You associate a servlet or other application with a virtual host instead of the actual DNS address.

The default virtual host (default_host)

The product provides a default virtual host (named default_host).

The virtual host configuration uses wildcard entries with the ports for its virtual host entries.
- The default alias is *:80, using an internal port that is not secure.
- Aliases of the form *:9080 use the secure internal port.
- Aliases of the form *:9443 use the external port that is not secure.
- Aliases of the form *:443 use the secure external port.

Unless you specifically want to isolate resources from one another on the same physical machine, you probably do not need any virtual hosts in addition to the default host.

How requests map to virtual host aliases

Virtual hosts let you manage a single application server on a single machine as if the application server were multiple application servers that are each on their own host machine. Resources associated with one virtual host cannot share data with resources associated with another virtual host, even though the virtual hosts share the same application server on the same physical machine.

When you request a resource, WebSphere Application Server tries to map the request to an alias of a defined virtual host.

Mappings are both case sensitive and insensitive. For example, the portion `http://host:port/` is not case sensitive, but the URL that follows is case sensitive. The match must be alphanumerically exact. Also, different port numbers are treated as different aliases.

For example, the request `http://www.myhost.com/myservlet` maps successfully to `http://WWW.MYHOST.COM/myservlet` but not to `http://WWW.MYHOST.COM/MYSERVLET` or `Www.Myhost.Com/Myservlet`. In the latter two cases, these mappings fail because of case sensitivity. The request `http://www.myhost.com/myservlet` does not map successfully to `http://myhost/myservlet` or to `http://myhost:9876/myservlet`. These mappings fail because they are not alphanumerically correct.

You can use wildcard entries for aliases by port and specify that all valid host name and address combinations on a particular port map to a particular virtual host.

If you request a resource using an alias that cannot be mapped to an alias of a defined virtual host, you receive a 404 error in the browser that was used to issue the request. A message states that the virtual host could not be found.

Two sets of associations occur for virtual hosts. Application deployment associates an application with a virtual host. Virtual host definitions associate the network address of the machine and the HTTP transport or Web server port assignment of the application server with the virtual host. Looking at the flow from the Web client request for the snoop servlet, for example, the following actions occur:

1. The Web client asks for the snoop servlet: at Web address `http://www.some_host.some_company.com:9080/snoop`
2. The some_host machine has the 9080 port assigned to the standalone application server, server1.
3. server1 looks at the virtual host assignments to determine the virtual host that is assigned to the alias `some_host.some_company.com:9080`.
4. The application server finds that no explicit alias for that DNS string exists. However, a wild card assignment for host name * at port 9080 does exist. This is a match. The virtual host that defines the match is default_host.
5. The application server looks at the applications deployed on the default_host and finds the snoop servlet.
6. The application server serves the application to the Web client and the requester is able to use the snoop servlet.

You can have any number of aliases for a virtual host. You can even have overlapping aliases, such as:

Virtual host	Alias	Port
default_host	*	9080
	localhost	9080
	my_machine	9080
	my_machine.my_company.com	9080
	localhost	80

The Application Server looks for a match using the explicit address specified on the Web client address. However, it might resolve the match to any other alias that matches the pattern before matching the explicit address. Simply defining an alias first in the list of aliases does not guarantee the search order when WebSphere Application Server is looking for a matching alias.

A problem can occur if you use the same alias for two different virtual hosts. For example, assume that you installed the default application and the snoop servlet on the default_host. You also have another virtual host called the admin_host. However, you have not installed the default application or the snoop servlet on the admin_host.

Assume that you define overlapping aliases for both virtual hosts because you accidentally defined port 9080 for the admin_host instead of port 9060:

Virtual host	Alias	Port
default_host	*	9080
	localhost	9080
admin_host	*	9060
	my_machine.com	9080

Assume that a Web client request comes in for `http://my_machine.com:9080/snoop`.

If the application server matches the request against `*:9080`, the application is served from the default_host. If the application server matches the request to `my.machine.com:9080`, the application cannot be found. A 404 error occurs in the browser that issues the request. A message states that the virtual host could not be found.

This problem is the result of not finding the requested application in the first virtual host that has a matching alias. The correct way to code aliases is for the alias name on an incoming request to match only one virtual host in all of your virtual host definitions. If the URL can match more than one virtual host, you can see the problem just described.

Configuring virtual hosts

For configuration purposes, a virtual host enables WebSphere Application Server to treat multiple host machines or port numbers as a single logical host (virtual host). You can combine multiple host machines into a single virtual host or assign host machines to different virtual hosts, to separate and control which WebSphere Application Server resources are available for client requests.

If your external HTTP server configuration uses the default port, 9080, you do not have to perform these steps.

You must update the HTTP port numbers associated with the default virtual host. or define a new virtual host and associate it with the ports your HTTP server configuration uses if:

- Your external HTTP server configuration uses a port other than the default port of 9080, you must define the port that you are using.
- You are using the default HTTP port 9080, but the port is no longer defined. You must define port 9080.
- You have created multiple application servers as either stand-alone servers or cluster members, and these servers use the same virtual host. Because each server must be listening on a different port, you must define a virtual host alias for the HTTP port of each server.

If you define new virtual host aliases, identify the port values that the aliases use on the “Host alias settings” on page 191 page in the administrative console.

To create a new virtual host or change the configuration of an existing virtual host:

1. In the administrative console, click **Environment > Virtual Hosts**.
2. **Optional:** Create a new virtual host. If you create a new virtual host, a default set of 90 MIME entries are automatically created for that virtual host.
   a. In the administrative console, click **New**.

   b. Enter the name of the new virtual host and click **OK**. The new virtual host appears in the list of virtual hosts you can configure.
3. Select the virtual host whose configuration you want to change.
4. Under Additional Properties, click **Host Aliases**.
5. Create new host aliases or update existing host aliases to associate each of your HTTP port numbers with this virtual host.

   There must be a virtual host alias corresponding to each port your HTTP server configuration uses. There is one HTTP port associated with each Web container, and it is usually assigned to the virtual host named `default_host`. You can change the default assignment to any valid virtual host.

   The host aliases associated with the `default_host` virtual host are set to * when you install WebSphere Application Server. The * (an asterisk) indicates that the alias name does not have to be specified or that any name can be specified.

   When the URL for the application is entered into a Web browser, the port number is included. For example, if 9082 is the port number, the specified URL might look like the following:

   ```
   http://localhost:9082/wlm/SimpleServlet
   ```

   To create a new host alias:
   a. Click **New**.
   b. Specify a host alias name in the Host Name field and one of your HTTP ports in the Port field.

      You can specify * (an asterisk) for the alias name if you do not want to require the specification of the alias name or if you want to allow any name to be specified.
   c. Click **OK** and **Save** to save your configuration change.

   To update an existing host alias:
   a. Select an existing host alias name.
   b. Change the value specified in the Port field to one of your HTTP ports.
   c. Click **OK** and **Save** to save your configuration change.
6. **Optional:** Define a MIME object type and its file name extension if you require a MIME type other than the pre-defined types.
   a. For each needed MIME entry on the “MIME type collection” on page 192 page, click **New**.
   b. On the “MIME type settings” on page 192 page, specify a MIME type and extension.
   c. Click **OK** and **Save** to save your configuration change.
7. Regenerate the Web server plug-in configuration.
   a. **Click Servers > Web servers**, then select the appropriate Web server.
   b. Click **Generate Plug-in**, then click **Propagate Plug-in**.
8. Restart the application server.

Virtual host collection

Use this page to create and manage configurations that each let a single host machine resemble multiple host machines. Such configurations are known as *virtual hosts.*

To view this administrative console page, click **Environment > Virtual Hosts**.

Each virtual host has a logical name (which you define on this panel) and is known by its list of one or more domain name system (DNS) aliases. A DNS alias is the TCP/IP host name and port number used to request the servlet, for example yourHostName:80. (Port 80 is the default.)

You define one or more alias associations by clicking an existing virtual host or by adding a new virtual host.

When a servlet request is made, the server name and port number entered into the browser are compared to a list of all known aliases in an effort to locate the correct virtual host to serve the servlet. No match returns an error to the browser.

An application server profile provides a default virtual host with some common aliases, such as the internet protocol (IP) address, the DNS short host name, and the DNS fully qualified host name. The alias comprises the first part of the path for accessing a resource such as a servlet.

For example, the alias is localhost:80 in the request http://localhost:80/myServlet.

A virtual host is not associated with a particular profile or node (machine), but is associated with a particular server instead. It is a configuration, rather than a ″live object.″ You can create a virtual host, but you cannot start or stop it.

For many users, creating virtual hosts is unnecessary because the default_host that is provided is sufficient.

Adding the host name and IP address of the localhost machine to the alias table lets a remote user access the administrative console.

Resources associated with one virtual host cannot share data with resources associated with another virtual host, even if the virtual hosts share the same physical machine.

Name

Specifies a logical name for configuring Web applications to a particular host name. The default virtual host is suitable for most simple configurations.

Virtual hosts enable you to isolate, and independently manage, multiple sets of resources on the same physical machine. Determine whether you need a virtual host alias for each port associated with an HTTP transport channel or an HTTP transport. There must be a virtual host alias corresponding to each port used by an HTTP transport channel or an HTTP transport. There is one HTTP transport channel or HTTP transport associated with each Web container, and there is one Web container in each application server.

When you create a virtual host, a default set of 90 MIME entries is created for the virtual host.

You must create a virtual host for each HTTP port in the following cases:

- You use the internal HTTP transport with a port other than the default value of 9080, or for some reason the virtual host does not contain the usual entry for port 9080.
- You create multiple application servers (stand-alone servers, managed servers, or cluster members) that are using the same virtual host. Because each server must be listening on a different HTTP port, you need a virtual host alias for the HTTP port of each server.

Virtual host settings

Use this page to configure a virtual host instance.

To view this administrative console page, click **Environment > Virtual Hosts >** *virtual_host_name*.

Name:

Specifies a logical name for configuring Web applications to a particular host name. The default virtual host is suitable for most simple configurations.

Data type	String
Default	default_host

Host alias collection

Use this page to manage host name aliases defined for a virtual host. An alias is the DNS host name and port number that a client uses to form the URL request for a Web application resource.

To view this administrative console page, click **Environment** > **Virtual Hosts** > *virtual_host_name* > **Host Aliases**.

Host Name:

Specifies the IP address, DNS host name with domain name suffix, or just the DNS host name, used by a client to request a Web application resource (such as a servlet, JavaServer Pages (JSP) file, or HTML page). For example, the host alias name is `myhost` in a DNS name of `myhost:8080`.

The product provides a default virtual host (named default_host). The virtual host configuration uses the wildcard character * (asterisk) along with the port number for its virtual host entries. Unless you specifically want to isolate resources from one another on the same node (physical machine), you probably do not need any virtual hosts in addition to the default host.

Port:

Specifies the port for which the Web server has been configured to accept client requests. For example, the port assignment is *8080* in a DNS name of `myhost:8080`. A URL refers to this DNS as: http://*myhost:8080*/servlet/snoop.

Host alias settings:

Use this page to view and configure a host alias.

To view this administrative console page, click **Environment** > **Virtual Hosts** > *virtual_host_name* > **Host Aliases** > *host_alias_name*.

Host name:

Specifies the IP address, domain name system (DNS) host name with domain name suffix, or the DNS host name that clients use to request a Web application resource, such as a servlet, JSP file, or HTML page.

For example, when the DNS name is `myhost`, the host alias is `myhost:8080`, where *8080* is the port. A URL request can refer to the snoop servlet on the host alias as: http://*myhost:8080*/servlet/snoop.

When there is no port number specified for a host alias, the default port is 80. For existing virtual hosts, the default host name and port reflect the values specified at product installation or configuration. For new virtual hosts, the default can be `*` to allow any value or no specification.

Data type	String
Default	* You can also use the IP address or the long or short DNS name.

Port:

Specifies the port where the Web server accepts client requests. Specify a port value in conjunction with the host name.

The default reflects the value specified at product setup. The default might be `80`, `81`, `9060` or a similar value.

Data type	Integer
Default	9060

MIME type collection

Use this page to view and configure multi-purpose internet mail extensions (MIME) object types and their file name extensions.

The list shows a collection of MIME type extension mappings defined for the virtual host. Virtual host MIME entries apply when you do not specify MIME entries at the Web module level.

To view a list of current virtual host Mime types in the administrative console, click **Environment > Virtual Hosts >** *virtual_host_name* **> MIME Types**.

MIME Type:

Specifies a MIME type, which can be application, audio, image, text, video, www, or x-world. An example value for MIME type is `text/html`.

Extensions:

Specifies file extensions of files that map the MIME type. Do not specify the period before the extension. Example extensions for a `text/html` MIME type are `htm` and `html`.

MIME type settings:

Use this page to configure a multi-purpose internet mail extensions (MIME) object type.

To view this administrative console page, click **Environment > Virtual Hosts >** *virtual_host_name* **> MIME Types >** *MIME_type*.

MIME Type:

Specifies a MIME type, which can be application, audio, image, text, video, www, or x-world. An example value for MIME type is `text/html`.

An example value for MIME type is `text/html`. A default value appears only if you are viewing the configuration for an existing instance.

Data type	String

Extensions:

Specifies file extensions of files that map the MIME type. Do not specify the period before the extension. Example extensions for a `text/html` MIME type are `htm` and `html`.

File extensions for a `text/html` MIME type are `.htm` and `.html`. A default value appears only if you are viewing the configuration for an existing MIME type.

Data type	String

Variables

A variable is a configuration property that can be used to provide a parameter for some values in the system. A variable has a name and a value.

Not all WebSphere components support the use of a variable that you can define using this function. Test your application to verify that variables that you define are being used correctly.

WebSphere variables are used for:

- Configuring WebSphere Application Server path names, such as JAVA_HOME, and APP_INSTALL_ROOT.
- Configuring certain customization values.

Each variable has a scope. A scope is the range of locations in the WebSphere Application Server network where the variable is applicable.

- A variable with a node-level scope is available only on the node and the servers on that node. If a node-level variable has the same name as a cell-wide variable, the node-level variable value takes precedence.
- A server variable is available only on the one server process. A server variable takes precedence over a variable with the same name that is defined at a higher level.

You can use variables in configuration values such as file system path settings. Use the following syntax to refer to a variable:

```
${variable_name}
```

The value of a variable can contain a reference to another variable. The value of the variable is computed by substituting the value of the referenced variable recursively.

Variables are useful when concatenating two path variables when the specification does not accept the `AND` operator. For example, suppose that the following variables exist:

Variable name	Variable value
ROOT_DIR	`/`
HOME_DIR	`${ROOT_DIR}home`
USER_DIR	`${HOME_DIR}/myuserdir`

The variable reference `${USER_DIR}` resolves to the value `/home/myuserdir`.

Configuring WebSphere variables

This topic describes how to create a WebSphere Application Server variable.

You can define a WebSphere Application Server variable to provide a parameter for some values in the system. After you define the name and value for a variable, the value is used in place of the variable name. Variables most often specify file paths. However, some system components also support the use of variables. The Variable settings topic supplies further details about specifying variables and highlights further details about WebSphere Application Server components that use them.

WebSphere variables are used for:

- Configuring WebSphere Application Server path names, such as JAVA_HOME, and APP_INSTALL_ROOT.
- Configuring certain customization values.

The scope of a variable can be node-wide or applicable to only one server process.

Define variables on the **Environment > WebSphere Variables** console page.

Define the scope to apply a variable node-wide, or to only one server process. A variable resolves to its new value when used in a component that supports the use of variables.

1. Click **Environment > WebSphere Variables** in the administrative console to define a new variable.
2. Specify the scope of the variable.

   Declare the new variable for the **Node**, or **Server** and click **Apply**.

   The variable exists at the level you specify. Define a variable at multiple levels to use multiple values. The more granular definition overrides the higher level setting.

   For instance, if you specify the same variable on a node and a server, the server setting overrides the node setting.

   Scoping variables is particularly important when testing data source objects. Variable scoping can cause a data source to fail the test connection, but to succeed at run time, or to pass the test connection, but fail at run time.

   See the *Administering applications and their environment* PDF for more information.
3. Click **New** on the WebSphere Variables page.
4. Specify a name, a value, and a description on the Variable page. Click **OK**.
5. Verify that the variable is displayed in the list of variables. The administrative console does not pick up typing errors. The variable is ignored if it is referred to incorrectly.
6. Save your configuration.
7. Stop the server and start the server again to put the variable configuration change into effect.

WebSphere variables collection

Use this page to view and change a list of substitution variables with their values and scope.

To view this administrative console page, click **Environment > WebSphere Variables**.

For information on a variable, click the variable and read the value in the **Description** field.

Name

Specifies the symbolic name for a WebSphere Application Server variable. For example, a variable name might represent a physical path or URL root used by WebSphere Application Server.

Value

Specifies the value that the symbolic name represents. For example, the value might be an absolute path value for a file or URL root.

Scope

Specifies the level at which a WebSphere Application Server variable is visible on the administrative console panel.

A resource can be visible in the administrative console collection table at the node or server scope.

Variable settings

Use this page to define the name and value of a WebSphere Application Server substitution variable.

To view this administrative console page, click **Environment > WebSphere Variables >** *WebSphere_variable_name*.

Name:

Specifies the symbolic name for a WebSphere Application Server variable. For example, a variable name might represent a physical path or URL root that is used by WebSphere Application Server.

WebSphere Application Server variables are used for:

- Configuring WebSphere Application Server path names, such as *JAVA_HOME*, and APP_INSTALL_ROOT.
- Configuring certain customization values.

WebSphere Application Server substitutes the symbolic name wherever its value displays in the system.

For example, *JAVA_HOME* is the symbolic name representing the file system path to the installation directory for the Java virtual machine (JVM). For example, the value is /opt/IBM/WebSphere/AppServer/java for the WebSphere Application Server product on a Linux machine.

You can create new variables for use in WebSphere Application Server components that support the use of variables.

The WebSphere Application Server security component supports variables when establishing administrative security, but only the following:

- APP_INSTALL_ROOT
- WAS_INSTALL_ROOT
- USER_INSTALL_ROOT
- WAS_TEMP_DIR
- WAS_PROPS_DIR
- WAS_ETC_DIR

The security component uses these variables as security defaults when making substitutions that identify a path to the security configuration settings.

Data type	String

Value:

Specifies the value that the symbolic name represents. For example, the value might be an absolute path value for a file or URL root.

For example, `/opt/IBM/WebSphere/AppServer/java` is the value on a Linux machine for a variable named JAVA_HOME.

Data type	String

Description:

Documents the purpose of a variable.

Data type	String

Variables

Variables in the WebSphere Application Server environment come in many varieties. They are used to control settings and properties relating to the server environment. Three main variable options that are important for a WebSphere Application Server user to know and understand are environment variables, and WebSphere Application Server variables, and custom properties.

Environment variables. Environment variables, also called *native environment variables*, are not specific to WebSphere Application Server and are defined by other elements, such as UNIX, Language

Environment (LE), or third-party vendors, among others. Some of the UNIX-specific native variables are LIBPATH and STEPLIB. These variables tend to be operating system-specific.

WebSphere Application Server variables

WebSphere Application Server variables are used for three purposes:

- Configuring WebSphere Application Server path names, such as JAVA_HOME, and APP_INSTALL_ROOT
- Configuring certain cell-wide customization values

WebSphere Application Server variables are specified in the administrative console by clicking **Environment > Manage WebSphere variables**. How the variable is set determines its scope.

A variable can apply to a node or a server.

If the variable is set:

- At the server level, it applies to the entire server.
- At the node level, it applies to all servers in the node, unless you set the same variable at the server level. In that case, for that server, the setting that is specified at the server level overrides the setting that is specified at the node level.

Custom properties

Custom properties are property settings meant for a specific functional component. Any configuration element can have a custom property. Common configuration elements are cell, node, server, Web container, and transaction service. A limited number of supported custom properties are available and these properties can be set in the administrative console using the custom properties link that is associated with the functional component.

For example, to set Web container custom properties, click **Servers > Application Servers >** *server_name* **> Web Container Settings > Web container > Custom Properties**

Custom properties set from the Web container custom properties page apply to all transports that are associated with that Web container; custom properties set from one of the Web container transport chain or HTTP transport custom properties pages apply only to that specific HTTP transport chain or HTTP transport. If the same property is set on both the Web container page and either a transport chain or HTTP transport page, the settings on the transport chain or HTTP transport page override the settings that are defined for the Web container for that specific transport.

IBM Toolbox for Java JDBC driver

WebSphere Application Server supports the **IBM Toolbox for Java** JDBC driver. The IBM Toolbox for Java JDBC driver is included with the IBM Toolbox for Java product.

IBM Toolbox for Java is a library of Java classes that are optimized for accessing iSeries and AS/400 data and resources. You can use the IBM Toolbox for Java JDBC driver to access local or remote **DB2 UDB for iSeries** databases from server-side and client Java applications that run on any platform that supports Java.

IBM Toolbox for Java is available in these versions:

IBM Toolbox for Java licensed program

The licensed program is available with every OS/400 release, starting with Version 4 Release 2 (V4R2). You can install the licensed program on your iSeries system, and then either copy the IBM Toolbox for Java JAR file (*jt400.jar*) to your system or update your system *classpath* to locate the server installation. Product documentation for IBM Toolbox for Java is available from the iSeries information center: http://publib.boulder.ibm.com/infocenter/iseries/v5r3/ic2924/index.htm Locate the documentation by traversing the following path in the left-hand navigation window of the iSeries information center: **Programming** > **Java** > **IBM Toolbox for Java**.

JTOpen

JTOpen is the open source version of IBM Toolbox for Java, and is more frequently updated than the licensed program version. You can download JTOpen from http://www-1.ibm.com/servers/eserver/iseries/toolbox/downloads.htm. You can also download the *JTOpen Programming Guide*. The guide includes instructions for installing JTOpen and information about the JDBC driver.

The JDBC driver for both versions supports JDBC 3.0. For more information about IBM Toolbox for Java and JTOpen, see the product Web site at http://www-1.ibm.com/servers/eserver/iseries/toolbox/index.html.

Note: If you are using WebSphere Application Server on platforms other than iSeries, use the **JTOpen** version of the Toolbox JDBC driver.

Configure and use the jt400.jar file

The following steps describe how to configure and utilize the jt400.jar file.

1. Download the *jt400.jar* file from the **JTOpen** URL at http://www-1.ibm.com/servers/eserver/iseries/toolbox/downloads.htm.

   Place it in a directory on your workstation such as */JDBC_Drivers/Toolbox*.
2. Open the administrative console.
3. Select **Environment > WebSphere Variables**.
4. Set the WebSphere variable *OS400_TOOLBOX_JDBC_DRIVER_PATH* at the **Node** level.
5. Double click **OS400_TOOLBOX_JDBC_DRIVER_PATH**.
6. Set the value to the full directory path to the `jt400.jar` file downloaded in step one. Do not include *jt400.jar* in this value.

   For example:

   ```
   OS400_TOOLBOX_JDBC_DRIVER_PATH == "/JDBC_Drivers/Toolbox"
   ```

   When you choose a Toolbox driver from the list of possible resource providers the **Classpath** field looks like:

   ```
   Classpath == ${OS400_TOOLBOX_JDBC_DRIVER_PATH}/jt400.jar
   ```

Shared library files

Shared library files in WebSphere Application Server consist of a symbolic name, a Java class path, and a native path for loading Java Native Interface (JNI) libraries.

You can define a shared library at the cell, node, or server level. Defining a library at one of the three levels does not cause the library to be placed into the application server's class loader. You must associate the library to an application or server in order for the classes represented by the shared library to be loaded in either a server-wide or application-specific class loader.

A separate class loader is used for shared libraries that are associated with an application server. This class loader is the parent of the application class loader, and the WebSphere Application Server extensions class loader is its parent. Shared libraries that are associated with an application are loaded by the application class loader.

Managing shared libraries

Shared libraries are files used by multiple applications. Using the administrative console, you can define a shared library at the cell, node, or server level. You can then associate the library to an application, module or server to load the classes represented by the shared library in either a server-wide or application-specific class loader. Using an installed optional package, you can associate a shared library to an application by declaring the dependent library `.jar` file in the `MANIFEST.MF` file of the application. Refer to the Java 2 Platform, Enterprise Edition (J2EE) 1.4 specification, section 8.2 for an example.

If your deployed applications use shared library files, define shared libraries for the library files and associate the libraries with specific applications or modules or with an application server. Associating a shared library file with a server associates the file with all applications on the server. Use the Shared Libraries page to define new shared library files to the system and remove them.

- Use the administrative console to define a shared library.
  1. Create a shared library for each library file that your applications need.
  2. Associate each shared library with an application or module.
     - Associate a shared library with an application or module that uses the shared library file.
     - Associate a shared library with an application server so every application on the server can use the shared library file.
- Use an installed optional package to declare a shared library for an application.
- Remove a shared library.
  1. Click **Environment > Shared Libraries** in the console navigation tree to access the Shared Libraries page.
  2. Select the library to be removed.
  3. Click **Delete**.

  The list of shared libraries is refreshed. The library file no longer displays in the list.

Creating shared libraries

Shared libraries are files used by multiple applications.

The first step for making a library file available to multiple applications deployed on a server is to create a shared library for each library file that your applications need. When you create the shared libraries, set variables for the library files.

Use the Shared Libraries page to create and configure shared libraries.

1. Go to the Shared Libraries page.

   Click **Environment > Shared Libraries** in the console navigation tree.
2. Change the scope of the collection table to see what shared libraries are in a cell, node, server, or cluster.
   a. Select a cell, node, server, or cluster.
   b. Click **Apply**.
3. Click **New**.
4. Configure the shared library.
   a. On the settings page for a shared library, specify the name, class path, and any other variables for the library file that are needed.

      If the shared library specifies a native library path, refer to “Configuring native libraries in shared libraries” on page 199.
   b. Click **Apply**.
5. Repeat steps 1 through 4 until you define a shared library instance for each library file that your applications need.

Using the administrative console, associate your shared libraries with specific applications or modules or with the class loader of an application server. Associating a shared library file with a server class loader associates the file with all applications on the server.

Alternatively, you can use an installed optional package to associate your shared libraries with an application.

Configuring native libraries in shared libraries

Native libraries are platform-specific library files, including `.dll`, `.so`, or *SRVPGM objects, that can be configured within shared libraries. Native libraries are visible to an application class loader whenever the shared library is associated with an application. Similarly, native libraries are visible to an application server class loader whenever the shared library is associated with an application server.

When designing a shared library, consider the following conditions regarding Java native library support:

- The Java virtual machine (JVM) allows only one class loader to load a particular native library.
- There is no application programming interface (API) to unload a native library from a class loader.

  Native libraries are unloaded by the JVM when the class loader that found the library is collected from the heap during garbage collection.
- Application server class loaders persist for the duration of the application server.
- Application class loaders persist until an application is stopped or dynamically reloaded.

  If a shared library that is configured with a native library path is associated with an application, whenever the application is restarted or dynamically reloaded the application might fail with an UnsatisfiedLinkError indicating that the library is already loaded. The error occurs because, when the application restarts, it invokes the shared library class to reload the native library. The native library, however, is still loaded in memory because the application class loader which previously loaded the native library has not yet been garbage collected.
- Only the JVM class loader can load a dependent native library.

  For example, if *NativeLib1* is dependent on *NativeLib2*, then *NativeLib2* must be visible to the JVM class loader. The path containing *NativeLib2* must be specified on Java library path defined by the LIBPATH environment variable. If a native library configured in a shared library is dependent on other native libraries, the dependent libraries must be configured on the LIBPATH of the JVM hosting the application server in order for that library to load successfully.

When configuring a shared library on a shared library settings page, if you specify a value for **Native Library Path**, the native libraries on this path are not located by the WebSphere Application Server application or shared library class loaders unless the class which loads the native library was itself loaded by the same class loader.

Because a native library cannot be loaded more than once by a class loader, it is preferable for native libraries to be loaded within shared libraries associated with the class loader of an application server, because these class loaders persist for the lifetime of the server.

1. Implement a static method in the class that loads the native library.

   In the class that loads the native library, call `System.loadLibrary(`*`native_library`*`)` in a static block. For example:

   ```
   static {System.loadLibrary("native_library");
   ```

   native_library loads during the static initialization of the class, which occurs exactly once when the class loads.
2. On the shared library settings page, set values for **Classpath** and **Native Library Path** that enable the shared library to load the native library.
3. Associate the shared library with the class loader of an application server.

   Associating a shared library with the class loader of an application server, rather than with an application, ensures that the shared library is loaded exactly once by the application server class loader, even though applications on the server are restarted or dynamically reloaded. Because the native library is loaded within a static block, the native library is never loaded more than once.

Shared library collection

Use this page to define a list of shared library files that deployed applications can use.

To view this administrative console page, click **Environment** > **Shared Libraries**.

Create a shared library for each library file that your application needs:

1. See what shared libraries are in a cell, node, or server.

   By default, a shared library is accessible to applications deployed (or installed) on the same node as the shared library file. Use the **Scope** field to change the scope to a different node or to a specific server. Under **Scope**, select the cell, a node, or a server and click **Apply**. Changing the scope of the collection table enables you to find shared libraries.
2. Select the scope for your new shared library.
3. Click **New**.
4. On the settings page for the new shared library, specify the name, class path, and any other variables for the library file that are needed.

After you create a shared library, associate it with an application or module or with the class loader of a server:

- To associate a shared library with an application or module, click **Applications** > **Enterprise Applications** > *application_name* > **Shared library references**. On the Shared library references page for the application, select the shared library file and click **OK**.
- To associate a shared library with a server class loader, click **Servers** > **Application servers** > *server_name* > **Java and Process Management** > **Class loader** > *class_loader_ID* > **Shared library references** > *shared_library_name*. On the settings page for the library reference for the server class loader, specify values that identify the shared library file.

Name

Specifies a name for the shared library.

Description

Describes the shared library file.

Shared library settings

Use this page to make a library file available to deployed applications.

To view this administrative console page, click **Environment** > **Shared Libraries** > *shared_library_name*.

Scope:

Specifies whether the shared library has its configuration file in a location that pertains to the cell, node, server, or cluster level.

Data type	String

Name:

Specifies a name for the shared library.

Data type	String

Description:

Describes the shared library file.

Data type	String

Classpath:

Specifies the class path used to locate the JAR files for the shared library support.

Data type	String
Units	Class path

Native Library Path:

Specifies the class path for locating platform-specific library files for shared library support; for example, `.dll`, `.so`, or *SRVPGM objects.

If you specify a value for **Native Library Path**, the native libraries are not located by the WebSphere Application Server application or shared library class loaders unless the following conditions exist:

- A class loads the native libraries.
- The application invokes a method in this class which loads the libraries.

  For example, in the class that loads the native library, call `System.loadLibrary(`*`native_library`*`)` in a static block:

  ```
  static {System.loadLibrary("native_library");
  ```
- The **Classpath** specified on this page contains the class that loads the libraries.

Native libraries cannot be loaded more than once by a class loader. Thus, it is preferable for native libraries to be loaded within shared libraries associated with the class loader of an application server.

Data type	String
Units	Class path

Associating shared libraries with applications or modules

You can associate a shared library with an application or module. Classes represented by the shared library are then loaded in the application's class loader, making the classes available to the application.

This topic assumes that you have defined a shared library at the cell, node, server, or cluster level. The shared library represents a library file used by multiple deployed applications.

This topic also assumes that you want to use the administrative console, and not an installed optional package, to associate a shared library with an application.

To associate a shared library with an application or module, create and configure a library reference using the administrative console. A library reference specifies the name of the shared library file.

If you associate a shared library with an application, do not associate the same shared library with a server class loader.

1. Click **Applications** > **Enterprise Applications** > *application_name* > **Shared library references** in the console navigation tree to access the Shared library references page.
2. On the Shared library references page, select an application or module to which you want to associate a shared library.
3. Click **Reference shared libraries**.
4. On the Shared Library Mapping page, select one or more shared libraries that the application or modules uses in the **Available** list, click **>>** to add them to the **Selected** list, and click **OK**.

5. Repeat steps 2 through 4 until you define a library reference instance for each shared library that your application or module requires.
6. On the Shared library references page, click **OK**.

When you run the application, classes represented by the shared library are loaded in the application's class loader, making the classes available to the application or module.

Shared library reference and mapping settings

Use the Shared library references and Shared Library Mapping pages to associate defined shared libraries with an application or Web module. A shared library is an external Java archive (JAR) file that is used by one or more applications. Using shared libraries enables multiple applications deployed on a server to use a single library, rather than use multiple copies of the same library. After you associate shared libraries with an application or module, the application or module class loader loads classes represented by the shared libraries and makes those classes available to the application or module.

To view the Shared library references console page, click **Applications > Enterprise Applications >** *application_name* **> Shared library references**. To view the Shared Library Mapping page, click **Reference shared libraries** on the Shared library references page. These pages are the same as the Shared Library Mapping for Modules and Shared Library Mapping pages in the application installation and update wizards.

On the Shared library references page, the first element listed is the application. The other elements are modules in the application.

To associate shared libraries with your application or module:

1. Select an application or module.
2. Click **Reference shared libraries**.
3. On the Shared Library Mapping page, select one or more shared libraries that the application or modules uses in the **Available** list, click **>>** to add them to the **Selected** list, and click **OK**.

A defined shared library for a file that your application or module uses must exist to associate your application or module to the library.

If no shared libraries are defined and the application is installed already, on the Shared Library Mapping page, click **New** and define a shared library.

You can otherwise define a shared library as follows:

1. Click **Environment > Shared Libraries**.
2. Specify whether the shared library is visible at the cell, node or server level.
3. Click **New**.
4. On the settings page for the new shared library, specify a name and one or more class paths. If the libraries are platform-specific files such as `.dll`, `.so`, or *SRVPGM objects, also specify a native library path. Then, click **Apply**.
5. Save the administrative configuration.

Application:

Specifies the name of the application that you are installing or that you selected on the Enterprise Applications page.

Module:

Specifies the name of the module associated with the shared libraries.

URI:

Specifies the location of the module relative to the root of the application EAR file.

Shared libraries:

Specifies the name of the shared library files associated with the application or module.

Associating shared libraries with servers

You can associate shared libraries with the class loader of a server. Classes represented by the shared library are then loaded in a server-wide class loader, making the classes available to all applications deployed on the server.

This topic assumes that you have defined a shared library at the cell, node, server, or cluster level. The shared library represents a library file used by multiple deployed applications.

To associate a shared library with the class loader of a server, create and configure a library reference using the administrative console. A library reference specifies the name of the shared library file.

If you associate a shared library with a server class loader, do not associate the same shared library with an application.

1. Configure class loaders for applications deployed on the server.
   a. Click **Servers > Application Servers > *server_name*** to access the settings page for the application server.
   b. Set values for the application **Class loader policy** and **Class loading mode** of the server.

      For information on these settings, see “Application server settings” on page 220 and the *Administering applications and their environment* PDF.
2. Create a library reference for each shared library file that your application needs.
   a. In the administrative console, click **Servers > Application servers >** *server_name* **> Java and Process Management > Class loader >** *class_loader_ID*.
   b. Click **Shared library references** to access the Library Reference page.
   c. Click **Add**.
   d. On the settings page for a library reference, name the library reference. The name identifies the shared library file that your application uses.
   e. Click **Apply**. The name of the library reference is shown in the list on the Library Reference page.

   Repeat the previous steps until you define a library reference for each shared library that your application needs.

Installed optional packages

Installed optional packages enable applications to use the classes in Java archive (`.jar`) files without having to include them explicitly in a class path. An installed optional package is a `.jar` file containing specialized tags in its manifest file that enable the application server to identify it. An installed optional package declares one or more shared library `.jar` files in the manifest file of an application. When the application is installed on a server, the classes represented by the shared libraries are loaded in the class loader of the application, making the classes available to the application.

When a Java 2 Platform, Enterprise Edition (J2EE) application is installed on a server, dependency information is specified in its manifest file. WebSphere Application Server reads the dependency information of the application (`.ear` file) to automatically associate the application with an installed optional package `.jar` file. WebSphere Application Server adds the `.jar` files in associated optional packages to the application class path. Classes in the installed optional packages are then available to application classes.

Installed optional packages used by WebSphere Application Server are described in section 8.2 of the J2EE specification, Version 1.4 at http://java.sun.com/j2ee/j2ee-1_4-fr-spec.pdf.

WebSphere Application Server supports using the manifest file (`manifest.mf`) in shared library `.jar` files and application `.ear` files. WebSphere Application Server does not support the Java 2 Platform Standard Edition (J2SE) Installed Optional Package semantics used in the J2SE specification (http://java.sun.com/j2se/1.3/docs/guide/extensions/spec.html), which primarily serve the applet environment. WebSphere Application Server ignores applet-specific tags within manifest files.

Sample manifest.mf file

A sample manifest file follows for an application `app1.ear` that refers to a single shared library file `util.jar`:

```
app1.ear:
    META-INF/application.xml
    ejb1.jar:
         META-INF/MANIFEST.MF:
             Extension-List: util
             util-Extension-Name: com/example/util
             util-Specification-Version: 1.4
         META-INF/ejb-jar.xml

util.jar:
    META-INF/MANIFEST.MF:
        Extension-Name: com/example/util
        Specification-Title: example.com's util package
        Specification-Version: 1.4
        Specification-Vendor: example.com
        Implementation-Version: build96
```

The syntax of a manifest entry depends on whether the entry applies to a member with a defining role (the shared library) or a member with a referencing role (a J2EE application or a module within a J2EE application).

Manifest entry tagging

Main tags used for manifest entries include the following:

Extension-List
A required tag with variable syntax. Within the context of the referencing role (application's manifest), this is a space delimited list that identifies and constructs unique Extension-Name, Extension-Specification tags for each element in the list. Within the context of the defining role (shared library), this tag is not valid.

Extension-Name
A required tag that provides a name and links the defining and referencing members. The syntax of the element within the referencing role is to prefix the element with the `<ListElement>` string. For each element in the Extension-List, there is a corresponding `<ListElement>`-Extension-Name tag. The defining string literal value for this tag (in the above sample `com/example/util1`) is used to match (in an equality test) the corresponding tags between the defining and referencing roles.

Specification-Version
A required tag that identifies the specification version and links the defining and referencing members.

Implementation-Version
An optional tag that identifies the implementation version and links the defining and referencing members.

Further information on these tags is in the .jar file specification at http://java.sun.com/j2se/1.4.2/docs/guide/jar/jar.html#Manifest%20Specification.

Using installed optional packages

You can associate one or more shared libraries with an application using an installed optional package that declares the shared libraries in the application's manifest file. Classes represented by the shared libraries are then loaded in the application's class loader, making the classes available to the application.

Read about installed optional packages in "Installed optional packages" on page 203 and in section 8.2 of the Java 2 Platform, Enterprise Edition (J2EE) specification, Version 1.4 at http://java.sun.com/j2ee/j2ee-1_4-fr-spec.pdf.

WebSphere Application Server does not support the Java 2 Platform Standard Edition (J2SE) Installed Optional Package semantics used in the J2SE specification (http://java.sun.com/j2se/1.3/docs/guide/extensions/spec.html), which primarily serve the applet environment. WebSphere Application Server ignores applet-specific tags within manifest files.

Installed optional packages expand the existing shared library capabilities of an application server. Prior to Version 6, an administrator was required to associate a shared library to an application or server. Installed optional packages enable an administrator to declare a dependency in an application's manifest file to a shared library, with installed optional package elements listed in the manifest file, and automatically associate the application to the shared library. During application installation, the shared library .jar file is added to the class path of the application class loader.

If you use an installed optional package to associate a shared library with an application, do not associate the same shared library with an application class loader or a server class loader using the administrative console.

1. Assemble the library file, including the manifest information that identifies it as an extension. Two sample manifest files follow. The first sample manifest file has application app1.ear refer to a single shared library file util.jar:

   ```
   app1.ear:
       META-INF/application.xml
       ejb1.jar:
            META-INF/MANIFEST.MF:
                Extension-List: util
                util-Extension-Name: com/example/util
                util-Specification-Version: 1.4
            META-INF/ejb-jar.xml

   util.jar:
       META-INF/MANIFEST.MF:
           Extension-Name: com/example/util
           Specification-Title: example.com's util package
           Specification-Version: 1.4
           Specification-Vendor: example.com
           Implementation-Version: build96
   ```

   The second sample manifest file has application app1.ear refer to multiple shared library .jar files:

   ```
   app1.ear:
       META-INF/application.xml
       ejb1.jar:
            META-INF/MANIFEST.MF:
                Extension-List: util1 util2 util3
                Util1-Extension-Name: com/example/util1
                Util1-Specification-Version: 1.4
                Util2-Extension-Name: com/example/util2
                Util2-Specification-Version: 1.4
                Util3-Extension-Name: com/example/util3
                Util3-Specification-Version: 1.4
   ```

```
         META-INF/ejb-jar.xml

util1.jar:
    META-INF/MANIFEST.MF:
        Extension-Name: com/example/util1
        Specification-Title: example.com's util package
        Specification-Version: 1.4
        Specification-Vendor: example.com
        Implementation-Version: build96

util2.jar:
    META-INF/MANIFEST.MF:
        Extension-Name: com/example/util2
        Specification-Title: example.com's util package
        Specification-Version: 1.4
        Specification-Vendor: example.com
        Implementation-Version: build96

util3.jar:
    META-INF/MANIFEST.MF:
        Extension-Name: com/example/util3
        Specification-Title: example.com's util package
        Specification-Version: 1.4
        Specification-Vendor: example.com
        Implementation-Version: build96
```

2. Create a shared library that represents the library file assembled in step 1. This installs the library file as a WebSphere Application Server shared library.
3. Assemble the application, declaring in the application manifest file dependencies to the library files named the manifest created for step 1.

   See the *Developing and deploying applications* PDF for more information.
4. Install the application on the server.

   See the *Developing and deploying applications* PDF for more information.

During application installation, the shared library `.jar` files are added to the class path of the application class loader.

Library reference collection

Use this page to view and manage library references that define how to use global libraries. For example, you can use this page to associate shared library files with a deployed application.

To view this administrative console page, click **Servers** > **Application servers** > *server_name* > **Java and Process Management** > **Class loader** > *class_loader_ID* > **Shared library references**.

If no shared libraries are defined in your environment, such as at the node or server scope, after you click **Add** a message is displayed stating that you must define a shared library before you can create a library reference. A shared library is a container-wide library file that can be used by deployed applications. To define a shared library, click **Environment** > **Shared Libraries** and specify the scope of the container. Then, click **New** and specify a name and one or more paths for the shared library. After you define a shared library, return to this page, click **Add**, and create a library reference.

Library name

Specifies a name for the library reference.

Library reference settings

Use this page to define library references, which specify how to use global libraries.

To view this administrative console page, click **Servers > Application servers >** *server_name* **> Java and Process Management > Class loader >** *class_loader_ID* **> Shared library references >** *library_reference_name*.

A shared library is a container-wide library file that can be used by deployed applications. To define a shared library, click **Environment > Shared Libraries** and specify the scope of the container. Then, click **New** and specify a name and one or more paths for the shared library.

Library name:

Specifies the name of the shared library to use for the library reference.

Data type	String

Environment: Resources for learning

Use the following links to find relevant supplemental information about configuring the WebSphere Application Server environment. 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.

Programming instructions and examples

- WebSphere Application Server education

Administration

- Listing of all IBM WebSphere Application Server Redbooks

Chapter 8. Working with server configuration files

This topic show how to manage application server configuration files.

Application server configuration files define the available application servers, their configurations, and their contents.

You should periodically save changes to your administrative configuration. You can change the default locations of configuration files, as needed.

- Edit configuration files.

  The master repository is comprised of .xml configuration files

  You can edit configuration files using
  - The administrative console. See the Using the administrative console topic in the *Using the administrative clients* PDF.
  - Scripting. See the Getting started with scripting topic in the *Using the administrative clients* PDF.
  - The wsadmin commands. See the Using command line tools topic in the *Using the administrative clients* PDF.
  - Programing. See the Using administrative programs (JMX) topic in the *Using the administrative clients* PDF.
  - By editing a configuration file directly.
- Save changes made to configuration files. Using the console, you can save changes as follows:
  1. In the navigation select **System Administration > Save changes to master repository**.
  2. Click **Save**.
- Handle temporary configuration files resulting from a session timing out.
- Change the location of temporary configuration files.
- Change the location of backed-up configuration files.
- Change the location of temporary workspace files.
- Back up and restore configurations.

Configuration documents

WebSphere Application Server stores configuration data for servers in several documents in a cascading hierarchy of directories. The configuration documents describe the available application servers, their configurations, and their contents. Most configuration documents have XML content.

Hierarchy of directories of documents

The cascading hierarchy of directories and the documents' structure support multinode replication to synchronize the activities of all servers in a cell. In a Network Deployment environment, changes made to configuration documents in the cell repository, are automatically replicated to the same configuration documents that are stored on nodes throughout the cell.

At the top of the hierarchy is the **cells** directory. It holds a subdirectory for each cell. The names of the cell subdirectories match the names of the cells. For example, a cell named *cell1* has its configuration documents in the subdirectory *cell1*. The name of the cell must be different from the cluster name pair.

On the Network Deployment node, the subdirectories under the cell contain the entire set of documents for every node and server throughout the cell. On other nodes, the set of documents is limited to what applies to that specific node. If a configuration document only applies to *node1*, then that document exists in the configuration on *node1* and in the Network Deployment configuration, but not on any other node in the cell.

Each cell subdirectory has the following files and subdirectories:

- The cell.xml file, which provides configuration data for the cell
- Files such as security.xml, virtualhosts.xml, resources.xml, and variables.xml, which provide configuration data that applies across every node in the cell
- The **clusters** subdirectory, which holds a subdirectory for each cluster defined in the cell. The names of the subdirectories under clusters match the names of the clusters.

  Each cluster subdirectory holds a cluster.xml file, which provides configuration data specifically for that cluster.
- The **nodes** subdirectory, which holds a subdirectory for each node in the cell. The names of the nodes subdirectories match the names of the nodes.

  Each node subdirectory holds files such as variables.xml and resources.xml, which provide configuration data that applies across the node. Note that these files have the same name as those in the containing cell's directory. The configurations specified in these node documents override the configurations specified in cell documents having the same name. For example, if a particular variable is in both cell- and node-level variables.xml files, all servers on the node use the variable definition in the node document and ignore the definition in the cell document.

  Each node subdirectory holds a subdirectory for each server defined on the node. The names of the subdirectories match the names of the servers. Each server subdirectory holds a server.xml file, which provides configuration data specific to that server. Server subdirectories might hold files such as security.xml, resources.xml and variables.xml, which provide configuration data that applies only to the server. The configurations specified in these server documents override the configurations specified in containing cell and node documents having the same name.
- The **applications** subdirectory, which holds a subdirectory for each application deployed in the cell. The names of the applications subdirectories match the names of the deployed applications.

  Each deployed application subdirectory holds a deployment.xml file that contains configuration data on the application deployment. Each subdirectory also holds a **META-INF** subdirectory that holds a J2EE application deployment descriptor file as well as IBM deployment extensions files and bindings files. Deployed application subdirectories also hold subdirectories for all .war and entity bean .jar files in the application. Binary files such as .jar files are also part of the configuration structure.

An example file structure is as follows:

```
cells
  cell1
     cell.xml resources.xml virtualhosts.xml variables.xml security.xml
     nodes
        nodeX
           node.xml variables.xml resources.xml serverindex.xml
           serverA
              server.xml variables.xml
           nodeAgent
              server.xml variables.xml
        nodeY
           node.xml variables.xml resources.xml serverindex.xml
     applications
        sampleApp1
           deployment.xml
           META-INF
              application.xml ibm-application-ext.xml ibm-application-bnd.xml
        sampleApp2
           deployment.xml
           META-INF
              application.xml ibm-application-ext.xml ibm-application-bnd.xml
```

Changing configuration documents

You can use one of the administrative tools (console, wsadmin, Java APIs) to modify configuration documents or edit them directly. It is preferable to use the administrative console because it validates changes made to configurations. ″“Configuration document descriptions”″ states whether you can edit a document using the administrative tools or must edit it directly.

Configuration document descriptions

Most configuration documents have XML content. The table describes the documents and states whether you can edit them using an administrative tool or must edit them directly.

If possible, edit a configuration document using the administrative console because it validates any changes that you make to configurations. You can also use one of the other administrative tools (wsadmin or Java APIs) to modify configuration documents. Using the administrative console or wsadmin scripting to update configurations is less error prone and likely quicker and easier than other methods.

However, you cannot edit some files using the administrative tools. Configuration files that you must edit manually have an X in the **Manual editing required** column in the table below.

Document descriptions

(The paths in the Locations column are split on multiple lines for publishing purposes.)

Configuration file	Locations	Purpose	Manual editing required

`admin-authz.xml`	`config/cells/`
	`cell_name``/`
		Define a role for administrative operation authorization.	X

`app.policy`	`config/cells/`
	`cell_name``/`
	`nodes/`*`node_name`*`/`
		Define security permissions for application code.	X

`cell.xml`	`config/cells/`
	`cell_name``/`
		Identify a cell.	

`cluster.xml`	`config/cells/`
	`cell_name``/`
	`clusters/`
	`cluster_name``/`
		Identify a cluster and its members and weights.
		This file is only available with the Network Deployment product.	

`deployment.xml`	`config/cells/`
	`cell_name``/`
	`applications/`
	`application_name``/`
		Configure application deployment settings such as target servers and application-specific server configuration.	

`filter.policy`	`config/cells/`
	`cell_name``/`
		Specify security permissions to be filtered out of other policy files.	X

`integral-jms-authorizations.xml`	`config/cells/`
	`cell_name``/`
		Provide security configuration data for the integrated messaging system.	X

`library.policy`	`config/cells/`
	`cell_name``/`
	`nodes/`*`node_name`*`/`
		Define security permissions for shared library code.	X

`multibroker.xml`	`config/cells/`
	`cell_name``/`
		Configure a data replication message broker.	

`namestore.xml`	`config/cells/`
	`cell_name``/`
		Provide persistent name binding data.	X

naming-authz.xml	config/cells/*cell_name*/	Define roles for a naming operation authorization.	X
node.xml	config/cells/*cell_name*/nodes/*node_name*/	Identify a node.	
pmirm.xml	config/cells/*cell_name*/	Configure PMI request metrics.	X
resources.xml	config/cells/*cell_name*/ config/cells/*cell_name*/nodes/*node_name*/ config/cells/*cell_name*/nodes/*node_name*/servers/*server_name*/	Define operating environment resources, including JDBC, JMS, JavaMail, URL, JCA resource providers and factories.	
security.xml	config/cells/*cell_name*/	Configure security, including all user ID and password data.	
server.xml	config/cells/*cell_name*/nodes/*node_name*/servers/*server_name*/	Identify a server and its components.	
serverindex.xml	config/cells/*cell_name*/nodes/*node_name*/	Specify communication ports used on a specific node.	
spi.policy	config/cells/*cell_name*/nodes/*node_name*/	Define security permissions for service provider libraries such as resource providers.	X
variables.xml	config/cells/*cell_name*/ config/cells/*cell_name*/nodes/*node_name*/ config/cells/*cell_name*/nodes/*node_name*/servers/*server_name*/	Configure variables used to parameterize any part of the configuration settings.	
virtualhosts.xml	config/cells/*cell_name*/	Configure a virtual host and its MIME types.	

Object names: What the name string cannot contain

When you create a new object using the administrative console or a wsadmin command, you often must specify a string for a name attribute.

Most characters are allowed in the name string. However, the name string cannot contain the following characters. The name string also cannot contain leading and trailing spaces.

/	forward slash
\	backslash
*****	asterisk
,	comma
:	colon
;	semi-colon
=	equal sign
+	plus sign
?	question mark
|	vertical bar
<	left angle bracket
>	right angle bracket
&	ampersand (and sign)
%	percent sign
’	single quote mark
″	double quote mark
]]>	No specific name exists for this character combination.
.	period (not valid if first character; valid if a later character)
#	Hash mark
$	Dollar sign

Configuration repositories

A configuration repository stores configuration data.

By default, configuration repositories reside in the *config* subdirectory of the product installation root directory.

Handling temporary configuration files resulting from session timeout

If the console is not used for 15 minutes or more, the session times out. The same thing happens if you close the browser window without saving the configuration file. Changes to the file are saved to a temporary file when the session times out, after 15 minutes. This topic discusses what happens depending on whether you load the saved file.

A configuration file must have been saved from a previous administrative console session for the user ID that you are currently using to access the administrative console.

When a session times out, the configuration file in use is saved under the `userid/timeout` directory under the ServletContext’s temp area. This value is the value of the javax.servlet.context.tempdir attribute of the ServletContext context. By default, it is: *`profile_root`*`/temp/hostname/Administration/admin/admin.war`

You can change the temp area by specifying it as a value for the tempDir init-param of the action servlet in the deployment descriptor (web.xml) of the administrative application.

The configuration file is also saved automatically when the same user ID logs into the non-secured console again, effectively starting a different session. This process is equivalent to forcing the existing user ID out of session, similar to a session timing out.

The next time you log on to the administrative console, you are prompted to load the saved configuration file. Do one of the following actions:

- Load the saved file.

   1. If a file with the same name exists in the `profile_root/config` directory, that file is moved to the `userid/backup` directory in the temp area.
   2. The saved file is moved to the `profile_root/config` directory.
   3. The file is then loaded.

- Do not load the saved file.

  The saved file is deleted from the `userid/timeout` directory in the temp area.

You loaded the saved configuration file if you chose to do so.

Once you have logged into the administrative console, do whatever administration of WebSphere Application Server that you need to do.

Changing the location of temporary configuration files

You can change the default directory where temporary configuration files are stored.

The configuration repository uses copies of configuration files and temporary files while processing repository requests. It also uses a backup directory while managing the configuration. You can change the default locations of these files from the configuration directory to a directory of your choice by using the administrative console.

The default location for the configuration temporary directory is *profile root*/config/temp. Change the location by doing the following actions:

1. Click **Servers** > **Application servers** in the navigation tree of the administrative console. Then, click ***server name*** > **Administration** > **Administration services** > **Repository service** > **Custom properties**.
2. On the Properties page, click **New**.
3. On the settings page for a property, define a property for the temporary file location. The key for this property is `was.repository.temp`. The value is the full path name to the desired location.
4. Click **OK**.

Changing the location of backed-up configuration files

You can change the default directory where backup files are stored.

During administrative processes like adding a node to a cell or updating a file, configuration files are temporarily backed up to a backup location.

The default location for the backup configuration directory is *profile root*/config/backup. Change the location by doing the following actions:

1. Click **Servers** > **Application servers** in the navigation tree of the administrative console. Then, click ***server name*** > **Administration** > **Administration services** > **Repository service** > **Custom properties**.
2. On the Properties page, click **New**.
3. On the settings page for a property, define a property for the backup file location. The key for this property is `was.repository.backup`. The value is the full path name to the desired location.
4. Click **OK**.

Changing the location of temporary workspace files

The default workspace root is calculated based on the user installation root. This topic discusses how to change the location of temporary workspace files.

You must first install WebSphere Application Server before you change the location of temporary workspace files.

With the administrative console workspace, client applications can navigate the configuration. Each workspace has its own repository location defined either in the system property or the property that is passed to a workspace manager when creating the workspace: workspace.user.root or workspace.root, which is calculated as %workspace.root%/*user_ID*/workspace/wstemp.

The default workspace root is calculated based on the user installation root: %user.install.root%/wstemp. You can change the default location of temporary workspace files:

Change the setting for the Java system property *workspace.user.root* or *workspace.root* so its value is no longer set to the default location.

Set the Java system property when launching a Java process using the -D option. For example, to set the default location to the full path of the root of all users' directories, use the following option:

```
-Dworkspace.user.root=full_path_for_root_of_all_user_directories
```

You changed the location of temporary workspace files.

Backing up and restoring administrative configuration files

This topic discusses how to back up and restore administrative configuration files.

WebSphere Application Server represents its administrative configurations as XML files. You should back up configuration files on a regular basis.

Restore the configuration only if the configuration files that you backed up are at the same level of the release, including fixes, as the release to which you are restoring.

1. Run the backupConfig command to back up configuration files. See the backupConfig command topic in the *Using the administrative clients* PDF for information.
2. Run the restoreConfig command to restore configuration files. See the restoreConfig command topic in the *Using the administrative clients* PDF for information. Specify backup files that do not contain invalid or inconsistent configurations.

Server configuration files: Resources for learning

Use the following links to find relevant supplemental information about administering WebSphere Application Server configuration files. 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:

Administration

- IBM WebSphere Application Server Redbooks

  This site contains a listing of all WebSphere Application Server Redbooks.
- IBM WebSphere developerWorks

  This site is the home of technical information for developers working with WebSphere products. You can download WebSphere software, take a fast path to developerWorks zones, such as VisualAge Java or WebSphere Application Server, learn about WebSphere products through a newcomers page, tutorials, technology previews, training, and Redbooks, get answers to questions about WebSphere products, and join the WebSphere community, where you can keep up with the latest developments and technical papers.

- WebSphere Application Server Support page

  Take advantage of the Web-based Support and Service resources from IBM to quickly find answers to your technical questions. You can easily access this extensive Web-based support through the IBM Software Support portal at URL `http://www-3.ibm.com/software/support/` and search by product category, or by product name. For example, if you are experiencing problems specific to WebSphere Application Server, click **WebSphere Application Server** in the product list. The WebSphere Application Server Support page appears.

Chapter 9. Administering application servers

An application server configuration provides settings that control how an application server provides services for running applications and their components.

Install WebSphere Application Server.

After you install WebSphere Application Server you might have to perform some of the following tasks. Other then creating application servers, these tasks can be performed in any order.

- Create application servers.
- Configure transport chains.
- Develop custom services.
- Define processes for the application server.
- Configure the Java virtual machine.

After preparing a server, deploy an application or component on the server. See the *Administering applications and their environment* PDF for a sample procedure that you might follow in configuring the application server runtime and resources.

Manage the application servers you created.

Application servers

Application servers extend the ability of a Web server to handle Web application requests, typically using Java technology. An application server makes it possible for a server to generate a dynamic, customized response to a client request.

For example, given:

1. A user at a Web browser on the Internet visits a company Web site. The user requests to use an application that provides access to data in a database.
2. The user request flows to the Web server.
3. The Web server determines that the request involves an application containing resources not handled directly by the Web server (such as servlets). It forwards the request to WebSphere Application Server.
4. WebSphere Application Server forwards the request to one of its application servers on which the application is running.
5. The invoked application then processes the user request. For example:
   - An application servlet prepares the user request for processing by an enterprise bean that performs the database access.
   - The application produces a dynamic Web page containing the results of the user query.
6. The application server collaborates with the Web server to return the results to the user at the Web browser.

WebSphere Application Server provides multiple application servers that can be either separately configured processes or nearly identical clones.

Creating application servers

WebSphere Application Server provides you with the capability to create application servers.

During the installation process WebSphere Application Server creates a default application server, named server1. You must issue the wsadmin **createApplicationServer** command from a command line to create additional application server. For information on how to perform this task, see the *Using the administrative clients* PDF.

Any additional servers you create cannot be managed using the administrative console. The only server you can manage using the administrative console is the default server .

Configuring application servers for UCS Transformation Format

You can use the client.encoding.override=UTF-8 JVM argument to configure an application server for UCS Transformation Format. This format enables an application server to handle most character encodings, including specialized mathematical and technical symbols.

The client.encoding.override=UTF-8 argument is provided for backwards compatibility. You should only specify this argument if you require multiple language encoding support in the administrative console and there is no other way for you to set the request character encoding required to parse post and query strings.

Before configuring an application server for UCS Transformation Format, you should try to either:

- Explicitly set the ServletRequest Encoding inside of the JSP or Servlet that is receiving the POST and or query string data, which is the preferred J2EE solution, or
- Enable the autoRequestEncoding, option, which uses the client's browser settings to determine the appropriate character encoding. Older browsers might not support this option.

Important: If the client.encoding.override=UTF-8 JVM argument is specified, the autoRequestEncoding option does not work even if it is enabled. Therefore, when an application server receives a client request, it checks to see if the charset option is set on the content type header of the request:

1. If it is set, the application server uses the content type header for character encoding.
2. If it is not set, the application server uses the character encoding that is specified for the default.client.encoding system property.
3. If neither charset nor the default.client.encoding system property is set, the application server uses the ISO-8859-1 character set.

The application server never checks for an Accept-Language header. However, if the autoRequestEncoding option is working, the application server checks for an Accept-Language header before checking to see if a character encoding is specified for the default.client.encoding system property.

To configure an application server for UCS Transformation Format:

1. In the administrative console, click **Servers > Application servers** and select the server you want to enable for UCS Transformation Format.
2. Click **Java and Process Management > Process Definition > Java Virtual Machine**.
3. Specify `-Dclient.encoding.override=UTF-8` for **Generic JVM Arguments** and click **OK**. When this argument is specified, UCS Transformation Format is used instead of the character encoding that would be used if the autoRequestEncoding option was in effect.
4. Click **Save** to save your changes.
5. Restart the application server.

The application server uses UCS Transformation Format for encoding.

Managing application servers

You can use the administrative console or command line tools to manage your application servers.

During the installation process, WebSphere Application Server creates a default application server, server1. You can use either the administrative console or command line tools to manage this application server. However, if you create additional application servers, you must use command line tools to manage those other application servers.

To use the administrative console to view and manage the default application server:

1. In the administrative console click **Servers > Application servers**. The Application servers page lists the application servers in your environment and the status of each of these servers. You can use this page to change the status of a listed application server.
2. Click **server1** to view or change the configuration settings for this application server.
3. Save any configuration changes you make.
4. Monitor the running application servers.

The application servers are properly configured and the appropriate application servers are running.

Create additional application server as required.

Server collection

Use this page to view information about and manage application servers, Java message service (JMS) servers, and Web servers. For the Network Deployment product, you can also use this page to view information about and manage generic servers.

Application servers

The Application servers page lists the application servers in the cell. You can use this page to start and stop these application servers.

To view this administrative console page, click **Servers > Application servers**.

Java message service (JMS) servers

The JMS servers page lists the JMS servers in the cell. You can use this page to start and stop these JMS servers. Each JMS server provides the functions of the JMS provider for a node in your administrative domain. There can be at most one JMS server on each node in the administration domain, and any application server within the domain can access JMS resources served by any JMS server on any node in the domain. To view this administrative console page, click **Servers > JMS servers**.

Note: JMS servers apply only to WebSphere Application Server Version 5.x nodes. You cannot create a JMS server on a node that is running WebSphere Application Server 6.x, but the existing Version 5.x JMS servers continue to be displayed, and you can modify their properties. However, you cannot use this page to delete a Version 5.x JMS server.

Web servers

The Web servers page lists the Web servers in your administrative domain. You can use this page to generate and propagate a Web server plug-in configuration file, create new Web servers, create new Web server templates, or delete existing Web servers. You can also use this page to start and stop these Web servers. To view this administrative console page, click **Servers > Web servers**.

Name

Specifies a logical name for the server. For WebSphere Application Server, server names must be unique within a node.

Node

Specifies the name of the node holding the server.

Version

Specifies the version of the WebSphere Application Server product on which the server runs.

Status

Indicates whether the server is started or stopped. (Network Deployment only)

Note that if the status is *Unavailable*, the node agent is not running in that node and you must restart the node agent before you can start the server.

Application server settings

Use this page to view or change the settings of an application server instance. An application server is a server which provides services required to run enterprise applications.

To view this administrative console page, click **Servers** > **Application Servers** > *server_name*.

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 the server is running.

Name:

Specifies a logical name for the server. Server names must be unique within a node. However, for multiple nodes within a cluster, you may have different servers with the same server name as long as the server and node pair are unique. You cannot change the value that displays in this field.

For example, a server named *server1* in a node named *node1* in the same cluster with a server named *server1* in a node named *node2* is allowed. Configuring two servers named *server1* in the same node is not allowed. WebSphere Application Server uses the server name for administrative actions, such as referencing the server in scripting.

Data type	String
Default	server1

Run in development mode:

Enabling this option may reduce the startup time of an application server. This may include JVM settings such as disabling bytecode verification and reducing JIT compilation costs. Do not enable this setting on production servers. This setting is only available on application servers running WebSphere Application Server Version 6.0 and later.

Specifies that you want to use the JVM settings **-Xverify** and **-Xquickstart** on startup. After selecting this option, save the configuration and restart the server to activate development mode.

The default setting for this option is `false`, which indicates that the server will not be started in development mode. Setting this option to `true` specifies that the server will be started in development mode (with settings that will speed server startup time).

Data type	Boolean
Default	false

Parallel start:

Select this field to start the server on multiple threads. This might shorten the startup time.

Specifies that you want the server components, services, and applications to start in parallel rather than sequentially.

The default setting for this option is `true`, which indicates that the server be started using multiple threads. Setting this option to `false` specifies that the server will not be started in using multiple threads (which may lengthen startup time).

Note that the order in which the applications start depends on the weights you assigned to each them. Applications that have the same weight are started in parallel. You set an application's weight with the *Starting weight* option on the **Applications > Enterprise Applications >** *application_name* page of the Administrative Console.

Data type	Boolean
Default	true

Class loader policy:

Select whether there is a single class loader to load all applications or a different class loader for each application.

Class loading mode:

Specifies whether the class loader should search in the parent class loader or in the application class loader first to load a class. The standard for Developer Kit class loaders and WebSphere Application Server class loaders is `Parent first`.

This field only applies if you set the Class loader policy field to `single`.

If you select `Parent last`, your application can override classes contained in the parent class loader, but this action can potentially result in ClassCastException or linkage errors if you have mixed use of overridden classes and non-overridden classes.

Access to internal server classes:

Specifies whether the applications can access many of the server implementation classes.

If you select `Allow`, applications can access many of the server implementation classes. If you select `Restrict`, applications can not access many of the server implementation classes. The applications get a ClassNotFoundException if they attempt to access those classes.

Applications typically use the supported APIs and do not need to access system internals.

Process Id:

The native operating system's process ID for this server.

The process ID property is read only. The system automatically generates the value.

Cell name:

The name of the cell in which this server is running.

The Cell name property is read only.

Node name:

The name of the node in which this server is running.

The Node name property is read only.

State:

The run-time execution state for this server.

The State property is read only.

Ports collection:

Use this page to view and manage communication ports used by run-time components running within a process. Communication ports provide host and port specifications for a server.

To view this administrative console page, click **Servers** > **Application Servers** > *server_name* > **Communications** > **Ports**.

This page displays only when you are working with ports for application servers.

Port Name:

Specifies the name of a port. Each name must be unique within the server.

Host:

Specifies the IP address, domain name server (DNS) host name with domain name suffix, or just the DNS host name, used by a client to request a resource (such as the naming service, or administrative service).

Port:

Specifies the port for which the service is configured to accept client requests. The port value is used in conjunction with the host name.

Transport Details:

Provides a link to the transport chains associated with this port. If no transport chains are associated with this port, the string ″No associated transports″ appears in this column.

Ports settings:

Use this to view and change the configuration for a communication port used by run-time components running within a process. A communication port provides host and port specifications for a server.

For base WebSphere Application Server, you can view this administrative console page, by clicking **Servers** > **Application Servers** > *server_name* > **Ports** > *port_name*

Port Name:

Specifies the name of the port. The name must be unique within the server.

Note that this field displays only when you are defining a port for an application server. You can select either:

Well-known Port
: When you select this option , you can select a previously defined port from the drop down list

User-defined Port
: When you select this option, you must create a port with a new name by entering the name of the new port in the text box

Data type	String

Host:

Specifies the IP address, domain name server (DNS) host name with domain name suffix, or just the DNS host name, used by a client to request a resource (such as the naming service, administrative service, or JMS broker).

For example, if the host name is `myhost`, the fully qualified DNS name can be `myhost.myco.com` and the IP address can be `155.123.88.201`.

Host names on the ports can be resolvable names or IP addresses. The server will bind to the specific host name or IP address that is supplied. That port will only be accessible through the IP address that is resolved from the given host name or IP address. The IP address may be of the IPv4 (Internet Protocol Version 4) format for all platforms, and IPv6 (Internet Protocol Version 6) format on specific operating systems where the server supports IPv6.

Data type	String
Default	* (asterisk)

Port:

Specifies the port for which the service is configured to accept client requests. The port value is used in conjunction with the host name.

Port numbers in the server can be reused among multiple ports as long as they have host names that resolve to unique IP addresses and there is not a port with the same port number and a wildcard (*) host name. A port number is valid in the range of 0 and 65535. 0 specifies that the server should bind to any ephemeral port available.

Important: Port sharing cannot be created using the administrative console. If you need to share a port, you must use wsadmin commands to define that port. You must also make sure that the same discrimination weights are defined for all of the transport channels associated with that port.

Protocol channels only accept their own protocol. However, application channels usually accept anything that reaches them. Therefore, for application channels, such as WebContainer or Proxy, you should specify larger discrimination weights when sharing levels with protocol channels, such as HTTP or SSL. The one exception to this rule is if you have application channels that perform discrimination tests faster than the protocol channels. For example, a JFAP channel is faster at deciding on a request than the SSL protocol channel, and should go first for performance reasons. However, the WebContainer and Proxy channels must always be last because they accept everything that is handed to them.

Data type	Integer
Default	None

The following table lists server endpoints and their respective port ranges. In contrast to the z/OS platform, on a distributed platform or on the i5/OS platform, the ORB_LISTENER_ADDRESS and the BOOTSTRAP_ADDRESS endpoints must not specify the same port.

Endpoint (port)	Acceptable values for the port field
BOOTSTRAP_ADDRESS	1 - 65536
CSIV2_SSL_MUTUALAUTH_LISTENER_ADDRESS	1 - 65535
CSIV2_SSL_SERVERAUTH_LISTENER_ADDRESS	1 - 65535
DCS_UNICAST_ADDRESS	1 - 65536

Endpoint (port)	Acceptable values for the port field
DRS_CLIENT_ADDRESS	1 - 65536
ORB_LISTENER_ADDRESS	0 - 65535 (If 0 is specified, the server starts on any available port and does not use the location service daemon)
SAS_SSL_SERVERAUTH_LISTENER_ADDRESS	1 - 65535
SIB_ENDPOINT_ADDRESS	1 - 65536
SIB_ENDPOINT_SECURE_ADDRESS	1 - 65536
SIB_MQ_ENDPOINT_ADDRESS	1 - 65536
SIB_MQ_ENDPOINT_SECURE_ ADDRESS	1 - 65536
SOAP_CONNECTOR_ ADDRESS	1 - 65536
WC_adminhost	1 - 65536
WC_adminhost_secure	1 - 65536
WC_defaulthost	1 - 65536
WC_defaulthost_secure	1 - 65536
ORB_SSL_LISTENER_ADDRESS	Not supported on the distributed and iSeries platforms

Custom property collection:

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.

Name:

Specifies the name (or key) for the property.

Each property name must be unique. If the same name is used for multiple properties, the value specified for the first property that has that name is used.

Do not start your property names with `was.` because this prefix is reserved for properties that are predefined in the application server.

Value:

Specifies the value paired with the specified name.

Description:

Provides information about the name-value pair.

Custom property settings:

Use this page to configure 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. Defining a new property enables you to configure a setting beyond that which is available in the administrative console.

To view this administrative console page, click **Servers > Application Servers >** *server_name*. Then, under Server Infrastructure, click **Administration** > **Custom Properties**

You can then click **New** to, create a new custom property, click on the name of an existing property to change its settings, or click **Delete** to delete an existing property.

Name:

Specifies the name (or key) for the property.

Each property name must be unique. If the same name is used for multiple properties, the value specified for the first property that has that name is used.

Do not start your property names with `was.` because this prefix is reserved for properties that are predefined in WebSphere Application Server.

Data type	String

Value:

Specifies the value paired with the specified name.

Data type	String

Description:

Provides information about the name and value pair.

Data type	String

Server component collection:

Use this page to view information about and manage server component types such as application servers, messaging servers, or name servers.

To view this administrative console page, click **Servers > Application Servers >** *server_name*. Then, under Server Infrastructure, click **Administration > Server Components**.

Type:

Specifies the type of internal server.

Server component settings:

Use this page to view or configure a server component instance.

To view this administrative console, click **Servers > Application Servers >** *server_name*. Then, under Server Infrastructure, click **Administration > Server Components >** *server_component_name*.

Name:

Specifies the name of the component.

Data type	String

Initial State:

Specifies the desired state of the component when the server process starts. The options are: *Started* and *Stopped*. The default is *Started*.

Data type	String
Default	Started

Generic server settings

Use this page to view or change the settings of a generic server.

A generic server is a server that is managed in the WebSphere Application Server administrative domain, although it is not a server that is supplied by the WebSphere Application Server product. The generic server can be any server or process that is necessary to support the Application Server environment, including a Java server, a C or C++ server or process, or a Remote Method Invocation (RMI) server.

To view this administrative console page, click **Servers > Generic Servers >** *server_name*.

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 the server is running.

Name:

Specifies a logical name for the generic server.

It is highly recommended that you use a naming scheme that makes it easy to distinguish your generic application servers from regular WebSphere Application Servers. This will enable you to quickly determine whether to use the Terminate or Stop button in the administrative console to stop a specific application server.

You must use the Terminate button to stop a generic application server.

Data type	String
Default	

Environment entries collection

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

To view this page, in the administrative console click **Proxy Servers >** *server_name*, and then under Server Infrastructure, click **Java and Process Management > Environment Entries**.

Name

Specifies the name (or key) for the environment entry. The name is a string that is used to set an internal system configuration environment entry.

Each environment entry name must be unique. If the same name is used for multiple environment entries, the value specified for the first environment entry that has that name is used.

Do not start your environment entry names with `was.` because this prefix is reserved for environment entries that are predefined for WebSphere Application Server.

Value

Specifies the value paired with the specified name.

Description

Provides information about the name-value pair.

Environment entries settings

Use this page to configure arbitrary name-value pairs of data, where the name is an environment entry key and the value is a string value that can be used to set internal system configuration environment entries. Defining a new environment entry enables you to configure a setting beyond that which is available in the administrative console.

To view this page, in the administrative console click **Proxy Servers >** *server_name*. Under Server Infrastructure, click **Java and Process Management > Environment Entries**. Then do one of the following:

- Click **New** to create a new environment entry.
- Click the name of an existing environment entry to change its settings,
- Select an existing environment entry and click **Delete** to delete that entry.

Name:

Specifies the name (or key) for the environment entry.

Each environment entry name must be unique. If the same name is used for multiple environment entries, the value specified for the first environment entry that has that name is used.

Do not start an environment entry name with `was.` because this prefix is reserved for environment entries that are predefined in WebSphere Application Server.

Data type	String

Value:

Specifies the value paired with the specified name.

Data type	String

Description:

Provides information about the name and value pair.

Data type	String

Starting an application server

Starting an application server starts a new server process based on the process definition settings of the current server configuration.

This procedure for starting a server also normally applies to restarting a server. The one exception might be if a server fails and you want the recovery functions to complete their processing prior to new work being started on that server. In this situation you must restart the server in recovery mode.

If you create a new application server definition using WebSphere Application Server Express, you cannot start, stop, or manage the new server using the administrative console that is associated with the original base server. You must use commands to perform these tasks for the new server.

There are several options for starting an application server:

- You can use the administrative console:

   1. In the administrative console, click **Servers > Application servers**.
   2. Select server1 and click **Start**. You can view the **Status** value and any messages or logs to make sure the application server starts.

- Windows You can use the **Start** menu on a Microsoft Windows system. For example, if you are using the Express version of the product, click **Start > Programs > IBM WebSphere > Express V***n* **> Profiles >** *profile_name* **> Start the server**. You can check that the server has successfully started by checking the startServer.log file. If the server has successfully started, the last two lines of the startServer.log file reads:

  ```
  Server launched.  Waiting for initialization status.
  Server server1 open for e-business; process id is 1932.
  ```

  The startServer.log file is located in the *profile_root*\logs\server1 directory if you have installed your server with the default settings. The server name and process id vary depending on your settings.

- AIX On an AIX platform, you can issue a command from the command line. Use the **startServer** command from the *app_server_root*/bin directory, as shown below. To start a server that is associated with a non-default profile, issue the **startServer** command from the *profile_root*/bin directory.

  ```
  #  ./startServer.sh server1
  ```

  You can check that the server has successfully started by checking the startServer.log file. If the server has successfully started, the last two lines of the startServer.log file reads:

  ```
  Server launched.  Waiting for initialization status.
  Server server1 open for e-business; process id is 1932.
  ```

  AIX On the AIX platform, the startServer.log file is located in the *profile_root*/logs/server1/ directory.

- On other distributed platforms, you can use a command line tool to issue a **startServer** command.
- **Optional:** Start an application server with tracing and debugging active.

  To start the Application Server with standard Java debugging enabled:

  1. In the administrative console, click **Servers > Application Servers**, click the application server whose processes you want to trace and debug, and then click **Java and Process Management > Process Definition > Java Virtual Machine**.
  2. On the Java virtual machine page, select the **Debug Mode** setting to enable the standard Java debugger. If needed, set debug arguments. Then, click **OK**.
  3. Save the changes to a configuration file.
  4. Stop the Application Server.
  5. Start the Application Server again as previously described.

After the server starts, install your applications.

You can use one of these same options to stop an application server.

Restarting an application server in recovery mode

When an application server instance with active transactions in progress restarts after a failure, the transaction service uses recovery logs to complete the recovery process. These logs, which each transactional resource maintains, are used to rerun any InDoubt transactions and return the overall system to a self-consistent state.

When you restart an application server in recovery mode:

- Transactional resources complete the actions in their recovery logs and then shut down. This action frees up any resource locks that the application server held prior to the failure.
- During the recovery period, only the subset of application server functions that are necessary for transactional recovery to proceed are available.
- The application server does not accept new work during the recovery process.

- The application server shuts down when the recovery is complete.

This recovery process begins as soon as all of the necessary subsystems within the application server are available. If the application server is not restarted in recovery mode, the application server can start accepting new work as soon as the server is ready, which might occur before the recovery work has completed.

Normally, this process is not a problem. However, situations exist when your operating procedures might not be compatible with supporting recovery work and new work simultaneously. For example, you might have a high availability environment where the work handled by the application server that failed is immediately moved to another application server. This backup application server then exclusively processes the work from the application server that failed until recovery has completed on the failed application server and the two application servers can be re-synchronized. In this situation, you might want the failing application server to only perform its transactional recovery process and then shut down. You might not want this application server to start accepting new work while the recovery process is taking place.

To prevent the assignment of new work to an application server that is going through its transaction recovery process, restart the application server in recovery mode.

If you want to be able restart an application server in recovery mode, you must perform the following steps before a failure occurs, and then restart the application server to enable your configuration changes:

- **Optional:** If the server is monitored by a node agent, you must clear the Automatic restart option for that server. Clearing this option prevents the node agent from automatically restarting the server in normal mode, before you have a chance to start it in recovery mode.
  1. In the administrative console, click **Servers > Application Servers >** *server_name*.
  2. Under Server Infrastructure, click **Java and Process Management > Monitoring Policy**.
  3. Clear the Automatic restart option.
- If the server is monitored by a node agent, you must clear the Automatic restart option for that server.
  1. In the administrative console, click **Servers > Application Servers >** *server_name*.
  2. Under Server Infrastructure, click **Java and Process Management > Monitoring Policy**.
  3. Clear the Automatic restart option.
- If a catastrophic failure occurs that leaves InDoubt transactions, issue the **startServer** *server_name* **-recovery** command from the command line. This command restarts the server in recovery mode. You must issue the command from the *install_root*/profiles/*profile_name*/bin directory for the profile with which the server is associated.

The application server restarts in recovery mode, performs transactional recovery, and shuts down. Any resource locks that the application server held prior to the failure are released.

Running application servers from a non-root user

By default, the root user ID is used to run all application server processes on a Linux and UNIX platform. However, you can run all application server processes under the same non-root user and user group. This task describes how to run an application server process from a non-root user.

If administrative security is enabled, the user account repositories must not be the local operating system. Using the local operating system user registry requires the node agent to run as root. Refer to the *Securing applications and their environment* PDF for details.

If you are using the Tivoli Access Manager to perform authentication or authorization for WebSphere Application Server, it is important to be aware of potential permissions problems. For more information, see the *Securing applications and their environment* PDF.

When WebSphere Application Server is run as a UNIX user, it can only access files owned by its primary group. If it tries to access files by its secondary group, a java.io.FileNotFoundException: will occur because the file access permissions do not allow this type of access.

Run your application servers as non-root when you no longer want to use root authority. For security or administrative reasons, you may want to change to non-root user IDs. Perform this task at any time to change the permissions of an application server. You must restart the application server in order for the changes to take effect.

For the following steps, assume that:

- was1 is the user to run the application server
- wasgroup is the primary user group for user was1
- wasnode is the node name
- server1 is the application server
- /opt/WebSphere/AppServer is the installation root
- nodeProfile1 is the profile name.

For information about creating a profile, see manageprofiles command.

To configure an application server to run as non-root, complete the following steps.

1. Log on to the application server system as the root user.
2. Create the user ID was1 with a primary user group of wasgroup. The user ID, was1, is an example. You can name the user something else.
3. Log off and back on as root.
4. Start server1 as root. Run the startServer.sh script from the /bin directory of the installation root:

   ```
   startServer.sh server1
   ```

5. Specify user and group ID values for the **Run As User** and **Run As Group** settings for a server:
   a. Start the administrative console.
   b. Go to the Process execution page of the administrative console. You must define all three properties in the following table. In the administrative console, click **Servers > Application Servers >** *server*, and then under Server Infrastructure, click **Java and Process Management > Process Definition > Process Execution**, and change all of the following values:

Property	Value
Run As User	was1
Run As Group	wasgroup
UMASK	022 The value 022 means the files the process creates are writable by the group and by others as defined on the Linux or UNIX platforms.

   c. Click **OK**.
   d. Save the configuration.

6. Stop the application server. Use the stopServer.sh script from the /bin directory of the installation root:

   ```
   stopServer.sh server1
   ```

7. Change file permissions as the root user. The following example assumes that the installation root directory for WebSphere Application Server is /opt/WebSphere/AppServer:

   ```
   chgrp wasgroup /opt/WebSphere
   chgrp wasgroup /opt/WebSphere/AppServer
   chgrp -R wasgroup /opt/WebSphere/AppServer/profiles
   chgrp -R wasgroup /opt/WebSphere/AppServer/profiles/profile name
   chgrp -R wasgroup /opt/WebSphere/AppServer/logs
   ```

```
chgrp -R wasgroup /opt/WebSphere/AppServer/properties
chgrp -R wasgroup /opt/WebSphere/AppServer/temp
chgrp -R wasgroup /opt/WebSphere/AppServer/cloudscape
chgrp -R wasgroup /opt/WebSphere/AppServer/bin
chgrp -R wasgroup /opt/WebSphere/AppServer/java
chgrp -R wasgroup /opt/WebSphere/AppServer/lib
chgrp -R wasgroup /opt/WebSphere/AppServer/installedChannels
chgrp -R wasgroup /opt/WebSphere/AppServer/profiles/profile name/installedFilters
chgrp -R wasgroup /opt/WebSphere/AppServer/etc/
chgrp -R wasgroup /opt/WebSphere/AppServer/classes
chgrp -R wasgroup /opt/WebSphere/AppServer/systemApps

chmod g+wr /opt/WebSphere
chmod g+wr /opt/WebSphere/AppServer
chmod -R g+wr /opt/WebSphere/AppServer/profiles
chmod -R g+rw /opt/WebSphere/AppServer/profiles/profile name
chmod -R g+wr /opt/WebSphere/AppServer/logs
chmod -R g+wr /opt/WebSphere/AppServer/properties
chmod -R g+wr /opt/WebSphere/AppServer/temp
chmod -R g+wr /opt/WebSphere/AppServer/cloudscape
chmod -R g+r /opt/WebSphere/AppServer/bin
chmod -R g+r /opt/WebSphere/AppServer/java
chmod -R g+r /opt/WebSphere/AppServer/lib
chmod -R g+rw /opt/WebSphere/AppServer/installedChannels
chmod -R g+rw /opt/WebSphere/AppServer/profiles/profile name/installedFilters
chmod -R g+rw /opt/WebSphere/AppServer/etc/
chmod -R g+rw /opt/WebSphere/AppServer/classes
chmod -R g+rw /opt/WebSphere/AppServer/systemApps
```

8. Log on to the application server system as user was1.
9. Start server1 as was1. Run the startServer.sh script from the /bin directory of the installation root:

   ```
   startServer.sh server1
   ```

10. If creating another server with a different user ID, follow this procedure again for the new user ID and server name.

    The two user IDs must share the same group, wasgroup.

You can start an application server from a non-root user.

Detecting and handling problems with runtime components

You must monitor the status of runtime components to ensure that, once started, they remain operational as needed.

1. Regularly examine the status of runtime components.

   Browse messages displayed under WebSphere Runtime Messages in the status area at the bottom of the console. The runtime event messages, marked with a red X, provide detailed information on event processing.
2. If an application stops running when it should be operational, examine the status of the application on an Applications page and try restarting the application.
3. If the runtime components do not restart, reexamine the messages and read information on problem determination to help you to restart the components.

Stopping an application server

Stopping an application server ends a server process based on the process definition settings in the current application server configuration.

There are several options for stopping an application server:

- Windows For the Windows platform, you can use the Start menu to stop your application server. In the administrative console, click **Start** > **Programs** > **IBM WebSphere** > **Express v***n* > **Stop the server**. When the server stops successfully, the stopServer.log file contains the following in the last two lines:

```
Server stop request issued.  Waiting for stop status.
Server server1 stop completed.
```

The server name varies depending on your settings.

- For other distributed platforms, except AIX, you can use the **stopServer** command to stop an application server from the command line.

  A warning message appears if you are stopping the application server that is running the administrative console application.

  AIX For the AIX platform, you can issue the **stopServer** or the **stopManager** commands from the /usr/WebSphere/AppServer/bin directory:

  ```
  #  ./stopServer.sh server1
  ```

  ```
  #   ./stopManager.sh
  ```

If you experience any problems shutting down a server, see the *Troubleshooting and support* PDF.

Setting the time zone for a single application server

You can ensure that your application components use the correct time zone. How you do this varies by the operating system on which WebSphere Application Server is installed and, in some cases, by the scope required.

In some application environments, it is important that application server components use the same time zone.

To change the time zone setting for a single application server, add the user.timezone Java virtual machine (JVM) custom property to the configuration settings of a specific application server:

1. In the administrative console click **Application Servers >** *server_name* .
2. Under Server Infrastructure, click **Java and Process Management > Process Definition > Java Virtual Machine > Custom Properties**.
3. Specify `user.timezone` in the Name field and the appropriate time zone in the Value field.
4. **Optional:** Enter a description of the variable and the setting you specified.
5. Click **Apply**.
6. Save your work.

   Make sure Synchronize changes with Nodes is selected, and click **Save**.

All of the components of this application server use the time zone specified for the custom property.

Stop and restart this application server. You must restart the server for the change to take effect.

Supported time zone values

Use this page as a reference for time zone variables that are supported by WebSphere Application Server.

The following table lists the time zone values that WebSphere Application Server supports:

- The **Time zone ID** column lists time zones, in boldface, and the locations within each time zone.
- The **Raw offset** column lists the difference, in hours and minutes, between Greenwich Mean Time (GMT) and the specified time zone.
- The **DST offset** column lists the offset, in minutes, for Daylight Savings Time (DST). If the field is blank, the time zone does not use DST.
- The **Display name** column lists the names of the time zones.
- The **QTIMZON variable** column only applies to the i5/OS operating system. The **QTIMZON variable** column lists the corresponding value for the QTIMZON system variable. If multiple values are specified in this column, either value is acceptable.

Time zone ID	Raw offset (Hours : Minutes)	DST offset (Minutes)	Display name	QTIMZON variable (i5/OS only)
Etc/GMT+12	-12 : 00		GMT-12:00	
Etc/GMT+11	-11 : 00		GMT-11:00	
MIT	-11 : 00		West Samoa Time	
Pacific/Apia	-11 : 00		West Samoa Time	QN1100UTCS
Pacific/Midway	-11 : 00		Samoa Standard Time	
Pacific/Niue	-11 : 00		Niue Time	
Pacific/Pago_Pago	-11 : 00		Samoa Standard Time	
Pacific/Samoa	-11 : 00		Samoa Standard Time	
US/Samoa	-11 : 00		Samoa Standard Time	
America/Adak	-10 : 00	60	Hawaii-Aleutian Standard Time	QN1000HAST
America/Atka	-10 : 00	60	Hawaii-Aleutian Standard Time	
Etc/GMT+10	-10 : 00		GMT-10:00	
HST	-10 : 00		Hawaii Standard Time	
Pacific/Fakaofo	-10 : 00		Tokelau Time	
Pacific/Honolulu	-10 : 00		Hawaii Standard Time	QN1000UTCS
Pacific/Johnston	-10 : 00		Hawaii Standard Time	
Pacific/Rarotonga	-10 : 00		Cook Is. Time	
Pacific/Tahiti	-10 : 00		Tahiti Time	
SystemV/HST10	-10 : 00		Hawaii Standard Time	
US/Aleutian	-10 : 00	60	Hawaii-Aleutian Standard Time	
US/Hawaii	-10 : 00		Hawaii Standard Time	
Pacific/Marquesas	-9 : 30		Marquesas Time	
AST	-9 : 00	60	Alaska Standard Time	QN0900AST
America/Anchorage	-9 : 00	60	Alaska Standard Time	
America/Juneau	-9 : 00	60	Alaska Standard Time	
America/Nome	-9 : 00	60	Alaska Standard Time	
America/Yakutat	-9 : 00	60	Alaska Standard Time	
Etc/GMT+9	-9 : 00		GMT-09:00	
Pacific/Gambier	-9 : 00		Gambier Time	QN0900UTCS
SystemV/YST9	-9 : 00	60	Alaska Standard Time	
US/Alaska	-9 : 00	60	Alaska Standard Time	
America/Dawson	-8 : 00	60	Pacific Standard Time	
America/Ensenada	-8 : 00	60	Pacific Standard Time	
America/Los_Angeles	-8 : 00	60	Pacific Standard Time	
America/Tiajuana	-8 : 00	60	Pacific Standard Time	
America/Vancouver	-8 : 00	60	Pacific Standard Time	
America/Whitehorse	-8 : 00	60	Pacific Standard Time	

Time zone ID	Raw offset (Hours : Minutes)	DST offset (Minutes)	Display name	QTIMZON variable (i5/OS only)
Canada/Pacific	-8 : 00	60	Pacific Standard Time	
Canada/Yukon	-8 : 00	60	Pacific Standard Time	
Etc/GMT+8	-8 : 00		GMT-08:00	
Mexico/BajaNorte	-8 : 00	60	Pacific Standard Time	
PST	-8 : 00	60	Pacific Standard Time	QN0800PST, QN0800U
PST8PDT	-8 : 00	60	Pacific Standard Time	
Pacific/Pitcairn	-8 : 00		Pitcairn Standard Time	QN0800UTCS
SystemV/PST8	-8 : 00		Pitcairn Standard Time	
SystemV/PST8PDT	-8 : 00	60	Pacific Standard Time	
US/Pacific	-8 : 00	60	Pacific Standard Time	
US/Pacific-New	-8 : 00	60	Pacific Standard Time	
America/Boise	-7 : 00	60	Mountain Standard Time	
America/Cambridge_Bay	-7 : 00	60	Mountain Standard Time	
America/Chihuahua	-7 : 00	60	Mountain Standard Time	
America/Dawson_Creek	-7 : 00		Mountain Standard Time	
America/Denver	-7 : 00	60	Mountain Standard Time	
America/Edmonton	-7 : 00	60	Mountain Standard Time	
America/Hermosillo	-7 : 00		Mountain Standard Time	
America/Inuvik	-7 : 00	60	Mountain Standard Time	
America/Mazatlan	-7 : 00	60	Mountain Standard Time	
America/Phoenix	-7 : 00		Mountain Standard Time	QN0700MST2, QN0700UTCS
America/Shiprock	-7 : 00	60	Mountain Standard Time	
America/Yellowknife	-7 : 00	60	Mountain Standard Time	
Canada/Mountain	-7 : 00	60	Mountain Standard Time	
Etc/GMT+7	-7 : 00		GMT-07:00	
MST	-7 : 00	60	Mountain Standard Time	QN0700MST, QN0700T
MST7MDT	-7 : 00	60	Mountain Standard Time	
Mexico/BajaSur	-7 : 00	60	Mountain Standard Time	
Navajo	-7 : 00	60	Mountain Standard Time	
PNT	-7 : 00	60	Mountain Standard Time	
SystemV/MST7	-7 : 00		Mountain Standard Time	
SystemV/MST7MDT	-7 : 00	60	Mountain Standard Time	
UA/Arizona	-7 : 00		Mountain Standard Time	
US/Mountain	-7 : 00	60	Mountain Standard Time	
America/Belize	-6 : 00		Central Standard Time	
America/Cancun	-6 : 00	60	Central Standard Time	
America/Chicago	-6 : 00	60	Central Standard Time	

Time zone ID	Raw offset (Hours : Minutes)	DST offset (Minutes)	Display name	QTIMZON variable (i5/OS only)
America/Costa_Rica	-6 : 00		Central Standard Time	QN0600UTCS
America/El_Salvador	-6 : 00		Central Standard Time	
America/Guatemala	-6 : 00		Central Standard Time	
America/Managua	-6 : 00		Central Standard Time	
America/Menominee	-6 : 00	60	Central Standard Time	
America/Merida	-6 : 00	60	Central Standard Time	
America/Mexico_City	-6 : 00	60	Central Standard Time	
America/Monterrey	-6 : 00	60	Central Standard Time	
America/North_Dakota/Center	-6 : 00	60	Central Standard Time	
America/Rainy_River	-6 : 00	60	Central Standard Time	
America/Rankin_Inlet	-6 : 00	60	Central Standard Time	
America/Regina	-6 : 00		Central Standard Time	
America/Swift_Current	-6 : 00		Central Standard Time	
America/Tegucigalpa	-6 : 00		Central Standard Time	
America/Winnipeg	-6 : 00	60	Central Standard Time	
CST	-6 : 00	60	Central Standard Time	QN0600CST, QN600S
CST6CDT	-6 : 00	60	Central Standard Time	
Canada/Central	-6 : 00	60	Central Standard Time	
Canada/East-Saskatchewan	-6 : 00		Central Standard Time	
Canada/Saskatchewan	-6 : 00		Central Standard Time	
Chile/EasterIsland	-6 : 00	60	Easter Is.Time	
Etc/GMT+6	-6 : 00		GMT-06:00	
Mexico/General	-6 : 00	60	Central Standard Time	
Pacific/Easter	-6 : 00	60	Easter Is. Time	
Pacific/Galapagos	-6 : 00		Galapagos Time	
Pacific/Easter	-6 : 00	60	Easter Is. Time	
Pacific/Galapagos	-6 : 00		Galapagos Time	
SystemV/CST6	-6 : 00		Central Standard Time	
SystemV/CST6CDT	-6 : 00	60	Central Standard Time	
US/Central	-6 : 00	60	Central Standard Time	
America/Bogota	-5 : 00		Colombia Time	
America/Cayman	-5 : 00		Eastern Standard Time	
America/Detroit	-5 : 00	60	Eastern Standard Time	
America/Eirunepe	-5 : 00		Acre Time	
America/Fort_Wayne	-5 : 00		Eastern Standard Time	
America/Grand_Turk	-5 : 00	60	Eastern Standard Time	
America/Guayaquil	-5 : 00		Ecuador Time	
America/Havana	-5 : 00	60	Central Standard Time	
America/Indiana/Indianapolis	-5 : 00		Eastern Standard Time	

Time zone ID	Raw offset (Hours : Minutes)	DST offset (Minutes)	Display name	QTIMZON variable (i5/OS only)
America/Indiana/Knox	-5 : 00		Eastern Standard Time	
America/Indiana/Marengo	-5 : 00		Eastern Standard Time	
America/Indiana/Vevay	-5 : 00		Eastern Standard Time	
America/Indianapolis	-5 : 00		Eastern Standard Time	QN0500UTCS
America/Iqaluit	-5 : 00	60	Eastern Standard Time	
America/Jamaica	-5 : 00		Eastern Standard Time	
America/Kentucky/Louisville	-5 : 00	60	Eastern Standard Time	
America/Kentucky/Monticello	-5 : 00	60	Eastern Standard Time	
America/Knox_IN	-5 : 00		Eastern Standard Time	
America/Lima	-5 : 00		Peru Time	
America/Louisville	-5 : 00	60	Eastern Standard Time	
America/Montreal	-5 : 00	60	Eastern Standard Time	
America/Nassau	-5 : 00	60	Eastern Standard Time	
America/New_York	-5 : 00	60	Eastern Standard Time	
America/Nipigon	-5 : 00	60	Eastern Standard Time	
America/Panama	-5 : 00		Eastern Standard Time	
America/Pangnirtung	-5 : 00	60	Eastern Standard Time	
America/Port-au-Prince	-5 : 00		Eastern Standard Time	
America/Porto_Acre	-5 : 00		Acre Time	
America/Rio_Branco	-5 : 00		Acre Time	
America/Thunder_Bay	-5 : 00	60	Eastern Standard Time	
Brazil/Acre	-5 : 00		Acre Time	
Canada/Eastern	-5 : 00	60	Eastern Standard Time	
Cuba	-5 : 00	60	Central Standard Time	
EST	-5 : 00	60	Eastern Standard Time	QN0500EST
EST5EDT	-5 : 00	60	Eastern Standard Time	
Etc/GMT+5	-5 : 00		GMT-05:00	
IET	-5 : 00		Eastern Standard Time	QN0500EST2
Jamaica	-5 : 00		Eastern Standard Time	
SystemV/EST5	-5 : 00		Eastern Standard Time	
SystemV/EST5EDT	-5 : 00	60	Eastern Standard Time	
US/East-Indiana	-5 : 00		Eastern Standard Time	
US/Eastern	-5 : 00	60	Eastern Standard Time	
US/Indiana-Starke	-5 : 00		Eastern Standard Time	
US/Michigan	-5 : 00	60	Eastern Standard Time	
America/Anguilla	-4 : 00		Atlantic Standard Time	
America/Antigua	-4 : 00		Atlantic Standard Time	
America/Aruba	-4 : 00		Atlantic Standard Time	
America/Asuncion	-4 : 00	60	Paraguay Time	

Time zone ID	Raw offset (Hours : Minutes)	DST offset (Minutes)	Display name	QTIMZON variable (i5/OS only)
America/Barbados	-4 : 00		Atlantic Standard Time	
America/Boa_Vista	-4 : 00		Amazon Standard Time	
America/Caracas	-4 : 00		Venezuela Time	QN0400UTC2
America/Cuiaba	-4 : 00	60	Amazon Standard Time	
America/Curacao	-4 : 00		Atlantic Standard Time	
America/Dominica	-4 : 00		Atlantic Standard Time	
America/Glace_Bay	-4 : 00	60	Atlantic Standard Time	
America/Goose_Bay	-4 : 00	60	Atlantic Standard Time	
America/Grenada	-4 : 00		Atlantic Standard Time	
America/Guadeloupe	-4 : 00		Atlantic Standard Time	
America/Guyana	-4 : 00		Guyana Time	
America/Halifax	-4 : 00	60	Atlantic Standard Time	
America/La_Paz	-4 : 00		Bolivia Time	
America/Manaus	-4 : 00		Amazon Standard Time	
America/Martinique	-4 : 00		Atlantic Standard Time	
America/Montserrat	-4 : 00		Atlantic Standard Time	
America/Port_of_Spain	-4 : 00		Atlantic Standard Time	
America/Porto_Velho	-4 : 00		Amazon Standard Time	
America/Puerto_Rico	-4 : 00		Atlantic Standard Time	QN0400UTCS
America/Santiago	-4 : 00	60	Chile Time	
America/Santo_Domingo	-4 : 00		Atlantic Standard Time	
America/St_Kitts	-4 : 00		Atlantic Standard Time	
America/St_Lucia	-4 : 00		Atlantic Standard Time	
America/St_Thomas	-4 : 00		Atlantic Standard Time	
America/St_Vincent	-4 : 00		Atlantic Standard Time	
America/Thule	-4 : 00	60	Atlantic Standard Time	
America/Tortola	-4 : 00		Atlantic Standard Time	
America/Virgin	-4 : 00		Atlantic Standard Time	
Antarctica/Palmer	-4 : 00	60	Chile Time	
Atlantic/Bermuda	-4 : 00	60	Atlantic Standard Time	QN0400AST
Atlantic/Stanley	-4 : 00	60	Falkland Is. Time	
Brazil/West	-4 : 00		Amazon Standard Time	
Canada/Atlantic	-4 : 00	60	Atlantic Standard Time	
Chile/Continental	-4 : 00	60	Chile Time	
Etc/GMT+4	-4 : 00		GMT-04:00	
PRT	-4 : 00		Atlantic Standard Time	
SystemV/AST4	-4 : 00		Atlantic Standard Time	
SystemV/AST4ADT	-4 : 00	60	Atlantic Standard Time	
America/St_Johns	-3 : 30	60	Newfoundland Standard Time	

Time zone ID	Raw offset (Hours : Minutes)	DST offset (Minutes)	Display name	QTIMZON variable (i5/OS only)
CNT	-3 : 30	60	Newfoundland Standard Time	QN0330NST
Canada/Newfoundland	-3 : 30	60	Newfoundland Standard Time	
AGT	-3 : 00		Argentine Time	
America/Araguaina	-3 : 00	60	Brazil Time	
America/Belem	-3 : 00		Brazil Time	
America/Buenos_Aires	-3 : 00		Argentine Time	QN0300UTCS
America/Catamarca	-3 : 00		Argentine Time	
America/Cayenne	-3 : 00		French Guiana Time	
America/Cordoba	-3 : 00		Argentine Time	
America/Fortaleza	-3 : 00		Brazil Time	
America/Godthab	-3 : 00	60	Western Greenland Time	
America/Jujuy	-3 : 00		Argentine Time	
America/Maceio	-3 : 00		Brazil Time	
America/Mendoza	-3 : 00		Argentine Time	
America/Miquelon	-3 : 00	60	Pierre & Miquelon Standard Time	
America/Montevideo	-3 : 00		Uruguay Time	
America/Paramaribo	-3 : 00		Suriname Time	
America/Recife	-3 : 00		Brazil Time	
America/Rosario	-3 : 00		Argentine Time	
America/Sao_Paulo	-3 : 00	60	Brazil Time	
Antarctica/Rothera	-3 : 00		Rothera Time	
BET	-3 : 00	60	Brazil Time	QN0300UTC2
Brazil/East	-3 : 00	60	Brazil Time	
Etc/GMT+3	-3 : 00		GMT-03:00	
America/Noronha	-2 : 00		Fernando de Noronha Time	QN0200UTCS
Atlantic/South_Georgia	-2 : 00		South Georgia Standard Time	
Brazil/DeNoronha	-2 : 00		Fernando de Noronha Time	
Etc/GMT+2	-2 : 00		GMT-02:00	
America/Scoresbysund	-1 : 00	60	Eastern Greenland Time	
Atlantic/Azores	-1 : 00	60	Azores Time	
Atlantic/Cape_Verde	-1 : 00		Cape Verde Time	QN0100UTCS
Etc/GMT+1	-1 : 00		GMT-01:00	
Africa/Abidjan	0 : 00		Greenwich Mean Time	
Africa/Accra	0 : 00		Greenwich Mean Time	
Africa/Bamako	0 : 00		Greenwich Mean Time	

Time zone ID	Raw offset (Hours : Minutes)	DST offset (Minutes)	Display name	QTIMZON variable (i5/OS only)
Africa/Banjul	0 : 00		Greenwich Mean Time	
Africa/Bissau	0 : 00		Greenwich Mean Time	
Africa/Casablanca	0 : 00		Western European Time	
Africa/Conakry	0 : 00		Greenwich Mean Time	
Africa/Dakar	0 : 00		Greenwich Mean Time	
Africa/El_Aaiun	0 : 00		Western European Time	
Africa/Freetown	0 : 00		Greenwich Mean Time	
Africa/Lome	0 : 00		Greenwich Mean Time	
Africa/Monrovia	0 : 00		Greenwich Mean Time	
Africa/Nouakchott	0 : 00		Greenwich Mean Time	
Africa/Ouagadougou	0 : 00		Greenwich Mean Time	
Africa/Sao_Tome	0 : 00		Greenwich Mean Time	
Africa/Timbuktu	0 : 00		Greenwich Mean Time	
America/Danmarkshavn	0 : 00		Greenwich Mean Time	
Atlantic/Canary	0 : 00	60	Western European Time	
Atlantic/Faeroe	0 : 00	60	Western European Time	
Atlantic/Madeira	0 : 00	60	Western European Time	
Atlantic/Reykjavik	0 : 00		Greenwich Mean Time	
Atlantic/St_Helena	0 : 00		Greenwich Mean Time	
Eire	0 : 00	60	Greenwich Mean Time	
Etc/GMT	0 : 00		GMT+00:00	
Etc/GMT+0	0 : 00		GMT+00:00	
Etc/GMT-0	0 : 00		GMT+00:00	
Etc/GMT0	0 : 00		GMT+00:00	
Etc/Greenwich	0 : 00		Greenwich Mean Time	
Etc/UCT	0 : 00		Coordinated Universal Time	
Etc/UTC	0 : 00		Coordinated Universal Time	
Etc/Universal	0 : 00		Coordinated Universal Time	
Etc/Zulu	0 : 00		Coordinated Universal Time	
Europe/Belfast	0 : 00	60	Greenwich Mean Time	
Europe/Dublin	0 : 00	60	Greenwich Mean Time	
Europe/Lisbon	0 : 00	60	Western European Time	
Europe/London	0 : 00	60	Greenwich Mean Time	Q0000GMT2
GB	0 : 00	60	Greenwich Mean Time	
GB-Eire	0 : 00	60	Greenwich Mean Time	
GMT	0 : 00		Greenwich Mean Time	Q0000GMT

Time zone ID	Raw offset (Hours : Minutes)	DST offset (Minutes)	Display name	QTIMZON variable (i5/OS only)
GMT0	0 : 00		GMT+00:00	
Greenwich	0 : 00		Greenwich Mean Time	
Iceland	0 : 00		Greenwich Mean Time	
Portugal	0 : 00	60	Western European Time	
UCT	0 : 00		Coordinated Universal Time	
UTC	0 : 00		Coordinated Universal Time	Q0000UTC
Universal	0 : 00		Coordinated Universal Time	
WET	0 : 00	60	Western European Time	
Zulu	0 : 00		Coordinated Universal Time	
Africa/Algiers	1 : 00		Central European Time	QP0100CET, QP0100UTCS
Africa/Bangui	1 : 00		Western African Time	
Africa/Brazzaville	1 : 00		Western African Time	
Africa/Ceuta	1 : 00	60	Central European Time	
Africa/Douala	1 : 00		Western African Time	
Africa/Kinshasa	1 : 00		Western African Time	
Africa/Lagos	1 : 00		Western African Time	
Africa/Libreville	1 : 00		Western African Time	
Africa/Luanda	1 : 00		Western African Time	
Africa/Malabo	1 : 00		Western African Time	
Africa/Ndjamena	1 : 00		Western African Time	
Africa/Niamey	1 : 00		Western African Time	
Africa/Porto-Novo	1 : 00		Western African Time	
Africa/Tunis	1 : 00		Central European Time	
Africa/Windhoek	1 : 00	60	Western African Time	
Arctic/Longyearbyen	1 : 00	60	Central European Time	
Atlantic/Jan_Mayen	1 : 00	60	Eastern Greenland Time	
CET	1 : 00	60	Central European Time	
ECT	1 : 00	60	Central European Time	QP0100CET3
Etc/GMT-1	1 : 00		GMT+01:00	
Europe/Amsterdam	1 : 00	60	Central European Time	
Europe/Andorra	1 : 00	60	Central European Time	
Europe/Belgrade	1 : 00	60	Central European Time	
Europe/Berlin	1 : 00	60	Central European Time	
Europe/Bratislava	1 : 00	60	Central European Time	
Europe/Brussels	1 : 00	60	Central European Time	
Europe/Budapest	1 : 00	60	Central European Time	

Time zone ID	Raw offset (Hours : Minutes)	DST offset (Minutes)	Display name	QTIMZON variable (i5/OS only)
Europe/Copenhagen	1 : 00	60	Central European Time	
Europe/Gibraltar	1 : 00	60	Central European Time	
Europe/Ljubljana	1 : 00	60	Central European Time	
Europe/Luxembourg	1 : 00	60	Central European Time	
Europe/Madrid	1 : 00	60	Central European Time	
Europe/Malta	1 : 00	60	Central European Time	
Europe/Monaco	1 : 00	60	Central European Time	
Europe/Oslo	1 : 00	60	Central European Time	
Europe/Paris	1 : 00	60	Central European Time	
Europe/Prague	1 : 00	60	Central European Time	
Europe/Rome	1 : 00	60	Central European Time	
Europe/San_Marino	1 : 00	60	Central European Time	
Europe/Sarajevo	1 : 00	60	Central European Time	
Europe/Skopje	1 : 00	60	Central European Time	
Europe/Stockholm	1 : 00	60	Central European Time	
Europe/Tirane	1 : 00	60	Central European Time	
Europe/Vaduz	1 : 00	60	Central European Time	
Europe/Vatican	1 : 00	60	Central European Time	
Europe/Vienna	1 : 00	60	Central European Time	
Europe/Warsaw	1 : 00	60	Central European Time	
Europe/Zagreb	1 : 00	60	Central European Time	
Europe/Zurich	1 : 00	60	Central European Time	QP0100CET2
MET	1 : 00	60	Middle Europe Time	
Poland	1 : 00	60	Central European Time	
ART	2 : 00	60	Eastern European Time	
Africa/Blantyre	2 : 00		Central African Time	
Africa/Bujumbura	2 : 00		Central African Time	
Africa/Cairo	2 : 00	60	Eastern European Time	
Africa/Gaborone	2 : 00		Central African Time	
Africa/Harare	2 : 00		Central African Time	
Africa/Johannesburg	2 : 00		South Africa Standard Time	QP0200SAST
Africa/Kigali	2 : 00		Central African Time	
Africa/Lubumbashi	2 : 00		Central African Time	
Africa/Lusaka	2 : 00		Central African Time	
Africa/Maputo	2 : 00		Central African Time	
Africa/Maseru	2 : 00		South Africa Standard Time	
Africa/Mbabane	2 : 00		South Africa Standard Time	

Time zone ID	Raw offset (Hours : Minutes)	DST offset (Minutes)	Display name	QTIMZON variable (i5/OS only)
Africa/Tripoli	2 : 00		Eastern European Time	
Asia/Amman	2 : 00	60	Eastern European Time	
Asia/Beirut	2 : 00	60	Eastern European Time	
Asia/Damascus	2 : 00	60	Eastern European Time	
Asia/Gaza	2 : 00	60	Eastern European Time	
Asia/Istanbul	2 : 00	60	Eastern European Time	
Asia/Jerusalem	2 : 00	60	Israel Standard Time	
Asia/Nicosia	2 : 00	60	Eastern European Time	
Asia/Tel_Aviv	2 : 00	60	Israel Standard Time	
CAT	2 : 00		Central African Time	
EET	2 : 00	60	Eastern European Time	QP0200EET
Egypt	2 : 00	60	Eastern European Time	
Etc/GMT-2	2 : 00		GMT+02:00	
Europe/Athens	2 : 00	60	Eastern European Time	
Europe/Bucharest	2 : 00	60	Eastern European Time	
Europe/Chisinau	2 : 00	60	Eastern European Time	
Europe/Helsinki	2 : 00	60	Eastern European Time	
Europe/Istanbul	2 : 00	60	Eastern European Time	
Europe/Kaliningrad	2 : 00	60	Eastern European Time	
Europe/Kiev	2 : 00	60	Eastern European Time	
Europe/Minsk	2 : 00	60	Eastern European Time	
Europe/Nicosia	2 : 00	60	Eastern European Time	
Europe/Riga	2 : 00	60	Eastern European Time	
Europe/Simferopol	2 : 00	60	Eastern European Time	
Europe/Sofia	2 : 00	60	Eastern European Time	
Europe/Tallinn	2 : 00	60	Eastern European Time	QP0200EET2, QP0200UTCS
Europe/Tiraspol	2 : 00	60	Eastern European Time	
Europe/Uzhgorod	2 : 00	60	Eastern European Time	
Europe/Vilnius	2 : 00	60	Eastern European Time	
Europe/Zaporozhye	2 : 00	60	Eastern European Time	
Israel	2 : 00	60	Israel Standard Time	
Libya	2 : 00		Eastern European Time	
Turkey	2 : 00	60	Eastern European Time	
Africa/Addis_Ababa	3 : 00		Eastern African Time	QP0300UTCS
Africa/Asmera	3 : 00		Eastern African Time	
Africa/Dar_es_Salaam	3 : 00		Eastern African Time	
Africa/Djibouti	3 : 00		Eastern African Time	
Africa/Kampala	3 : 00		Eastern African Time	
Africa/Khartoum	3 : 00		Eastern African Time	

Time zone ID	Raw offset (Hours : Minutes)	DST offset (Minutes)	Display name	QTIMZON variable (i5/OS only)
Africa/Mogadishu	3 : 00		Eastern African Time	
Africa/Nairobi	3 : 00		Eastern African Time	
Antarctica/Syowa	3 : 00		Syowa Time	
Asia/Aden	3 : 00		Arabia Standard Time	
Asia/Baghdad	3 : 00	60	Arabia Standard Time	
Asia/Bahrain	3 : 00		Arabia Standard Time	
Asia/Kuwait	3 : 00		Arabia Standard Time	
Asia/Qatar	3 : 00		Arabia Standard Time	
Asia/Riyadh	3 : 00		Arabia Standard Time	
EAT	3 : 00		Eastern African Time	
Etc/GMT-3	3 : 00		GMT+03:00	
Europe/Moscow	3 : 00	60	Moscow Standard Time	
Indian/Antananarivo	3 : 00		Eastern African Time	
Indian/Comoro	3 : 00		Eastern African Time	
Indian/Mayotte	3 : 00		Eastern African Time	
W-SU	3 : 00	60	Moscow Standard Time	
Asia/Riyadh87	3 : 07		GMT+03:07	
Asia/Riyadh88	3 : 07		GMT+03:07	
Asia/Riyadh89	3 : 07		GMT+03:07	
Mideast/Riyadh87	3 : 07		GMT+03:07	
Mideast/Riyadh88	3 : 07		GMT+03:07	
Mideast/Riyadh89	3 : 07		GMT+03:07	
Asia/Tehran	3 : 30	60	Iran Standard Time	
Iran	3 : 30	60	Iran Standard Time	
Asia/Aqtau	4 : 00	60	Aqtau Time	QP0400UTC2
Asia/Baku	4 : 00	60	Azerbaijan Time	
Asia/Dubai	4 : 00		Gulf Standard Time	QP0400UTCS
Asia/Muscat	4 : 00		Gulf Standard Time	
Asia/Oral	4 : 00	60	Oral Time	
Asia/Tbilisi	4 : 00	60	Georgia Time	
Asia/Yerevan	4 : 00	60	Armenia Time	
Etc/GMT-4	4 : 00		GMT+04:00	
Europe/Samara	4 : 00	60	Samara Time	
Indian/Mahe	4 : 00		Seychelles Time	
Indian/Mauritius	4 : 00		Mauritius Time	
Indian/Reunion	4 : 00		Reunion Time	
NET	4 : 00	60	Armenia Time	
Asia/Kabul	4 : 30		Afghanistan Time	
Asia/Aqtobe	5 : 00	60	Aqtobe Time	QP0500UTC2

Time zone ID	Raw offset (Hours : Minutes)	DST offset (Minutes)	Display name	QTIMZON variable (i5/OS only)
Asia/Ashgabat	5 : 00		Turkmenistan Time	
Asia/Ashkhabad	5 : 00		Turkmenistan Time	
Asia/Bishkek	5 : 00	60	Kirgizstan Time	
Asia/Dushanbe	5 : 00		Tajikistan Time	
Asia/Karachi	5 : 00		Pakistan Time	QP0500UTCS
Asia/Samarkand	5 : 00		Turkmenistan Time	
Asia/Tashkent	5 : 00		Uzbekistan Time	
Asia/Yekaterinburg	5 : 00	60	Yekaterinburg Time	
Etc/GMT-5	5 : 00		GMT+05:00	
Indian/Kerguelen	5 : 00		French Southern & Antarctic Lands Time	
Indian/Maldives	5 : 00		Maldives Time	
PLT	5 : 00		Pakistan Time	
Asia/Calcutta	5 : 30		India Standard Time	
IST	5 : 30		India Standard Time	QP0530IST
Asia/Katmandu	5 : 45		Nepal Time	
Antarctica/Mawson	6 : 00		Mawson Time	
Antarctica/Vostok	6 : 00		Vostok Time	
Asia/Almaty	6 : 00	60	Alma-Ata Time	QP0600UTC2
Asia/Colombo	6 : 00		Sri Lanka Time	
Asia/Dacca	6 : 00		Bangladesh Time	
Asia/Dhaka	6 : 00		Bangladesh Time	QP0600UTCS
Asia/Novosibirsk	6 : 00	60	Novosibirsk Time	
Asia/Omsk	6 : 00	60	Omsk Time	
Asia/Qyzylorda	6 : 00	60	Qyzylorda Time	
Asia/Thimbu	6 : 00		Bhutan Time	
Asia/Thimphu	6 : 00		Bhutan Time	
BST	6 : 00		Bangladesh Time	
Etc/GMT-6	6 : 00		GMT+06:00	
Indian/Chagos	6 : 00		Indian Ocean Territory Time	
Asia/Rangoon	6 : 30		Myanmar Time	
Indian/Cocos	6 : 30		Cocos Islands Time	
Antarctica/Davis	7 : 00		Davis Time	
Asia/Bangkok	7 : 00		Indochina Time	
Asia/Hovd	7 : 00		Hovd Time	
Asia/Jakarta	7 : 00		West Indonesia Time	QP0700WIB
Asia/Krasnoyarsk	7 : 00	60	Krasnoyarsk Time	
Asia/Phnom_Penh	7 : 00		Indochina Time	
Asia/Pontianak	7 : 00		West Indonesia Time	

Time zone ID	Raw offset (Hours : Minutes)	DST offset (Minutes)	Display name	QTIMZON variable (i5/OS only)
Asia/Saigon	7 : 00		Indochina Time	QP0700UTCS
Asia/Vientiane	7 : 00		Indochina Time	
Etc/GMT-7	7 : 00		GMT+07:00	
Indian/Christmas	7 : 00		Christmas Island Time	
VST	7 : 00		Indochina Time	
Antarctica/Casey	8 : 00		Western Standard Time (Australia)	
Asia/Brunei	8 : 00		Brunei Time	
Asia/Chongqing	8 : 00		China Standard Time	
Asia/Chungking	8 : 00		China Standard Time	
Asia/Harbin	8 : 00		China Standard Time	
Asia/Hong_Kong	8 : 00		Hong Kong Time	QP0800JIST, QP0800UTCS
Asia/Irkutsk	8 : 00	60	Irkutsk Time	
Asia/Kashgar	8 : 00		China Standard Time	
Asia/Kuala_Lumpur	8 : 00		Malaysia Time	
Asia/Kuching	8 : 00		Malaysia Time	
Asia/Macao	8 : 00		China Standard Time	
Asia/Macau	8 : 00		China Standard Time	
Asia/Makassar	8 : 00		Central Indonesia Time	
Asia/Manila	8 : 00		Philippines Time	
Asia/Shanghai	8 : 00		China Standard Time	
Asia/Singapore	8 : 00		Singapore Time	
Asia/Taipei	8 : 00		China Standard Time	
Asia/Ujung_Pandang	8 : 00		Central Indonesia Time	QP0800WITA
Asia/Ulaanbaatar	8 : 00		Ulaanbaatar Time	
Asia/Ulan_Bator	8 : 00		Ulaanbaatar Time	
Asia/Urumqi	8 : 00		China Standard Time	
Australia/Perth	8 : 00		Western Standard Time (Australia)	QP0800AWST
Australia/West	8 : 00		Western Standard Time (Australia)	
CTT	8 : 00		China Standard Time	QP0800BST
Etc/GMT-8	8 : 00		GMT+08:00	
Hongkong	8 : 00		Hong Kong Time	
PRC	8 : 00		China Standard Time	
Singapore	8 : 00		Singapore Time	
Asia/Choibalsan	9 : 00		Choibalsan Time	
Asia/Dili	9 : 00		East Timor Time	
Asia/Jayapura	9 : 00		East Indonesia Time	QP0900WIT

Time zone ID	Raw offset (Hours : Minutes)	DST offset (Minutes)	Display name	QTIMZON variable (i5/OS only)
Asia/Pyongyang	9 : 00		Korea Standard Time	
Asia/Seoul	9 : 00		Korea Standard Time	QP0900KST
Asia/Tokyo	9 : 00		Japan Standard Time	QP0900UTCS
Asia/Yakutsk	9 : 00	60	Yakutsk Time	
Etc/GMT-9	9 : 00		GMT+09:00	
JST	9 : 00		Japan Standard Time	QP0900JST
Japan	9 : 00		Japan Standard Time	
Pacific/Palau	9 : 00		Palau Time	
ROK	9 : 00		Korea Standard Time	
ACT	9 : 30		Central Standard Time (Northern Territory)	
Australia/Adelaide	9 : 30	60	Central Standard Time (South Australia)	QP0930ACST
Australia/Broken_Hill	9 : 30	60	Central Standard Time (South Australia/New South Wales)	
Australia/Darwin	9 : 30		Central Standard Time (Northern Territory)	
Australia/North	9 : 30		Central Standard Time (Northern Territory)	
Australia/South	9 : 30	60	Central Standard Time (South Australia)	
Australia/Yancowinna	9 : 30	60	Central Standard Time (South Australia/New South Wales)	
AET	10 : 00	60	Eastern Standard Time (New South Wales)	QP1000AEST
Antarctica/DumontDUrville	10 : 00		Dumont-d'Urville Time	
Asia/Sakhalin	10 : 00	60	Sakhalin Time	
Asia/Vladivostok	10 : 00	60	Vladivostok Time	
Australia/ACT	10 : 00	60	Eastern Standard Time (New South Wales)	
Australia/Brisbane	10 : 00		Eastern Standard Time (Queensland)	
Australia/Canberra	10 : 00	60	Eastern Standard Time (New South Wales)	
Australia/Hobart	10 : 00	60	Eastern Standard Time (Tasmania)	
Australia/Lindeman	10 : 00		Eastern Standard Time (Queensland)	
Australia/Melbourne	10 : 00	60	Eastern Standard Time (Victoria)	
Australia/NSW	10 : 00	60	Eastern Standard Time (New South Wales)	

Time zone ID	Raw offset (Hours : Minutes)	DST offset (Minutes)	Display name	QTIMZON variable (i5/OS only)
Australia/Queensland	10 : 00		Eastern Standard Time (Queensland)	
Australia/Sydney	10 : 00	60	Eastern Standard Time (New South Wales)	
Australia/Tasmania	10 : 00	60	Eastern Standard Time (Tasmania)	
Australia/Victoria	10 : 00	60	Eastern Standard Time (Victoria)	
Etc/GMT-10	10 : 00		GMT+10:00	
Pacific/Guam	10 : 00		Chamorro Standard Time	QP1000UTCS
Pacific/Port_Moresby	10 : 00		Papua New Guinea Time	
Pacific/Saipan	10 : 00		Chamorro Standard Time	
Pacific/Truk	10 : 00		Truk Time	
Pacific/Yap	10 : 00		Yap Time	
Australia/LHI	10 : 30	30	Load Howe Standard Time	
Australia/Lord_Howe	10 : 30	30	Load Howe Standard Time	
Asia/Magadan	11 : 00	60	Magadan Time	
Etc/GMT-11	11 : 00		GMT+11:00	
Pacific/Efate	11 : 00		Vanuatu Time	
Pacific/Guadalcanal	11 : 00		Solomon Is. Time	QP1100UTCS
Pacific/Kosrae	11 : 00		Kosrae Time	
Pacific/Noumea	11 : 00		New Caledonia Time	
Pacific/Ponape	11 : 00		Ponape Time	
SST	11 : 00		Solomon Is. Time	
Pacific/Norfolk	11 : 30		Norfolk Time	
Antarctica/McMurdo	12 : 00	60	New Zealand Standard Time	
Antarctica/South_Pole	12 : 00	60	New Zealand Standard Time	
Asia/Anadyr	12 : 00	60	Anadyr Time	
Asia/Kamchatka	12 : 00	60	Petropavlovsk-Kamchatski Time	
Etc/GMT-12	12 : 00		GMT+12:00	
Kwajalein	12 : 00		Marshall Islands Time	
NST	12 : 00	60	New Zealand Standard Time	QP1200NZST
NZ	12 : 00	60	New Zealand Standard Time	
Pacific/Auckland	12 : 00	60	New Zealand Standard Time	

Time zone ID	Raw offset (Hours : Minutes)	DST offset (Minutes)	Display name	QTIMZON variable (i5/OS only)
Pacific/Fiji	12 : 00		Fiji Time	QN1200UTCS, QP1200UTCS
Pacific/Funafuti	12 : 00		Tuvalu Time	
Pacific/Kwajalein	12 : 00		Marshall Islands Time	
Pacific/Majuro	12 : 00		Marshall Islands Time	
Pacific/Nauru	12 : 00		Nauru Time	
Pacific/Tarawa	12 : 00		Gilbert Is. Time	
Pacific/Wake	12 : 00		Wake Time	
Pacific/Wallis	12 : 00		Wallis & Futuna Time	
NZ-CHAT	12 : 45	60	Chatham Standard Time	
Pacific/Chatham	12 : 45	60	Chatham Standard Time	QP1245UTCS
Etc/GMT-13	13 : 00		GMT+13:00	
Pacific/Enderbury	13 : 00		Phoenix Is. Time	
Pacific/Tongatapu	13 : 00		Tonga Time	
Etc/GMT-14	14 : 00		GMT+14:00	
Pacific/Kiritimati	14 : 00		Line Is. Time	

Web module or application server stops processing requests

Use this information to help determine why a Web module or application server has stopped processing new requests.

If an application server's process spontaneously closes, or its Web modules stop responding to new requests:

- Isolate the problem by installing Web modules on different servers, if possible.
- You can use the Tivoli performance viewer to determine which resources have reached their maximum capacity, such as Java heap memory (indicating a possible memory leak) and database connections. If a particular resource appears to have reached its maximum capacity, review the application code for a possible cause:
  - If database connections are used and never freed, ensure that application code performs a **close()** on any opened **Connection** object within a **finally{}** block.
  - If there is a steady increase in servlet engine threads in use, review application **synchronized** code blocks for possible deadlock conditions.
  - If there is a steady increase in a JVM heap size, review application code for memory leak opportunities, such as static (class-level) collections, that can cause objects to never get garbage-collected.

  See the *Tuning guide* PDF for more information about this tool.
- As an alternative to using the performance viewer to detect memory leak problems, enable verbose garbage collection on the application server. This feature adds detailed statements to the JVM error log file of the application server about the amount of available and in-use memory. To set up verbose garbage collection:
  1. Select **Servers > Application Servers >** *server_name* **>Java and Process Management > Process Definition > Java Virtual Machine**, and enable **Verbose Garbage Collection**.
  2. Stop and restart the application server.
  3. Periodically, or after the application server stops, browse the log file for garbage collection statements. Look for statements beginning with ″allocation failure″. The string indicates that a need for memory allocation has triggered a JVM garbage collection (freeing of unused memory).

Allocation failures themselves are normal and not necessarily indicative of a problem. The allocation failure statement is followed by statements showing how many bytes are needed and how many are allocated.

If there is a steady increase in the total amount of free and used memory (the JVM keeps allocating more memory for itself), or if the JVM becomes unable to allocate as much memory as it needs (indicated by the bytes needed statement), there might be a memory leak.

- If either the performance viewer or verbose garbage collection output indicates that the application server is running out of memory, one of the following problems might be present:
  – There is a memory leak in application code that you must address. To pinpoint the cause of a memory leak, enable the **RunHProf** property on the **Servers > Application Servers >** *server_name* **> Process Definition > Java Virtual Machine** page of the administrative console. *server_name* is the name of the problem application server. Then:
    - Set the **HProf Arguments** field to a value similar to depth=20,file=heapdmp.txt. This value shows exception stacks to a maximum of 20 levels, and saves the heapdump output to the *app_server_root*/bin/heapdmp.txt file.
    - Save the settings.
    - Stop and restart the application server.
    - If possible, reenact the scenario or access the resource that caused the application server's process to spontaneously close, or its Web modules to stop responding to new requests. Then stop the application server. If can not reenact the scenario or access the resource, wait until the problem reoccurs, and then stop the application server.
    - Examine the file into which the heapdump was saved. For example, examine the *app_server_root*/bin/heapdmp.txt file:
      - Search for the string, ″SITES BEGIN″. This finds the location of a list of Java objects in memory, which shows the amount of memory allocated to the objects.
      - The list of Java objects occurs each time there was a memory allocation in the JVM. There is a record of what type of object the memory instantiated and an identifier of a trace stack, listed elsewhere in the dump, that shows the Java method that made the allocation.
      - The list of Java object is in descending order by number of bytes allocated. Depending on the nature of the leak, the problem class should show up near the top of the list, but this is not always the case. Look throughout the list for large amounts of memory or frequent instances of the same class being instantiated. In the latter case, use the ID in the trace stack column to identify allocations occurring repeatedly in the same class and method.
      - Examine the source code indicated in the related trace stacks for the possibility of memory leaks.

  – The default maximum heap size of the application server needs to be increased. For information about JVM configuration, see the JVM settings chapter in the *Administering applications and their environment* PDF book.

  – There is a defect in the WebSphere Application Server product that you must either report, or correct by installing a fix or fix pack from a maintenance download. Contact IBM support.
- If an application server's process spontaneously closes, look for a Java thread dump file. The JVM creates the file in the product directory structure, with a name like javacore[number].txt.
- Force an application to create a thread dump (or javacore). Here is the process for forcing a thread dump, which is different from the process in earlier releases of the product:
  1. Using the wsadmin command prompt, get a handle to the problem application server:
     ```
     wsadmin>set jvm [$AdminControl completeObjectName type=JVM,process=server1,*]
     ```
  2. Generate the thread dump:
     ```
     wsadmin>$AdminControl invoke $jvm dumpThreads
     ```
  3. Look for an output file in the WebSphere Application Server installation root directory with a name like javacore.*date.time.id*.txt.
- Browse the thread dump for clues:
  – If the JVM creates the thread dump as it closes (the thread dump is not manually forced), there might be ″error″ or ″exception information″ strings at the beginning of the file. These strings indicate the thread that caused the application server's process to spontaneously close.

- The thread dump contains a snapshot of each thread in the process, starting in the section labeled ″Full thread dump.″
  - Look for threads with a description that contains ″state:R″. Such threads are active and running when the dump is forced, or the process exited.
  - Look for multiple threads in the same Java application code source location. Multiple threads from the same location might indicate a deadlock condition (multiple threads waiting on a monitor) or an infinite loop, and help identify the application code with the problem.

IBM Support has documents and tools that can save you time gathering information needed to resolve problems as described in Troubleshooting help from IBM. Before opening a problem report, see the Support page:

- http://www.ibm.com/software/webservers/appserv/was/support/

Creating generic servers

A generic server is a server that is managed in the WebSphere administrative domain, although it is not a server that is supplied by WebSphere Application Server. The WebSphere Application Server generic servers function enables you to define a generic server as an application server instance within the WebSphere Application Server administration, and associate it with a non-WebSphere server or process.

There are two basic types of generic application servers:

- Non-Java applications or processes.
- Java applications or processes

Therefore, a generic server can be any server or process that is necessary to support the Application Server environment, including:

- A Java server
- A C or C++ server or process
- A CORBA server
- A Remote Method Invocation (RMI) server

You can use the wsadmin tool or the administrative console to create a generic server.

- **Create a non-Java application as a generic server.** The following steps describe how to use the administrative console to create a non-Java application as a generic application server.
  1. Select **Servers** > **Generic servers**
  2. Click **New**.
  3. Type in a name for the generic server.

     The name must be unique within the WebSphere Application Server environment. It is recommended that you use a naming scheme that makes it easy to distinguish your generic application servers from regular WebSphere Application Servers.
  4. Select a template for the new server. You can use a default application server template for your new server or use an existing application server as a template. The new application server will inherit all properties of the template server. If you create the new server using an existing application server do not enable the option to map applications from the existing server to the new server. This option does not apply for a generic server.
  5. Click **Next**
  6. Click **Finish**. The generic server now appears as an option on the **Generic servers** page in the administrative console.
  7. On the **Generic servers** page, click on the name of the generic server.
  8. Under Additional Properties, click **Process Definition**.
  9. In the Executable name field under General Properties, enter the name of the non-WebSphere Application Server program that is launched when you start this generic server. Executable target

type and Executable target properties are not used for non-Java applications. Executable target type and Executable target properties are only used for Java applications

10. Click **OK**.

- **Create a Java application as a generic server:** The following steps describe how to use the administrative console to create a Java application as a generic application server.
  1. Select **Servers** > **Generic servers**
  2. Click **New**.
  3. Type in a name for the generic server.

     The name must be unique within the application server. It is highly recommended that you use a naming scheme that makes it easy to distinguish your generic application servers from regular Websphere Application Servers. This will enable you to quickly determine whether to use the **Terminate** or **Stop** button in the administrative console to stop specific application server. You must use the **Terminate** button to stop a generic application server.
  4. Click **Next**
  5. Click **Finish**. The generic server now appears as an option on the **Applications Server** page in the administrative console.
  6. Click **Finish**. The generic server now appears as an option on the **Generic servers** page in the administrative console.
  7. On the Generic servers page, click on the name of the generic server.
  8. Under Additional Properties, click **Process Definition**.
  9. In the Executable name field under General Properties, enter the path for the WebSphere Application Server default JVM, ${JAVA_HOME}/bin/java, which is used to run the Java application when you start this generic server.
  10. In the Executable target type field under General Properties, select whether a Java class name, **JAVA_CLASS**, or the name of an executable JAR file, **EXECUTABLE_JAR**, is used as the executable target of this Java process. The default for WebSphere Application Server is **JAVA_CLASS**.
  11. In the Executable target field under General Properties, enter the name of the executable target. Depending on the executable target type, this is either a Java class containing a main() method, or the name of an executable JAR file.) The default for WebSphere Application Server is com.ibm.ws.runtime.WsServer.
  12. Click **OK**.

      Note: If the generic server is to run an application server other than the WebSphere Application Server, leave the Executable name field set to the default value and specify the Java class containing the main function for your application serve in the Executable target field.

After you define a generic server, use the Application Server administrative console to start, stop, and monitor the associated non-WebSphere server or process when stopping or starting the applications that rely on them.

Important: You can use either the **Terminate** or **Stop** buttons in the administrative console to stop any application server, including a generic application server.

Starting and terminating generic application servers

This topic describes how to start and terminate generic servers.

If you create a generic server on a base WebSphere Application Server, you cannot use the base Application Server administrative console to start or terminate this server. You must use the wsadmin tool to manage this server.

1. Start a generic application server.

Use the MBean NodeAgent launchProcess operation of the wsadmin tool to start a generic application server.

a. View the **Status** value and any messages or logs to see whether the generic server starts.

2. Terminate generic servers.

   Use the MBean terminate launchProcess operation of the wsadmin tool to terminate a generic server.

   a. In the administrative console, click **Servers > Application Servers**.

   b. Select the check box beside the name of the generic server, and then click **Terminate**.

      Restriction: The **Stop** and **Stop Immediate** buttons on the administrative console do not work for generic servers.

   c. View the **Status** value and any messages or logs to see whether the generic server terminates.

Configuring transport chains

A transport chain consists of one or more types of channels, each of which supports a different type of I/O protocol, such as TCP or HTTP. Network ports can be shared among all of the channels within a chain. The channel framework function automatically distributes a request arriving on that port to the correct I/O protocol channel for processing.

Ensure that a port is available for the new transport chain. If you need to set up a shared port, you must:

- Use wsadmin commands to create your transport chain.
- Make sure that all channels sharing that port have the same discrimination weight assigned to them.

You need to configure transport chains to provide networking services to such functions as the service integration bus component of IBM service integration technologies, WebSphere Secure Caching Proxy, and the high availability manager core group bridge service.

You can either use the administrative console or wsadmin commands to create a transport chain. If you want to use wsadmin commands, see the *Using the administrative clients* PDF for more information. If you use the administrative console, complete the following steps:

1. In the administrative console, click **Servers > Application servers >** *server_name*, and then select one of the following options, depending on the type of chain you are creating:
   - Under **SIP Container Settings**, click **SIP container transport chains**.
   - Under **Web container settings**, click **Web container transport chains**.
   - Under **Server messaging**, click either **Messaging engine inbound transports** or **WebSphere MQ link inbound transports**.
2. Click **New**. The Create New Transport Chain wizard initializes. During the transport chain creation process, you are asked to:
   - Specify a name for the new chain.
   - Select a transport chain template
   - Select a port, if one is available to which the new transport chain is bound. If a port is not available or you want to define a new port, specify a port name, the host name or IP address for that port, and a valid port number.

   When you click **Finish**, the new transport chain is added to the list of defined transport chains on the **Transport chain** panel.
3. Click the name of a transport chain to view the configuration settings that are in effect for the transport channels contained in that chain. To change any of these settings:

   a. Click the name of the channel whose settings you need to change.

   b. Change the configuration settings. Some of the settings, such as the port number are determined by what is specified for the transport chain when it is created and cannot be changed.

   c. Click on **Custom properties** to set any custom properties that are defined for your system.

4. When you your configuration changes, click **OK**.
5. Stop the application server and start it again.

   You must stop the application server and start it again before your changes take effect.

Update any routines you have that issue a call to start transports during server startup. When a routine issues a call to start transports during server startup, WebSphere Application Server converts the call to a transport channel call.

Transport chains

Transport chains represent a network protocol stack that is used for I/O operations within an application server environment. Transport chains are part of the channel framework function that provides a common networking service for all components, including the service integration bus component of IBM service integration technologies, WebSphere Secure Caching Proxy, and the high availability manager core group bridge service.

A transport chain consists of one or more types of channels, each of which supports a different type of I/O protocol, such as TCP, DCS. or HTTP. Network ports can be shared among all of the channels within a chain. The channel framework function automatically distributes a request arriving on that port to the correct I/O protocol channel for processing.

transition: If you have a routine that issues a call to start transports during server startup, unless you have a mixed-node environment and that server is running in a V5.1 node, WebSphere Application Server converts the call to a transport chain call.

The transport chain configuration settings determine which I/O protocols are supported for that chain. Following are some of the more common types of channels. Custom channels that support requirements unique to a particular customer or environment can also be added to a transport chain.

DCS channel
Used by the core group bridge service, the data replication service (DRS), and the high availability manger to transfer data, objects, or events among application servers.

HTTP inbound channel
Used to enable communication with remote servers. It implements the HTTP 1.0 and 1.1 standards and is used by other channels, such as the Web container channel, to serve HTTP requests and to send HTTP specific information to servlets expecting this type of information.

HTTP inbound channels are used instead of HTTP transports to establish the request queue between a WebSphere Application Server plug-in for Web servers and a Web container in which the Web modules of an application reside.

HTTP proxy inbound channel
Used to handle HTTP requests between a proxy server and application server nodes.

HTTP Tunnel channel
Used to provide client applications with persistent HTTP connections to remote hosts that are either blocked by firewalls or require an HTTP proxy server (including authentication) or both. An HTTP Tunnel channel enables the exchange of application data in the body of an HTTP request or response that is sent to or received from a remote server. An HTTP Tunnel channel also enables client-side applications to poll the remote host and to use HTTP requests to either send data from the client or to receive data from an application server. In either case, neither the client nor the application server is aware that HTTP is being used to exchange the data.

JFAP channel
Used by the Java Message Service (JMS) server to create connections to JMS resources on a service integration bus.

MQ channel
Used in combination with other channels, such as a TCP channel, within the confines of WebSphere MQ support to facilitate communications between a WebSphere System Integration Bus and a WebSphere MQ client or queue manager.

SIP channel
Used to create a bridge in the transport chain between a session initiation protocol (SIP) inbound channel, and a servlet and JavaServer Page engine.

SIP container inbound channel
Used to handle communication between the SIP inbound channel and the SIP servlet container.

SIP inbound channel
Used to handle inbound SIP requests from a remote client.

SSL channel
Used to associate an Secure Sockets Layer (SSL) configuration repertoire with the transport chain. This channel is only available when SSL support is enabled for the transport chain. An SSL configuration repertoire is defined in the administrative console, under security, on the **SSL configuration repertoires > SSL configuration repertoires** page.

TCP channel
Used to provide client applications with persistent connections within a Local Area Network (LAN) when a node uses transmission control protocol (TCP) to retrieve information from a network.

UDP channel
Used to provide client applications with persistent connections within a Local Area Network (LAN) when a node uses user datagram protocol (UDP) to retrieve information from a network.

Web container channel
Used to create a bridge in the transport chain between an HTTP inbound channel and a servlet and JavaServer Page (JSP) engine.

HTTP transport collection

Use this page to view or manage HTTP transports. Transports provide request queues between WebSphere Application Server plug-ins for Web servers and Web containers in which the Web modules of applications reside. When you request an application in a Web browser, the request is passed to the Web server, then along the transport to the Web container.

Important: You can use HTTP transports only on a Version 5.1 node in a mixed WebSphere Application Server environment. You must use HTTP transport channels instead of HTTP transports to handle your HTTP requests on all of your other nodes.

The use of IPv6 (Internet Protocol Version 6) and WS-AT (Web Services Atomic Transactions) are not supported on HTTP transports; they are only supported on HTTP transport channel chains.

To view the HTTP Transport administrative console page, click **Servers > Application Servers >** *server_name* **> Web Container Settings > Web Container > HTTP Transports**.

Host

Specifies the host IP address to bind for transport. If the application server is on a local machine, the host name might be `localhost`.

Port

Specifies the port to bind for transport. The port number can be any port that currently is not in use on the system. The port number must be unique for each application server instance on a given machine.

For i5/OS and distributed platforms, there is no limit to the number of HTTP ports that are allowed per process.

SSL Enabled

Specifies whether to protect connections between the WebSphere plug-in and application server with Secure Sockets Layer (SSL). The default is not to use SSL.

HTTP transport settings

Use this page to view and configure an HTTP transport. The name of the page might be that of an SSL setting such as DefaultSSLSettings. This page will not be visible if you do not have an HTTP transport defined for your system.

Important: You can use HTTP transports only on a V5.1 node in a mixed WebSphere Application Server environment. You must use HTTP transport channels instead of HTTP transports to handle your HTTP requests on all of your other nodes.

The use of IPv6 (Internet Protocol Version 6) and WS-AT (Web Services Atomic Transactions) are not supported on HTTP transports; they are only supported on HTTP transport channel chains.

To view the HTTP Transport panel on the administrative console, click **Servers** > **Application Servers** > *server_name* > **Web Container Settings** > **Web Container** > **HTTP Transports** > *host_name*.

Host

Specifies the host IP address to bind for transport.

If the application server is on a local machine, the host name might be `localhost`.

Data type	String

Port

Specifies the port to bind for transport. Specify a port number between 1 and 65535. The port number must be unique for each application server on a given machine.

Data type	Integer
Range	1 to 65535

SSL Enabled

Specifies whether to protect connections between the WebSphere Application Server plug-in and application server with Secure Sockets Layer (SSL). The default is not to use SSL.

Data type	Boolean
Default	false

SSL

Specifies the Secure Sockets Layer (SSL) settings type for connections between the WebSphere Application Server plug-in and application server. The options include one or more SSL settings defined in the Security Center; for example, DefaultSSLSettings, ORBSSLSettings, or LDAPSSLSettings.

Data type	String
Default	An SSL setting defined in the Security Center

HTTP transport custom properties

Use this page to set custom properties for an HTTP transport.

Important: You can use HTTP transports only on a V5.1 node in a mixed WebSphere Application Server environment. You must use HTTP transport channels instead of HTTP transports to handle your HTTP requests on all of your other nodes.

The use of IPv6 (Internet Protocol Version 6) and WS-AT (Web Services Atomic Transactions) are not supported on HTTP transports; they are only supported on HTTP transport channel chains.

If you are using HTTP transports, you can set the following custom properties on either the Web Container or HTTP Transport Custom Properties page on the administrative console. When set on the Web container Custom Properties page, all transports inherit the properties. Setting the same properties on a transport overrides like settings defined for a Web container.

To specify custom properties for a specific transport on the HTTP Transport:

1. In the administrative console click **Servers > Application Servers >** *server_name* **> Web Container settings > Web Container >HTTP Transport**
2. Select a host.
3. Under **Additional Properties** select **Custom Properties**.
4. On the Custom Properties page, click **New**.
5. On the settings page, enter the property you want to configure in the **Name** field and the value you want to set it to in the **Value** field.
6. Click **Apply** or **OK**.
7. Click **Save** on the console task bar to save your configuration changes.
8. Restart the server.

Following is a list of custom properties provided with the Application Server. These properties are not shown on the settings page for an HTTP transport.

ConnectionIOTimeOut:

Use the `ConnectionIOTimeOut` property to specify how long the J2EE server waits for an I/O operation to complete. Set this variable for each of the HTTP transport definitions on the server. You will need to set this variable for both SSL transport and non-SSL transport. Specifying a value of zero disables the time out function.

Data type	Integer
Default	For the i5/OS and distributed platforms: 5 seconds

ConnectionKeepAliveTimeout:

Use the `ConnectionKeepAliveTimeout` property to specify the maximum number of seconds to wait for the next request on a keep alive connection.

Data type	Integer
Default	For the i5/OS and distributed platforms: 5 seconds

MaxConnectBacklog: This property is only valid for i5/OS and distributed platforms. Use the `MaxConnectBacklog` property to specify the maximum number of outstanding connect requests that the operating system will buffer while it waits for the application server to accept the connections. If a client attempts to connect when this operating system buffer is full, the connect request will be rejected.

Keep in mind that a single client browser might need to open multiple concurrent connections (perhaps 4 or 5); however, also keep in mind that increasing this value consumes more kernel resources. The value of this property is specific to each transport.

Data type	Integer
Default	511

MaxKeepAliveConnections:

This property is only valid for i5/OS and distributed platforms. It is ignored on the z/OS platform because asynchronous I/O sockets are used to maintain connections in that environment.Use the `MaxKeepAliveConnections` property to specify the maximum number of concurrent keep alive (persistent) connections across all HTTP transports. To make a particular transport close connections after a request, you can set MaxKeepAliveConnections to `0` (zero) or you can set KeepAliveEnabled to `false` on that transport.

The Web server plug-in keeps connections open to the application server as long as it can. However, if the value of this property is too small, performance is negatively impacted because the plug-in has to open a new connection for each request instead of sending multiple requests through one connection. The application server might not accept a new connection under a heavy load if there are too many sockets in TIME_WAIT state. If all client requests are going through the Web server plug-in and there are many TIME_WAIT state sockets for port 9080, the application server is closing connections prematurely, which decreases performance. The application server closes the connection from the plug-in, or from any client, for any of the following reasons:

- The client request was an HTTP 1.0 request when the Web server plug-in always sends HTTP 1.1 requests.
- The maximum number of concurrent keep-alives was reached. A keep-alive must be obtained only once for the life of a connection, that is, after the first request is completed, but before the second request can be read.
- The maximum number of requests for a connection was reached, preventing denial of service attacks in which a client tries to hold on to a keep-alive connection forever.
- A time out occurred while waiting to read the next request or to read the remainder of the current request.

Data type	Integer
Default	90% of the maximum number of threads in the Web container thread pool. This prevents all of the threads from being held by keep alive connections so that there are threads available to handle new incoming connect requests.

MaxKeepAliveRequests:

Use the `MaxKeepAliveRequests` property to specify the maximum number of requests which can be processed on a single keep alive connection. This parameter can help prevent denial of service attacks when a client tries to hold on to a keep-alive connection. The Web server plug-in keeps connections open to the application server as long as it can, providing optimum performance.

On the i5/OS and distributed platforms, when this property is set to 0 (zero), the connection stays open as long as the application server is running.

Data type	Integer
Default	100 requests

KeepAliveEnabled: This property is only valid for i5/OS and distributed platforms. Use the `KeepAliveEnabled` property to specify whether or not to keep connections alive

Data type	String

Value	true or false
Default	true

RemoveServerHeader: Use this property to specify whether an existing server header is removed before a response message is sent. If this property is set to `true`, the value specified for the ServerHeaderValue property is ignored.

Data type	String
Value	true or false
Default	false

ServerHeaderValue: Use this property to specify a server header this is added to outgoing response messages if server header is not already provided. This property is ignored if the RemoveServerHeader property is set to true.

Data type	string
Default	WebSphere Application Server/*x.x* *x.x* is the version of WebSphere Application Server that you are using.

SoLingerValue: Use this property to specify, in seconds, the amount, that the socket close operation waits for data contained in the TCP/IP send buffer to be sent. This property is ignored if the UseSoLinger property is set to false.

Data type	Integer
Default	20 seconds

TcpNoDelay: Use this property to set the socket TCP_NODELAY option which enables and disables the use of the TCP Nagle algorithm for connections received on this transport. When this property is set to `true`, use of the Nagle algorithm is disabled.

Data type	String
Value	true or false
Default	true

Trusted: This property is only valid for i5/OS and distributed platforms. Use the `Trusted` property to indicate that the application server can use the private headers that the Web server plug-in adds to requests.

Data type	String
Value	true or false
Default	false

UseSoLinger: Use this property to set the socket SO_LINGER option. This property configures whether the socket close operation waits until all of the data contained in the TCP/IP send buffer is sent before closing a connection. If this property is enabled, and the time expires before the all of the content of the send buffer sent, any data remaining in the send buffer is lost.

The SoLingerValue property is ignored if this property is set to false.

Data type	String
Value	true or false

Default	true

HTTP transport channel custom properties

If you are using an HTTP transport channel, you can add the following custom property to the configuration settings for that HTTP transport channel.

To add a custom property:

1. In the administrative console, click **Application servers >** *server_name* **Web container settings > Web container transport chains >** *chain_name* **> HTTP Inbound Channel > Custom Properties > New**
2. Under **General Properties** specify the name of the custom property in the Name field and a value for this property in the Value field. You can also specify a description of this property in the Description field.
3. Click **Apply** or **OK**.
4. Click **Save** to save your configuration changes.
5. Restart the server.

Following is a list of custom properties provided with the application server. These properties are not shown on the settings page for an HTTP transport channel.

inProcessLogFilenamePrefix

Use the inProcessLogFilenamePrefix property to specify a prefix for the filename of the network log file. Normally, when inprocess optimization is enabled, requests through the inprocess path are logged based on the logging attributes set up for the Web container's network channel chain. You can use this property to add a prefix to the filename of the network log file. This new filename is then used as the filename for the log file for inprocess requests. Requests sent through the inprocess path are logged to this file instead of to the network log file. For example, if the log file for a network transport chain is named .../httpaccess.log, and this property is set to local for the HTTP channel in that chain, the filename of the log file for inprocess requests to the host associated with that chain is .../localhttpaccess.log.

Data type	String

listenBacklog

Use the listenBacklog property to specify the maximum number of outstanding connect requests that the operating system will buffer while it waits for the application server to accept the connections. If a client attempts to connect when this operating system buffer is full, the connect request will be rejected. Set this value to the number of concurrent connections that you would like to allow. Keep in mind that a single client browser might need to open multiple concurrent connection; also keep in mind that increasing this value consumes more kernel resources. The value of this property is specific to each transport.

Data type	Integer
Default	511

HTTP Tunnel transport channel custom property

If you are using an HTTP Tunnel transport channel, you can add the following custom property to the configuration settings for that HTTP Tunnel transport channel.

To add a custom property:

1. In the administrative console, click **Servers > Application servers >** *server_name* **> Ports**. Click on **View associated transports** for the HTTP Tunnel port to whose configuration settings you want to add this custom property.
2. Click **New**.
3. Under **General Properties** specify the name of the custom property in the Name field and a value for this property in the Value field. You can also specify a description of this property in the Description field.
4. Click **Apply** or **OK**.
5. Click **Save** to save your configuration changes.
6. Restart the server.

Following is a description of the custom property that is provided with the application server. This property is not shown on the settings page for an HTTP Tunnel transport channel.

pluginConfigurable

Indicates whether or not the configuration settings for the HTTP Tunnel transport channel are included in the plugin-cfg.xml file for the Web server associated with the application server that is using this channel. Configuration settings for each of the Web container transport channels defined for an application server are automatically included in the plugin-cfg.xml file for the Web server associated with that application server.

Data type	Boolean
Default	False

Web container transport custom properties

Use this page to set custom properties for a Web container transport.

Unless you are not migrating from an previous version of WebSphere Application Server, you must use HTTP transport channels instead of HTTP transports to handle your HTTP requests.

If you are using Web container transports, you can set the following custom properties on the Web Container **Custom Properties** panel on the administrative console. When set on the Web container **Custom Properties** page, all transports inherit the properties. Setting the same properties on a transport overrides like settings defined for a Web container.

To specify custom properties for a specific transport on the HTTP Transport:

1. In the administrative console click **Servers > Application Servers >** *server_name* **> Web Container settings > Web Container**
2. Select a host.
3. Under **Additional Properties** select **Custom Properties**.
4. On the Custom Properties page, click **New.**
5. On the settings page, enter the property you want to configure in the **Name** field and the value you want to set it to in the **Value** field.
6. Click **Apply** or **OK**.
7. Click **Save** on the console task bar to save your configuration changes.
8. Restart the server.

Following is a list of custom properties provided with the Application Server. These properties are not shown on the settings page for a Web container transport.

disableRequestMessageChunking

This custom property disables request message chunking when set to true. All the request body up to `maxRequestMessageBodySize` is buffered in memory.

Value
True or False. When a value is not specified, the default value is false.

maxRequestMessageBodySize

If `disableRequestMessageChunking` is false, this is the maximum amount of request body that is buffered in memory before sending the next chunk to the SR. If `disableRequestMessageChunking` is true, this is the maximum amount of request body data that is buffered in memory before sending the complete request to the SR. An HTTP 404 is returned if `maxRequestMessageBodySize` is exceeded.

Value
If `disableRequestMessageChunking` is false, the default size is 32K and maximum size is 10MB. If `disableRequestMessageChunking` is true, the default size is 10MB and the maximum size is 100MB.

Configuring inbound HTTP request chunking

Inbound HTTP request chunking, is configured at the Web container transport chain level. You can configure each Web container chain to enable or disable chunking. You can also configure the maximum message size for chunking disabled and the maximum chunk size for chunking enabled for each chain.

See the page for details on these settings.

The chains that host the Integrated Solutions console have chunking enabled by default, while all other Web container chains have chunking disabled by default. The reason for this is that there are some limitations in the use of inbound HTTP chunking on WebSphere Application Server for z/OS V6.1.

The following are limitations of inbound HTTP chunking

- The applications that are served by chains with chunking enabled must support chunked HTTP encoding. See RFC 2616 for more information on chunked encoding. If your application does not support chunked encoding, then you must map it to a Web container chain with chunking disabled.
- The current implementation requires a Web application to read the entire request, both headers and body, before sending any response data back to the client. If the Web application does not read the entire request, this results in an error in the servlet as well as an HTTP 500 Internal Server Error returned to the client.

1. In the administrative console click **Servers** > **Application Servers** > *server_name* > **Web Container settings** > **Web Container**
2. Select a host.
3. **Optional:** Enable request message chunking. See the article, “Web container transport custom properties” on page 260 for details on these settings.
   a. Under **Additional Properties** select **Custom Properties**.
   b. On the Custom Properties page, click **New**.
   c. On the settings page, enter the property, **disableRequestMessageChunking** in the **Name** field and the enter the value false in the **Value** field.
   d. Specify the maximum amount of request body that is buffered in memory before sending the next chunk to the SR.
   e. Click **Apply** or **OK**.
4. **Optional:** Disable request message chunking. See the article, “Web container transport custom properties” on page 260 for details on these settings.
   a. Under **Additional Properties** select **Custom Properties**.
   b. On the Custom Properties page, click **New**.

c. On the settings page, enter the property, **disableRequestMessageChunking** in the **Name** field and the enter the value true in the **Value** field.

d. Specify the maximum amount of request body data that is buffered in memory before sending the complete request to the SR.

e. Click **Apply** or **OK**.

5. Click **Save** on the console task bar to save your configuration changes.
6. Restart the server.

Transport chain problems

Review the following topics if you encounter a transport chain problem.

TCP transport channel fails to bind to a specific host/port combination

If a TCP transport channel fails to bind to a specific port, one of the following situations might have occurred:

- You are trying to bind the channel to a port that is already bound to another application, such as another instance of a WebSphere Application Server.
- You are trying to bind to a port that is in a transitional state waiting for closure. This socket must transition to closed before you restart the server. The port might be in TIME_WAIT, FIN_WAIT_2, or CLOSE_WAIT state. Issue the `netstat -a` command from a command prompt to display the state of the port to which you are trying to bind.

  If you need to change the amount of elapse time that must occur before TCP/IP can release a closed connection and reuse its resources, see the *Tuning guide* PDF.

Error message CHFW0030E indicates there is ″No such file or directory,″ HP-UX

If you receive an Error message CHFW0030E that indicates there is no such file or directory, and you are running on an HP-UX operating system, make sure you have the most current patches for that operating system installed.

Deleting a transport chain

Transport chains cannot be deleted the same way that HTTP transports can be deleted. Because you cannot have multiple HTTP transports associated with the same port, when you delete an HTTP transport, you effectively delete the associated port and stop all traffic on that port. However, the process is more complicated for a transport chain because multiple transport chains might be associated with the same port and you do not want to disrupt traffic on transport chains that you are not deleting.

Determine whether you want to delete a particular transport chain or all of the transport chains that are associated with a specific port.

You might have to delete one or more transport chains if you have to delete a port.

To delete a transport chain:

1. In the administrative console, click **Servers** > **Application servers** > *server* > **Ports**.
2. In the list of available ports, locate the port that you want to delete and click **View associated transports** for that port.
3. Select the transport chain you want to delete, and click **Delete**. If you intend to delete the port that is associated with this transport chain, repeat this step for all of the transport chains associated with this port.
4. Click **Save** to save your changes.

If you delete all of the transport chains associated with a port, you can delete the port.

Disabling ports and their associated transport chains

Transport chains cannot be disabled the same way that HTTP transports can be disabled. Because you cannot have multiple HTTP transports associated with the same port, when you disable an HTTP transport, you effectively disable the associated port and stop all traffic on that port. However, the process is more complicated for a port that has associated transport chains because multiple transport chains might be associated with the same port, and you might not want to disrupt traffic on all of the transport chains at the same time.

Determine whether you want to disable a particular transport chain or all of the transport chains that are associated with a specific port.

You might need to disable a transport chain if you want to temporarily stop all incoming traffic on a particular port or on a particular transport chain that is associated with that port.

To disable a specific transport chain:

1. In the administrative console, click **Servers > Application servers >** *server* **> Ports**.
2. In the list of available ports, locate the port that you want to delete and click **View associated transports** for that port.
3. Click the transport chain you want to disable.
4. Unselect the **Enabled** field, and click **OK**. If you want to temporarily stop all of the incoming traffic on a port, repeat this step for all of the transport chains associated with this port.
5. Click **Save** to save your changes.

When you want traffic to resume on these disabled transport chains, repeat the preceding steps for all of the transport chains you disabled, and select the **Enabled** field.

Transport chains collection

Use this page to view or manage transport chains. Transport chains enable communication through transports, or protocol stacks, which are usually socket based.

A transport chain consists of one or more types of channels, each of which supports a different type of I/O protocol, such as TCP or HTTP. Network ports can be shared among all of the channels within a chain. The Channel Framework function automatically distributes a request arriving on that port to the correct I/O protocol channel for processing.

The **Transport chains** page lists the transport chains defined for the selected application server. Transport chains represent network protocol stacks operating within this application server.

To view this administrative console page, click **Servers > Application servers >** *server_name* **> Ports**. Click on **View associated transports** for the port whose transport chains you want to view.

Name

Specifies a unique identifier for the transport chain. The name must consist of alphanumeric or national language characters and can start with a number. The name must be unique within a WebSphere Application Server configuration. Click on the name of a transport chain to change its configuration settings.

Enabled

When set to true, indicates that the transport chain is activated at application server startup.

Host

Specifies the host IP address to bind for transport. If the application server is on a local machine, the host name might be localhost.

Port

Specifies the port to bind for transport. The port number can be any port that currently is not in use on the system, might be localhost or the wildcard character * (an asterisk). The port number must be unique for each application server instance on a given machine

SSL Enabled

When enabled, users are notified that there is a channel that enables Secure Sockets Layer (SSL) in the listed chain. When SSL is enabled, all traffic going through this transport is encrypted and digitally secured.

Transport chain settings

Use this page to view a list of the types of transport channels configured for the selected transport chain. A transport chain consists of one or more types of channels, each of which supports a different type of I/O protocol, such as TCP, HTTP, or DCS.

To view this administrative console page, click **Servers** > **Application servers** > *server_name* > **Ports**. Click on **View associated transports** for the port whose transport chains you want view and then click on the name of a specific chain.

Name

Specifies the name of the selected transport chain.

You can edit this field to rename this transport chain. However, remember that the name must be unique within a WebSphere Application Server configuration.

Enabled

When checked, this transport chain is activated at application server startup.

Transport channels

Lists the transport channels configured for this transport chain and their configuration settings. To change a transport channel's configuration settings, click on the name of that transport channel.

HTTP tunnel transport channel settings

Use this page to view and configure an HTTP tunnel transport channels. Inbound connections sent through this channel are tunneled over HTTP, allowing intermediates to view this data as the body of an HTTP message instead of in its natural format. This type of channel is often used to circumvent firewalls with protocol restrictions.

To view this administrative console page, click **Servers** > **Application servers** > *server_name* > **Ports** . Click on **View associated transports** for the port associated with the HTTP Tunnel transport channel whose settings you want to look at.

Transport channel name

Specifies the name of the HTTP tunnel transport channel.

The name field cannot contain the following characters: `# \ / , : ; " * ? < > | = + & % '`

This name must be unique across all channels in a WebSphere Application Server environment. For example, an HTTP tunnel transport channel and a TCP transport channel cannot have the same name if they reside within the same system.

Data type	string

Discrimination weight

Specifies the priority this channel has in relation to the other channels in this transport chain. This property is only used when port sharing is enabled and the channel chain includes multiple channels to which it might forward data. The channel in the chain with the lowest discrimination weight is the first one given the opportunity to look at incoming data and determine whether or not it owns that data.

Data type	Positive integer
Default	0

HTTP transport channel settings

Use this page to view and configure an HTTP transport channel. This type of transport channel handles HTTP requests from a remote client.

An HTTP transport channel parses HTTP requests and then finds an appropriate application channel to handle the request and send a response.

To view this administrative console page, click **Servers > Application servers >** *server_name* **> Ports** . Click on **View associated transports** for the port associated with the HTTP transport channel whose settings you want to look at.

Transport channel name

Specifies the name of the HTTP transport channel.

The name field cannot contain the following characters: # \ / , : ; " * ? < > | = + & % '

This name must be unique across all channels in a WebSphere Application Server environment. For example, an HTTP transport channel and a TCP transport channel cannot have the same name if they reside within the same system.

Data type	String

Discrimination weight

Specifies the priority this channel has in relation to the other channels in this transport chain. This property is only used when port sharing is enabled and the channel chain includes multiple channels to which it might forward data. The channel in the chain with the lowest discrimination weight is the first one given the opportunity to look at incoming data and determine whether or not it owns that data.

Data type	Positive integer
Default	0

Maximum persistent requests

Specifies the maximum number of persistent (keep-alive) requests that are allowed on a single HTTP connection. If a value of 0 (zero) is specified, only one request is allowed per connection. If a value of -1 is specified, an unlimited number of requests is allowed per connection.

Data type	Integer
Default	100

Use persistent (keep-alive) connections

When selected, the HTTP transport channel, when sending an outgoing HTTP message, uses a persistent (keep-alive) connection instead of a connection that closes after one request or response exchange occurs.

Note: If a value other than 0 is specified for the maximum persistent requests property, the Use persistent (keep-alive) connections property setting is ignored.

The default for this property is selected.

Read timeout

Specifies the amount of time, in seconds, the HTTP transport channel waits for a read request to complete on a socket after the first read request occurs. The read being waited for could be an HTTP body (such as a POST) or part of the headers if they were not all read as part of the first read request on the socket.

Data type	Integer
Default	60 seconds

Write timeout

Specifies the amount of time, in seconds, that the HTTP transport channel waits on a socket for each portion of response data to be transmitted. This timeout usually only occurs in situations where the writes are lagging behind new requests. This can occur when a client has a low data rate or the server’s network interface card (NIC) is saturated with I/O.

Data type	Integer
Default	60 seconds

Persistent timeout

Specifies the amount of time, in seconds, that the HTTP transport channel allows a socket to remain idle between requests.

Data type	Integer
Default	30 seconds

Enable access and error logging

When selected, the HTTP transport channel performs NCSA access and error logging. Enabling NCSA access and error logging slows server performance.

To configure NCSA access and error logging, click **HTTP error and NCSA access logging** under **Related Items**. Even if HTTP error and NCSA access logging is configured, it is not enabled unless the Enable access and error logging property is selected.

The default value for the Enable access and error logging property is not selected.

TCP transport channel settings

Use this page to view and configure a TCP transport channels. This type of transport channel handles inbound TCP/IP requests from a remote client.

To view this administrative console page, click **Servers > Application servers >** *server_name* **> Ports >** . Click on **View associated transports** for the port associated with the TCP transport channel whose settings you want to view.

Transport channel name

Specifies the name of the TCP transport channel.

The name field cannot contain the following characters: `# \ / , : ; ″ * ? < > | = + & % ’`

This name must be unique across all channels in a WebSphere Application Server environment. For example, an HTTP proxy inbound channel and a TCP transport channel cannot have the same name if they reside within the same system.

Data type	string

Port

Specifies the TCP/IP port this transport channel uses to establish connections between a client and an application server. The TCP transport channel binds to the hostnames and ports listed for the Port property. You can specify the wildcard * (an asterisk), for the hostname if you want this channel to listen to all hosts that are available on this system. However, before specifying the wildcard value, make sure this TCP transport channel does not have to bind to a specific hostname.

Data type	string

Thread pool

This field only applies for i5/OS and distributed platforms. Select from the drop-down list of available thread pools the thread pool you want the TCP transport channel to use when dispatching work.

Maximum open connections

Specifies the maximum number of connections that can be open at one time.

Data type	Integer between 1 and 20,000 inclusive
Default	20,000

Inactivity timeout

Specifies the amount of time, in seconds, that the TCP transport channel waits for a read or write request to complete on a socket.

Note: The value specified for this property might be overridden by the wait times established for channels above this channel. For example, the wait time established for an HTTP transport channel overrides the value specified for this property for every operation except the initial read on a new socket.

Data type	Integer
Default	60 seconds

Address exclude list

Lists the IP addresses that are not allowed to make inbound connections.

Use a comma to separate the IPv4 or IPv6 or both addresses to which you want to deny access on inbound TCP connection requests.

All four numeric values in an IPv4 address must be represented by a number or the wildcard character * (an asterisk).

Following are examples of valid IPv4 addresses that can be included in an Address exclude list:

```
*.1.255.0
254.*.*.9
1.*.*.*
```

All eight numeric values of an IPv6 address must be represented by a number or the wildcard character * (an asterisk). No shortened version of the IPv6 address should be used. Even though a shortened version is processed with no error given, it does not function correctly in this list. Each numeric entry should be a 1- 4 digit hexadecimal number.

Following are examples of valid IPv6 addresses that can be included in an Address exclude list:

```
0:*:*:0:007F:0:0001:0001
F:FF:FFF:FFFF:1:01:001:0001
1234:*:4321:*:9F9f:*:*:0000
```

Note: The **Address include list** and **Host name include list** are processed before the **Address exclude list** and the **Host name exclude list**. If all four lists are defined:

- An address that is defined on either inclusion list will be allowed access provided it is not included on either of the exclusion lists.
- If an address is included in both an inclusion list and in an exclusion list, it will not be allowed access.

Address include list

Lists the IP addresses that are allowed to make inbound connections. Use a comma to separate the IPv4 or IPv6 or both addresses to which you want to grant access on inbound TCP connection requests.

All four numeric values in an IPv4 address must be represented by a number or the wildcard character * (an asterisk).

Following are examples of valid IP addresses that can be included in an Address include list:

```
 *.1.255.0
 254.*.*.9
 1.*.*.*
```

All eight numeric values of an IPv6 address must be represented by a number or the wildcard character * (an asterisk). No shortened version of the IPv6 address should be used. Even though a shortened version is processed with no error given, it does not function correctly in this list. Each numeric entry should be a 1- 4 digit hexadecimal number.

Following are examples of valid IPv6 addresses that can be included in an **Address include list**:

```
0:*:*:0:007F:0:0001:0001
F:FF:FFF:FFFF:1:01:001:0001
1234:*:4321:*:9F9f:*:*:0000
```

Note: The Address include list and the Host name include list are processed before the Address exclude list and the Host name exclude list. If all four lists are defined:

- An address that is defined on either inclusion list will be allowed access provided it is not included on either of the exclusion lists.
- If an address is included in both an inclusion list and in an exclusion list, it will not be allowed access.

Host name exclude list

List the host names that are not allowed to make connections. Use a comma to separate the URL addresses to which you want to deny access on inbound TCP connection requests.

A URL address can start with the wildcard character * (an asterisk) followed by a period; for example, `*.Rest.Of.Address`. If a period does not follow the wildcard character, the asterisk will be treated as a normal non-wildcard character. The wildcard character cannot appear any where else in the address. For example, `ibm.*.com` is not a valid hostname.

Following are examples of valid URL addresses that can be included in a Host name exclude list:

```
*.ibm.com
www.ibm.com
*.com
```

Note: The Address include list and Host name include list are processed before the Address exclude list and the Host name exclude list. If all four lists are defined:

- An address that is defined on either inclusion list will be allowed access provided it is not included on either of the exclusion lists.
- If an address is included in both an inclusion list and in an exclusion list, it is not allowed access.

Host name include list

Lists the host names that are allowed to make inbound connections. Use a comma to separate the URL addresses to which you want to grant access on inbound TCP connection requests.

A URL address can start with the wildcard character * (an asterisk) followed by a period; for example, `*.Rest.Of.Address`. If a period does not follow the wildcard character, the asterisk will be treated as a normal non-wildcard character. The wildcard character cannot appear any where else in the address. For example, `ibm.*.com` is not a valid hostname.

Following are examples of valid URL addresses that can be included in a hostname include list:

```
*.ibm.com
www.ibm.com
*.com
```

Note: The Address include list and Host name include list are processed before the Address exclude list and the Host name exclude list. If all four lists are defined:

- An address that is defined on either inclusion list will be allowed access provided it is not included on either of the exclusion lists.
- If an address is included in both an inclusion list and in an exclusion list, it is not allowed access.

DCS transport channel settings

Use this page to view and configure an DCS transport channels. This type of transport channel handles inbound Distribution and Consistency Services (DCS) messages.

By default, two channel transport chains are defined for an application server that contains a DCS channel:

- The chain named DCS contains a TCP and a DCS channel.
- The chain named DCS-Secure contains a TCP, an SSL, and a DCS channel.

Both of these chains terminate in, or use the same TCP channel instance. This TCP channel is associated with the DCS_UNICAST_ADDRESS port and is not used in any other transport chain. One instance of an SSL channel is reserved for use in the DCS-Secure chain. It also is not used in any other transport chains.

To view this administrative console page, click **Servers** > **Application servers** > *server_name* > **Ports** > . Click **View associated transports** for the port associated with the DCS transport channel whose settings you want to look at.

Transport channel name

Specifies the name of the DCS transport channel.

The name field cannot contain the following characters: **# \ / , : ; " * ? < > | = + & % '**

This name must be unique across all channels in a WebSphere Application Server environment. For example, a DCS transport channel and a TCP transport channel cannot have the same name if they reside within the same system.

Data type	String

Discrimination weight

Specifies the priority this channel has in relation to the other channels in this transport chain. This property is only used when port sharing is enabled and the channel chain includes multiple channels to which it might forward data. The channel in the chain with the lowest discrimination weight is the first one given the opportunity to look at incoming data and determine whether or not it owns that data.

Data type	Positive integer
Default	0

SSL inbound channel

Use this page to determine which SSL inbound channel options to specify for the application server.

To view this administrative console page:

1. Click **Servers** > **Application Servers** > *server_name*.
2. Under Container settings, click **Web container transport chains** > *secure_transport_chain*.
3. Under Transport channels, click **SSL Inbound Channel (SSL_1)**.

Transport Channel Name

Specifies the name of the SSL inbound channel.

The name field cannot contain the following characters: **# \ / , : ; ″ * ? < > | = + & % ’**

This name must be unique across all channels in an application server environment. For example, an SSL inbound channel and a TCP transport channel cannot have the same name if they reside within the same system.

Data type	String

Discrimination weight

Specifies the priority this channel has in relation to the other channels in this transport chain. This property is only used when port sharing is enabled and the channel chain includes multiple channels to which it might forward data. The channel in the chain with the lowest discrimination weight is the first one given the opportunity to look at incoming data and determine whether or not it owns that data.

Data type	Positive integer
Default	0

Centrally managed

Specifies that the selection of an SSL configuration is based upon the outbound topology view for the Java Naming and Directory Interface (JNDI) platform.

Centrally managed configurations support one location to maintain SSL configurations rather than spreading them across the configuration documents.

Default:	Enabled

Specific to this endpoint

Specifies the SSL configuration alias that you want to use for outbound SSL communications.

This option overrides the centrally managed configuration for the JNDI (LDAP) protocol.

Session Initiation Protocol (SIP) inbound channel settings

Use this page to configure the SIP inbound channel settings.

To view this administrative console page, click **Servers** > **Application servers** > *server_name* > **Ports** . Click on **View associated transports** for the port associated with the UDP transport channel whose settings you want to view.

Transport channel name

Specifies the name of the SIP inbound transport channel.

The name field cannot contain the following characters: # \ / , : ; " * ? < > | = + & % '

This name must be unique across all channels in a WebSphere Application Server environment. For example, a SIP transport channel and a TCP transport channel cannot have the same name if they reside within the same system.

Default	UDP_(n) where (n) represents the number of instances of this channel in the system

Discrimination weight

Specifies the priority this channel has in relation to the other channels in this transport chain. This property is only used when port sharing is enabled and the channel chain includes multiple channels to which it might forward data. The channel in the chain with the lowest discrimination weight is the first one given the opportunity to look at incoming data and determine whether or not it owns that data.

Data type	Positive integer
Default	10

Session Initiation Protocol (SIP) container inbound channel settings

Use this page to configure the SIP container inbound channel settings.

To view this administrative console page, click **Servers** > **Application servers** > *server_name* > **Ports** . Click on **View associated transports** for the port associated with the UDP transport channel whose settings you want to view.

Transport channel name

Specifies the name of the SIP container inbound transport channel.

The name field cannot contain the following characters: # \ / , : ; " * ? < > | = + & % '

This name must be unique across all channels in a WebSphere Application Server environment. For example, a SIP container transport channel and a TCP transport channel cannot have the same name if they reside within the same system.

Default	UDP_(n) where (n) represents the number of instances of this channel in the system

Descrimination weight

Specifies the priority this channel has in relation to the other channels in this transport chain. This property is only used when port sharing is enabled and the channel chain includes multiple channels to which it might forward data. The channel in the chain with the lowest discrimination weight is the first one given the opportunity to look at incoming data and determine whether or not it owns that data.

Data type	Positive integer
Default	10

User Datagram Protocol (UDP) Inbound channel settings

Use this page to configure the UDP Inbound channel settings.

To view this administrative console page, click **Servers** > **Application servers** > *server_name* > **Ports** . Click on **View associated transports** for the port associated with the UDP transport channel whose settings you want to view.

Transport channel name

Specifies the name of the UDP inbound transport channel.

The name field cannot contain the following characters: **# \ / , : ; ″ * ? < > | = + & % ’**

This name must be unique across all channels in a WebSphere Application Server environment. For example, a UDP transport channel and a TCP transport channel cannot have the same name if they reside within the same system.

Default
UDP_(n) where (n) represents the number of instances of this channel in the system

Address exclude list

Specifies the IP addresses that are not allowed to make inbound connections. Use a comma to separate the IPv4 and/or IPv6 addresses to which you want to deny access on inbound UDP connection requests.

The address include list and host name include list are processed before the address exclude list and the host name exclude list. If all four lists are defined:

- An address that is defined on either inclusion list will be allowed access provided it is not included on either of the exclusion lists.
- If an address is included in both an inclusion list and in an exclusion list, it is not allowed access.

Data type
String

Range
Valid IPv4 and IPv6 addresses with a wildcard character (*), an asterisk. All four elements of an IPv4 address must be represented by a number or a wildcard character. All eight numeric values of an IPv6 address must be represented by a number or the wildcard character (*).

Example
The following examples are valid IPv4 addresses that can be included in an Address exclude list:

```
*.1.255.0
254.*.*.9
1.*.*.*
```

All eight numeric values of an IPv6 address must be represented by a number or the wildcard character (*), an asterisk. No shortened version of the IPv6 address should be used. Even though a shortened version is processed with no error given, it does not function correctly in this list. Each numeric entry should be a 1- 4 digit hexadecimal number. The following examples are valid IPv6 addresses that can be included in an Address exclude list:

```
0:*:*:0:007F:0:0001:0001
F:FF:FFF:FFFF:1:01:001:0001
1234:*:4321:*:9F9f:*:*:0000
```

Address include list

Specifies the IP addresses that are allowed to make inbound connections. Use a comma to separate the IPv4 and/or IPv6 addresses to which you want to allow access on inbound UDP connection requests.

Data type	String
Range	Valid IPv4 and IPv6 addresses with a wildcard character (*), an asterisk. All four elements of an IPv4 address must be represented by a number or a wildcard character (*). All eight numeric values of an IPv6 address must be represented by a number or the wildcard character (*).

Web container inbound transport channel settings

Use this page to view and configure a Web container inbound channel transport. This type of channel transport handles inbound Web container requests from a remote client.

To view this administrative console page, click **Servers** > **Application servers** > *server_instance* > **Container Settings** > **Web Container Settings**> **Web container transport chains** > *transport_chain* > **Web container inbound channel (***transport_channel_name***)** .

Transport Channel Name

Specifies the name of the Web container inbound transport channel.

The name field cannot contain the following characters: # \ / , : ; ″ * ? < > | = + & % ’

This name must be unique across all channels in a WebSphere Application Server environment. For example, a Web container inbound transport channel and a TCP transport channel cannot have the same name if they reside within the same system.

Data type	String

Discrimination weight

Specifies the priority this channel has in relation to the other channels in this transport chain. This property is only used when port sharing is enabled and the channel chain includes multiple channels to which it might forward data. The channel in the chain with the lowest discrimination weight is the first one given the opportunity to look at incoming data and determine whether or not it owns that data.

Data type	Positive integer
Default	0

Write buffer size

Specifies the amount of content in bytes to buffer unless the servlet explicitly calls flush/close on the response/writer output stream.

Data type	bytes
Default	9192 bytes

Developing custom services

A custom service provides the ability to plug into a WebSphere Application Server application server to define a hook point that runs when the server starts and shuts down. An application server configuration provides settings that control how an application server provides services for running applications and their components.

A developer implements a custom service containing a class that implements a particular interface. The administrator configures the custom service in the administrative console, identifying the class created by the developer. When an application server starts, any custom services defined for the application server are loaded and the server runtime calls their initialize methods.

The following restrictions apply to the WebSphere Application Server custom services implementation:

- The init and shutdown methods must return control to the runtime.
- No work is dispatched into the server instance until all custom service initialize methods return.
- The init and shutdown methods are called only once on each service, and once for each operating system process that makes up the server instance. File I/O is supported.
- Initialization of process level static data, without leaving the process, is supported.
- Only JDBC RMLT (resource manager local transaction) operations are supported. Every unit of work (UOW) must be completed before the methods return.
- Creation of threads is not supported.
- Creation of sockets and I/O, other than file I/O, is not supported. Running standard J2EE code, such as client code, servlets, and enterprise beans, is not supported.
- The Java Transaction API (JTA) interface is not available. This feature is available in J2EE server processes and distributed generic server processes only.
- While the runtime makes an effort to call shutdown, there is no guarantee that shutdown will be called prior to process termination.

These restrictions apply to the shutdown and init methods equally. Some JNDI operations are available.

1. Develop a custom service class that implements the com.ibm.websphere.runtime.CustomService interface. The properties passed by the application server runtime to the initialize method can include one for an external file containing configuration information for the service. You can use the externalConfigURLKey property to retrieve this information. In addition, these properties can contain any name-value pairs that are stored for the service, along with the other system administration configuration data for the service. The properties are passed to the initialize method of the service as a Properties object.

   There is a shutdown method for the interface as well. Both methods of the interface declare that they may create an exception, although no specific exception subclass is defined. If an exception is created, the runtime logs it, disables the custom service, and proceeds with starting the server.
2. Configure the custom service.

   In the administrative console, click **Servers > Application Servers**, and then under Server Infrastructure, click **Custom Services > New**. Then, on the settings page for a custom service instance, create a custom service configuration for an existing application server or node agent, supplying the name of the class implemented. If your custom service class requires a configuration file, specify the fully-qualified path name to that configuration file in the **externalConfigURL** field. This file name is passed into your custom service class.

   To invoke a native library from the custom service, provide the path name in the **Classpath** field in addition to the path names that are used to locate the classes and JAR files for the custom service. Doing this adds the path name to the WebSphere Application Server extension classloader, which allows the custom service to locate and correctly load the native library.
3. Stop the application server and restart it.

   If you are developing a custom service for an application server, stop the application server and then restart the server.
4. Ensure the initialize and shutdown methods of the custom service perform as intended.

As mentioned above, your custom services class must implement the CustomService interface. In addition, your class must implement the initialize and shutdown methods. Suppose the name of the class that implements your custom service is *ServerInit*, your code would declare this class as shown below. The code below assumes that your custom services class needs a configuration file. It shows how to process the input parameter in order to get the configuration file. If your class does not require a configuration file, the code that processes configProperties is not needed.

```
public class ServerInit implements CustomService
{
/**
* The initialize method is called by the application server run-time when the
* server starts. The Properties object passed to this method must contain all
* configuration information necessary for this service to initialize properly.
*
* @param configProperties java.util.Properties
*/
    static final java.lang.String externalConfigURLKey =
       "com.ibm.websphere.runtime.CustomService.externalConfigURLKey";

    static String ConfigFileName="";

    public void initialize(java.util.Properties configProperties) throws Exception
    {
        if (configProperties.getProperty(externalConfigURLKey) != null)
        {
           ConfigFileName = configProperties.getProperty(externalConfigURLKey);
        }

       // Implement rest of initialize method
    }
/**
* The shutdown method is called by the application server run-time when the
* server begins its shutdown processing.
*
* @param configProperties java.util.Properties
*/
    public void shutdown() throws Exception
    {
        // Implement shutdown method
    }
```

Custom service collection

Use this page to view a list of services available to the application server and to see whether the services are enabled. A custom service provides the ability to plug into a WebSphere application server and define code that runs when the server starts or shuts down.

To view this administrative console page, click **Servers > Application servers >** *server_name*. Then, under Server Infrastructure, click **Administration > Custom Services**.

External Configuration URL

Specifies the URL for a custom service configuration file.

If your custom services class requires a configuration file, the value provides a fully-qualified path name to that configuration file. This file name is passed into your custom service class.

Classname

Specifies the class name of the service implementation. This class must implement the Custom Service interface.

Display Name

Specifies the name of the service.

Enable service at server startup

Specifies whether the server attempts to start and initialize the service when its containing process (the server) starts. By default, the service is not enabled when its containing process starts.

Custom service settings

Use this page to configure a service that runs in an application server.

To view this administrative console page, click **Servers > Application servers >** *server_name*. Then, under Server Infrastructure, click **Administration > Custom services >***custom_service_name*.

Enable service at server startup:

Specifies whether the server attempts to start and initialize the service when its containing process (the server) starts. By default, the service is not enabled when its containing process starts.

Data type	Boolean
Default	false

External Configuration URL:

Specifies the URL for a custom service configuration file.

If your custom services class requires a configuration file, specify the fully-qualified path name to that configuration file for the value. This file name is passed into your custom service class.

Data type	String
Units	URL

Classname:

Specifies the class name of the service implementation. This class must implement the Custom Service interface.

Data type	String
Units	Java class name

Display Name:

Specifies the name of the service.

Data type	String

Description:

Describes the custom service.

Data type	String

Classpath:

Specifies the class path used to locate the classes and JAR files for this service.

Data type	String
Units	Class path

Defining application server processes

To enhance the operation of an application server, you can define command-line information for starting or initializing an application server process. Such settings define runtime properties such as the program to run, arguments to run the program, and the working directory.

A process definition can include characteristics such as Java virtual machine (JVM) settings, standard in, error and output paths, and the user ID and password under which a server runs.

1. In the administrative console, click **Servers > Application Servers** click on an application server name, and then click **Java and Process Management > Process Definition**.

   You can also define application server processes using the wsadmin tool. For more information, see the *Using the administrative clients* PDF.
2. On the settings page for a process definition, specify the name of the executable to run, any arguments to pass when the process starts running, and the working directory in which the process will run. Then click **OK**.
3. Specify process execution statementsfor starting or initializing a UNIX or i5/OS process.
4. Specify monitoring policies to track the performance of a process.
5. Specify process logs to which standard out and standard error streams write. Complete this step if you do not want to use the default file names.
6. Specify name-value pairs for properties needed by the process definition.

   Important: Each custom property name must be unique. If the same name is used for multiple properties, the process uses the value specified for the first property that has that name.
7. Stop the application server and then restart the server.
8. Check the application server to ensure that the process definition runs and operates as intended.

Process definition settings

Use this page to configure a process definition. A process definition includes the command line information necessary to start or initialize a process.

For the base WebSphere Application Server and the WebSphere Application Server - Express products, only the command-line information for starting or initializing a process applies.

To view this administrative console page, click **Servers > Application Servers >** *server_name*. Then under Server Infrastructure click **Java™ Process Management > Process Definition**.

Executable Name

This command line information specifies the executable name that is invoked to start the process.

Data type	String

Executable Arguments

This command line information specifies the arguments that are passed *arg1 arg2 arg3*.

Data type	String
Units	Java command-line arguments

Start Command (startCommand)

This command line information specifies the platform-specific command to launch the server process.

Start Command Args (startCommandArgs)

This command line information specifies any additional arguments required by the start command.

Stop Command (stopCommand)

This command line information specifies the platform-specific command to stop the server process

Specify two commands in the field, one for the Stop command and one for the Immediate Stop (CANCEL) command.

Data type	String
Format	STOP *server_short_name*;CANCEL *server_short_name*

Stop Command Args (stopCommandArgs)

This command line information specifies any additional arguments required by the stop command.

Specify arguments for the Stop command and the Immediate Stop (CANCEL) command.

Data type	String
Format	*stop command arg string*;*immediate stop command arg string*
Example	;ARMRESTART In this example, Stop has no arguments. Immediate Stop has the argument ARMRESTART. A semicolon precedes ARMRESTART.

Terminate Command (terminateCommand)

This command line information specifies the platform-specific command to terminate the server process.

Data type	String
Format	FORCE *server_short_name*

Terminate Command Args (terminateCommandArgs)

This command line information specifies any additional arguments required by the terminate command.

The default is an empty string.

Data type	String
Format	*terminate command arg string*

Working directory

Specifies the file system directory that the process uses as its current working directory. The process uses this directory to determine the locations of input and output files with relative path names.

Data type	String

Executable target type

Select whether the executable target is a Java class or an executable JAR file.

Executable target

Specifies the name of the executable target. If the target type is a Java class name, this field contains the main() method. If the target type is an executable JAR file, this field contains the name of that JAR file.

Data type	String

Process execution settings

Use this page to view or change the process execution settings for a server process that applies to either an application server, a node agent or a deployment manager.

To view this administrative console page for an application server, click **Servers > Application Servers >** *server_name*. Then, under Server Infrastructure, click **Java and Process Management > Process Execution**.

To view this administrative console page for a node agent, click **System Administration > Node agents >***node_agent_name*. Then, under Server Infrastructure , click **Java and Process Management > Process Definition > Process Execution**.

To view this administrative console page for a deployment manager, click **System Administration > Deployment manager**. Then, under Server Infrastructure, click **Java and Process Management > Process Definition > Process Execution**.

Process Priority:

Specifies the operating system priority for the process. The administrative process that launches the server must have root operating system authority in order to honor this setting.

Data type	Integer
Default	20 for WebSphere Application Server on all operating systems.

UMASK:

Specifies the user mask under which the process runs (the file-mode permission mask).

Data type	Integer

Run As User:

Specifies the user that the process runs as.

Data type	String

Run As Group:

Specifies the group that the process is a member of and runs as.

On i5/OS, the Run As Group setting is ignored.

Data type	String

Run In Process Group:

Specifies a specific process group for the process. A process group is a mechanism that the operating system uses to logically associate multiple processes and operate on them as a single unit. Usually, the operating system uses this mechanism for signal distribution.

Specific operating systems might allow other operations to be performed on a process group. Refer to your operating system documentation for more information on the operations that can be performed on a process group.

Windows On i5/OS, and Windows operation systems, the Run In Process Group setting is ignored.

Data type	Integer
Default	0, which indicates that the process is not assigned to a specific process group.

Process logs settings

Use this page to view or change settings for specifying the files to which standard out and standard error streams write.

To view this administrative console page, in the administrative console:

For an application server on an i5/OS or distributed platform, click **Servers > Application Servers >** *server_name*, and then, under Server Infrastructure, click **Java and Process Management > Process Definition > Process Logs**.
For a deployment manager on an i5/OS or distributed platform, click **System Administration** > **Deployment Manager**, and then under Server Infrastructure, click **Java and Process Management > Process Definition > Process Logs**.

Stdout File Name:

Specifies the file to which the standard output stream is directed. The file name can include a symbolic path name defined in the variable entries.

Use the field on the configuration tab to specify the file name. Use the field on the Runtime tab to select a file for viewing. View the file by clicking **View**.

Direct server output to the administrative console or to the process that launched the server, by either deleting the file name or specifying `console` on the configuration tab.

Data type	String
Units	File path name

Stderr File Name:

Specifies the file to which the standard error stream is directed. The file name can include a symbolic path name defined in the variable entries.

Use the field on the configuration tab to specify the file name. Use the field on the runtime tab to select a file for viewing. View the file by clicking **View**.

Data type	String
Units	File path name

Monitoring policy settings

Use this page to view or change settings that control how the node agent monitors and restarts a process.

To view this administrative console page, click **Servers > Application Servers >** *server_name*. Then, under Server Infrastructure, click **Java and Process Management > Monitoring Policy**.

Maximum Startup Attempts:

Specifies the maximum number of times to attempt to start the application server before giving up.

Data type	Integer

Ping Interval:

Specifies the frequency of communication attempts between the parent process, such as the node agent, and the process it has spawned, such as an application server. Adjust this value based on your requirements for restarting failed servers. Decreasing the value detects failures sooner; increasing the value reduces the frequency of pings, reducing system overhead.

Data type	Integer
Range	Set the value greater than or equal to 0 (zero) and less than 2147483647. If you specify a value greater than 2147483647, the application server acts as though you set the value to 0.

Ping Timeout:

When a parent process is spawning a child process, such as when a process manager spawns a server, the parent process pings the child process to see whether the child was spawned successfully. This value specifies the number of seconds that the parent process should wait (after pinging the child process) before assuming that the child process failed.

Data type	Integer
Units	Seconds
Range	Set the value greater than or equal to 0 (zero) and less than 2147483647. If you specify a value greater than 2147483647, the application server acts as though you set the value to 0.

Automatic Restart:

Specifies whether the process should restart automatically if it fails. On distributed systems, the default is to restart the process automatically. On a z/OS system, the default is to not start the process automatically.

If you change the value specified for this field, you must restart the application server and the node agent before the new setting takes effect.

This setting does not affect what you specified for the Node Restart State setting. The two settings are mutually exclusive.

Data type	Boolean
Default	true (distributed) / false (z/OS)

Node Restart State:

Specifies the desired behavior of the servers after the node completely shuts down and restarts.

If a server is already running when the node agent stops, that server is still running after the node agent restarts. If a server is stopped when the node agent restarts, whether the node agent starts the server depends on the setting for this property:

- If this property is set to STOPPED, node agent does not start the server.
- If this property is set to RUNNING, the node agent always starts the server.

- If this property is set to PREVIOUS, the node agent starts the server only if the server was running when the node agent stopped.

This setting does not affect what you specified for the Automatic Restart setting. The two settings are mutually exclusive.

Data type	String
Default	STOPPED
Range	Valid values are STOPPED, RUNNING, or PREVIOUS. If you want the process to return to its current state after the node restarts, use PREVIOUS.

Automatically restarting server processes

There are several server processes related to WebSphere Application Server products that the operating system can monitor and automatically restart when the server processes stop abnormally. This task describes how to set up these *monitored* processes.

To set up this function on a Linux or UNIX-based operating system, you must have root authority to edit the inittab file.

To set up this function on a Microsoft Windows operating system, you must belong to the Administrator group and have the following advanced user rights:

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

The Installation wizard grants you the user rights if your user ID is part of the administrator group.

If you are running on a Microsoft Windows Operating System, the Installation wizard displays a message that states that although the advanced user rights are now effective, they do not display as effective until the next time you log on to the Windows machine.

You can also add the advanced user rights manually if you are performing a silent installation on a Windows operating system. For example, to grant the user rights to your administrator group user ID on a Windows operating system, perform the following procedure:

1. Click **Administrative Tools** in the Control Panel.
2. Click **Local Security Policy**.
3. Click **Local Policies**.
4. Click **User Rights Assignments**.
5. Right click **Act as part of the operating system**.
6. Click **Security**.
7. Click **Add**.
8. Click your user ID.
9. Click **Add**.
10. Click **OK**.
11. Click **OK**.
12. Right click **Log on as a service**.
13. Click **Security**.
14. Click **Add**.
15. Click **OK**.
16. Click **OK**.

17. Reboot your machine to make the settings effective.

Consult your Windows help system for more information.

You can use this function to automatically restart Express servers. You can restart the ***server1*** process, for example.

On a Linux or UNIX-based operating system, you must manually create a shell script that automatically starts any of the processes previously mentioned. Each UNIX shell script controls a single process, such as a stand-alone WebSphere Application Server instance. Multiple stand-alone Application Server processes require multiple UNIX scripts, which you can define.

If you do not install the WebSphere Application Server base product as a Windows service during installation, you can use the **WASService** command in the *app_server_root*/bin directory to do so at a later time. You can use this command to add any WebSphere Application Server process as a Windows service. The operating system can then monitor each server process and restart the process if it stops.

1. On a Windows operating system, **Use the installation wizard** to set up a Windows service to automatically monitor and restart processes related to the WebSphere Application Server product.
   - Perform the following procedure from the installation wizard to select services that the installation wizard can set up:
     a. Click **Run WebSphere Application Server as a service**.

        If you select this option, the installation wizard creates the following service during the installation:

        IBMWAS6Service - *node_name*

        The **IBMWAS6Service -** *node_name* service controls the *node_name* process.

        After you complete and verify the installation, use the Windows Services panel to change the **IBMWAS6Service -** *node_name* service to an automatic startup type.
        1) Right click **IBMWAS6Service -** *node_name* and click **Properties**.
        2) Click **Automatic** from the **Startup type** list box and click **OK**.

     b. Click **Run IBM HTTP Server as a service**.

        Select this option on the machine where you are installing the IBM HTTP Server.

        If you select this option, the installation wizard creates the following services during the installation:
        – **IBM HTTP Server 2.0.*x***
        – **IBM HTTP Administration 2.0.*x***

        The installation wizard defines the startup type of these services as **automatic**. It is not necessary for you to change the type from manual to automatic.

     c. Enter your user ID and password and click **Next**.

     In a coexistence environment, you can change the default service names to make them unique. In a same version coexistence scenario for IBM HTTP Server 2.0.x on a Windows platform, you cannot use the default service names created by the installer because they are common.

     To work around this problem:

     a. Install the first copy of IBM HTTP Server, either by itself or with WebSphere Application Server and select to install the services.

     b. Customize the service names for the first install by running the following commands from the first install location:

```
apache -k install -n "IHS 2.0(1)"
apache -k install -f conf\admin.conf -n "IHS 2.0 Administration (1)"
```

     c. Edit the AdminAlias directive in the *installLocation 1*\conf\admin.conf file to point to the new service name, such as **IHS 2.0(1)**.

     d. Remove the default service names installed by the first install by running the following commands:

```
apache -k uninstall -n "IBM HTTP Server 2.0"
apache -k uninstall -n "IBM HTTP Administration 2.0"
```

e. Install the second copy of IBM HTTP Server, either by itself or with WebSphere Application Server. The default service names correspond to the second install.

Note: Customized service names must be unique on your system.

2. On a Linux or UNIX operating system, after you install the WebSphere Application Server product, set up a Linux and UNIX-based shell script to automatically monitor and restart the node agent process or any other related server process.
   a. Locate the rc.was example shell script, which is in the *app_server_root*/bin directory.
   b. Create a new shell script for each process that the operating system is to monitor and restart.
   c. Edit each shell script according to comments in its header, which provide instructions for identifying a WebSphere Application Server process.
   d. Edit the inittab table of the operating system, to add an entry for each shell script you have created.

      Comments in the header of the rc.was file show a sample inittab entry line for adding the script. This inittab entry causes the Linux and UNIX-based system to call each shell script whenever the system initializes. As it runs, each shell script monitors and starts the server process you specified.

   Each shell script monitors and restarts an Express server process.
3. On a Windows operating system, after installing the WebSphere Application Server product, you can use the WASService.exe command in the *app_server_root*\bin directory to manually define a Windows service for another installation instance or for another configuration instance of the server1 process.

On a Windows operating system, you can

- Use the **net start** and **net stop** commands to control the IBM HTTP Server services on a Windows system. For more information about these commands, see the Windows help file. Access these commands from the Start menu, clicking **Start > Programs > IBM HTTP Server**.
- Use the **Start the Server** and **Stop the Server** commands to control the IBM WebSphere Application Server process. Access these commands from the Start menu, clicking **Start > Programs > IBM WebSphere > Application Server V6**.
-

Processes started by a **startServer**command are not running as monitored processes, regardless of how they are configured.

For example, you can configure a server1 process as a monitored process. However, if you start the server1 process using the **startServer** command, the operating system does not monitor or restart the server1 process because the operating system did not originally start the process as a monitored process.

After the process is set up, the operating system can monitor each server process and restart the process if it stops.

Return to Defining application server processes to continue.

WASService command

The **WASService** command line tool lets you create a Windows service for any WebSphere Application Server Java process.

You can create services for WebSphere Application Server Java processes. Potential services include the following server processes:

- The default server1 process on an application server node
- Application server processes that you create on an application server node

When the installation wizard creates a service, the uninstaller program can remove the service. If you use the **WASService** command to create a service yourself, it is your responsibility to remove the service when it is no longer valid. The uninstaller program does not remove services that you create with the **WASService** command.

Location of the command file: Windows The WASService.exe command file is located in the *app_server_root*\bin directory.

Linux The wasservice.sh command file is located in the *app_server_root*\bin directory.

Command syntax:

Command syntax for starting an existing service

Windows The command syntax is as follows:

```
WASService.exe -start "service_name" [optional startServer.bat parameters]
```

Linux The command syntax is as follows:

```
wasservice.sh -start "service_name" [optional startServer.bat parameters]
```

Command syntax for creating a service or updating an existing service

Windows The command syntax is as follows:

```
WASService.exe -add "service_name"
   -serverName server
   -profilePath server_profile_directory
  [-wasHome  app_server_root]
  [-configRoot configuration_repository_directory]
  [-startArgs additional_start_arguments]
  [-stopArgs additional_stop_arguments]
  [-userid user_id -password password]
  [-logFile service_log_file]
  [-logRoot server_log_directory]
  [-restart true | -restart false]
  [-startType automatic | manual | disabled]
```

Linux The command syntax is as follows:

```
wasservice.sh -add "service_name"
   -serverName server
   -profilePath server_profile_directory

  [-wasHome  app_server_root]
  [-startArgs additional_start_arguments]
  [-stopArgs additional_stop_arguments]
  [-userid user_id -password password]
```

Command syntax for deleting a service

The command syntax is as follows:

Windows

```
WASService.exe -remove "service_name"
```

Linux

```
wasservice.sh -remove "service_name"
```

Command syntax for stopping a running service

The command syntax is as follows:

Windows

```
WASService.exe -stop "service_name" [optional stopServer.bat parameters]
```

Linux

```
wasservice.sh -stop "service_name" [optional stopServer.bat parameters]
```

Command syntax for retrieving service status

The command syntax is as follows:

Windows

```
WASService.exe -status "service_name"
```

Linux

```
wasservice.sh -status "service_name"
```

Command syntax for encoding parameters

The command syntax is as follows:

Windows

```
WASService.exe -encodeParams "service_name"
```

Parameters: Supported arguments include:

-add *"service_name"*
: Creates a service named *service_name* or updates an existing service. The syntax is the same for both cases.

-configRoot *configuration_repository_directory*
: Optional parameter that identifies the configuration directory of the installation root directory of a WebSphere Application Server product.

-encodeParams *service_name*
: Optional parameter that forces the service to encode the -startArgs and -stopArgs so that the arguments cannot be determined by editing the registry. Use the parameter when creating a service with the -add parameter by adding -encodeParams to the command line with no arguments. Or encode the parameters of an existing service:

  Windows

  ```
  WASService -encodeParams service_name
  ```

-logFile *service_log_file*
: Optional parameter that identifies a log file that the WASService command uses to record its activity.

-logRoot *server_log_directory*
: Required parameter that identifies the server log directory for the profile. The WASService command looks for a file named *server_name*.pid to determine if the server is running.

-profilePath *server_profile_directory*
: Specifies the directory path of the profile that defines the server process.

-remove *service_name*
: Deletes the specified service.

-restart true | false
Restarts the existing service automatically if the service fails when set to true.

-serverName *Server_name*
Identifies the server that the service controls.

-start *″service_name″* **[optional startServer.bat parameters]**
Starts the existing service.

-startArgs *additional_start_arguments*
Optional parameter that identifies additional parameters.

-startType automatic | manual | disabled
Defines the startup type of the new service. An automatic startup type starts automatically when the system starts or when the service is called for the first time. You must start a manual service before the operating system can load it and make it available. You cannot start a disabled service before changing the startup type.

-status *service_name*
Returns the current status of the service, which includes whether the service is running or stopped.

-stop *service_name* **[optional stopServer.bat parameters]**
Stops the specified service.

-stopArgs *additional_stop_arguments*
Optional parameter that identifies additional parameters.

-userid *user_ID* **-password** *password*
Optional parameters that identify a privileged user ID and password that the Windows service will run as.

-wasHome *app_server_root*
Optional parameter that identifies the installation root directory of the WebSphere Application Server product.

Default names for services that are created by the wizard: The name of the service that is created by the Installation wizard is IBM WebSphere Application Server V6.x - DefaultNode.

Viewing the services panel: To view services, open the Control panel and click **Administrative Tools > Services**. Select a service to view information about it. Right click the service and click **Properties**. Four tabs provide information and functionality. For example, select the **Setup type** field on the **General** tab to change the setup type.

Examples:
Creating an Application Server service

This example creates a service called *IBM WebSphere Application Server V6.x - server2* that starts an Application Server process:

Windows

```
WASService -add server2
   -servername server2
   -profilePath "profile_root"
   -wasHome "app_server_root"
   -logfile "profile_root\logs\WS_startNode.log"
   -logRoot "profile_root\logs"
   -restart true
```

After entering the command, messages that are similar to those in the following example display in the command window:

Windows

```
Adding Service: server2
  Config Root: profile_root\config
  Server Name: server2
  Profile Path: profile_root
  Was Home: app_server_root
  Start Args:
  Restart: 1
IBM WebSphere Application Server V6.x - server2 service successfully added.
```

Updating an existing Application Server service

This example updates an existing service called IBM WebSphere Application Server V6.x - server2 with additional stop arguments, username and password. The user name and password are required by the stopServer command to stop the application server with security enabled.

Windows

```
WASService -add server2
  -servername server2
  -profilePath "profile_root"
   -stopArgs "-username user_name  -password password"
   -encodeParams
```

Starting and stopping a server process after creating a Windows service: If you issue the startServer server1 command or the stopServer server1 after creating a Windows service for server1, a message that is similar to the following example displays:

Windows

```
Because server1 is registered to run as a Windows Service, the
request to start this server will be completed by starting the
associated Windows Service.
```

Windows

```
Because nodeagent is registered to run as a Windows Service, the
request to start or stop this server will be completed by
starting or stopping the associated Windows Service. Examine
the log files to view messages related to this command.
```

Windows

```
Because dmgr is registered to run as a Windows Service, the
request to start or stop this server will be completed by
starting or stopping the associated Windows Service. Examine
the log files to view messages related to this command.
```

Stopping a server after enabling security

If you enable security while a service is running, you can stop the server from the command line by using the username and password parameters on the stopServer command.

Configuring the JVM

As part of configuring an application server, you might define settings that enhance the way your operating system uses of the Java virtual machine (JVM).

The Java virtual machine (JVM) is an interpretive computing engine responsible for running the byte codes in a compiled Java program. The JVM translates the Java byte codes into the native instructions of the host machine. The application server, being a Java process, requires a JVM in order to run, and to support the Java applications running on it. JVM settings are part of an application server configuration.

To view and change the JVM configuration for an application server's process, use the Java virtual machine page of the administrative console or use wsadmin to change the configuration through scripting.

1. In the administrative console, click **Servers > Application Servers >***server***> Java and Process Management > Process Definition > Java Virtual Machine**.
2. Specify values for the JVM settings as needed and click **OK**.
3. Click **Save** on the console task bar.
4. Restart the application server.

"Configuring application servers for UCS Transformation Format" on page 218 provides an example that involves specifying a value for the **Generic JVM Arguments** property on the Java virtual machine page to enable UTF-8 encoding on an application server. Enabling UTF-8 allows multiple language encoding support to be used in the administrative console.

"Configuring JVM sendRedirect calls to use context root" on page 294 provides an example that involves defining a property for the JVM.

Java virtual machine settings

Use this page to view and change the Java virtual machine (JVM) configuration settings of a process for an application server.

To view this administrative console page, connect to the administrative console and navigate to the Java virtual machine panel:

For the i5/OS and distributed platforms base WebSphere Application Server and the WebSphere Application Server - Express products:

Click **Servers > Application Servers >** *server1* **> Java and Process Management > Process definition > Java Virtual Machine**

Classpath

Specifies the standard class path in which the Java virtual machine code looks for classes.

Enter each classpath entry into a table row. You do not need to add the colon or semicolon at the end of each entry.

Data type	String
Units	Class path

Boot classpath

Specifies bootstrap classes and resources for JVM code. This option is only available for JVM instructions that support bootstrap classes and resources. You can separate multiple paths by a colon (:) or semi-colon (;), depending on operating system of the node.

Data type	String

Verbose class loading

Specifies whether to use verbose debug output for class loading. The default is not to enable verbose class loading.

Data type	Boolean

Default	false

Verbose garbage collection

Specifies whether to use verbose debug output for garbage collection. The default is not to enable verbose garbage collection.

Data type	Boolean
Default	false

Verbose JNI

Specifies whether to use verbose debug output for native method invocation. The default is not to enable verbose Java Native Interface (JNI) activity.

Data type	Boolean
Default	false

Initial heap size

Specifies the initial heap size available to the JVM code, in megabytes.

Increasing the minimum heap size can improve startup. The number of garbage collection occurrences are reduced and a 10% gain in performance is realized.

Increasing the size of the Java heap improves throughput until the heap no longer resides in physical memory, in general. After the heap begins swapping to disk, Java performance suffers drastically.

Data type	Integer
Default	For distributed platforms, the default is 50.

Maximum heap size

Specifies the maximum heap size available to the JVM code, in megabytes.

Increasing the heap size can improve startup. By increasing heap size, you can reduce the number of garbage collection occurrences with a 10% gain in performance.

Increasing the size of the Java heap usually improves throughput until the heap no longer resides in physical memory. When the heap size exceeds the physical memory, the heap begins swapping to disk which causes Java performance to drastically decrease. Therefore, it is important to set the maximum heap size to a value that allows the heap to be contained within physical memory.

To prevent paging, you should allow a minimum of 256MB of physical memory for each processor and 512 MB of physical memory for each application server. If possible, adjust the available memory when paging occurs if processor utilization is low because of this paging.

Data type	Integer
Default	For z/OS and distributed platforms, the default is 256. Keep the value low enough to avoid paging or swapping-out-memory-to-disk.

Run HProf

Specifies whether to use HProf profiler support. To use another profiler, specify the custom profiler settings using the HProf Arguments setting. The default is not to enable HProf profiler support.

If you set the Run HProf property to true, then you must specify command-line profiler arguments as values for the HProf Arguments property.

Data type	Boolean
Default	false

HProf arguments

Specifies command-line profiler arguments to pass to the JVM code that starts the application server process. You can specify arguments when HProf profiler support is enabled.

HProf arguments are only required if the Run HProf property is set to true.

Data type	String

Debug mode

Specifies whether to run the JVM in debug mode. The default is not to enable debug mode support.

If you set the Debug Mode property to true, then you must specify command-line debug arguments as values for the Debug Arguments property.

Data type	Boolean
Default	false

Debug arguments

Specifies command-line debug arguments to pass to the JVM code that starts the application server process. You can specify arguments when Debug Mode is enabled.

For base WebSphere Application Server and the WebSphere Application Server - Express configurations, Debug arguments are only required if the Debug Mode property is set to true. If you enable debugging on multiple application servers on the same server, make sure that the servers are using different `address` arguments, which define the port for debugging. For example, if you enable debugging on two servers and leave the default debug port for each server as `address=7777`, the servers could fail to start properly.

Data type	String
Units	Java command-line arguments

Generic JVM arguments

Specifies command line arguments to pass to the Java virtual machine code that starts the application server process.

The following are optional command line arguments that you can ente int the Generic JVM arguments field. If you enter more than one argument, enter a space between each argument.

Important: If the argument says it is for the IBM Developer Kit only, you cannot use the argument with another JVM, such as the Sun JDK or the HP JDK.

- **-Xquickstart**

  You can use **-Xquickstart** for initial compilation at a lower optimization level than in default mode. Later, depending on sampling results, you can recompile to the level of the initial compile in default mode. Use **-Xquickstart** for applications where early moderate speed is more important than long run throughput. In some debug scenarios, test harnesses and short-running tools, you can improve startup time between 15-20%.
- **-Xverify:none**

You can use **-Xverify:none** if you want to skip the class verification stage during class loading . Using **-Xverify:none** with the just in time (JIT) compiler enabled, improves startup time by 10-15%.

- **-Xnoclassgc**

  You can use **-Xnoclassgc** to disable class garbage collection. This action leads to more class reuse and slightly improved performance. The trade-off is that you won't be collecting the resources owned by these classes. You can monitor garbage collection using the `verbose:gc` configuration setting, which will output class garbage collection statistics. Examining these statistics will help you understand the trade-off between the reclaimed resources and the amount of garbage collection required to reclaim the resources. However, if the same set of classes are garbage collected repeatedly in your workload, you should disable garbage collection. Class garbage collection is enabled by default.
- **-Xgcthreads**

  You can use several garbage collection threads at one time, also known as *parallel garbage collection*. When entering this value in the Generic JVM arguments field, also enter the number of processors that your machine has, for example, **-Xgcthreads***n*, where *n* is the number of processors. On a node with *n* processors, the default number of threads is *n*. You should use parallel garbage collection if your machine has more than one processor. This argument is valid only for the IBM Developer Kit.
- **-Xnocompactgc**

  You can use **-Xnocompactgc** to disable heap compaction, which is the most expensive garbage collection operation. Avoid compaction in the IBM Developer Kit. If you disable heap compaction, you eliminate all associated overhead.
- **-Xinitsh**

  You can use **-Xinitsh** to set the initial heap size where class objects are stored. The method definitions and static fields are also stored with the class objects. Although the system heap size has no upper bound, set the initial size so that you do not incur the cost of expanding the system heap size, which involves calls to the operating system memory manager. You can compute a good initial system heap size by knowing the number of classes loaded in the WebSphere Application Server product, which is about 8,000 classes, and their average size. Having knowledge of the applications helps you include them in the calculation. You can use this argument only with the IBM Developer Kit.
- **-Xgpolicy**

  You can use **-Xgpolicy** to set the garbage collection policy. If the garbage collection policy (gcpolicy) is set to `optavgpause`, concurrent marking is used to track application threads starting from the stack before the heap becomes full. The garbage collector pauses become uniform and long pauses are not apparent. The trade-off is reduced throughput because threads might have to do extra work. The default, recommended value is `optthruput`. Enter the value as `-Xgcpolicy:[optthruput|optavgpause]`. You can use this argument only with the IBM Developer Kit.
- **-XX**

  The Sun-based Java 2 Standard Edition (J2SE) 5 has generation garbage collection, which allows separate memory pools to contain objects with different ages. The garbage collection cycle collects the objects independently from one another depending on age. With additional parameters, you can set the size of the memory pools individually. To achieve better performance, set the size of the pool containing short lived objects so that objects in the pool do not live through more then one garbage collection cycle. The size of new generation pool is determined by the `NewSize` and `MaxNewSize` parameters.

  Objects that survive the first garbage collection cycle are transferred to another pool. The size of the survivor pool is determined by parameter `SurvivorRatio`. If garbage collection becomes a bottleneck, you can try customizing the generation pool settings. To monitor garbage collection statistics, use the object statistics in Tivoli Performance Viewer or the verbose:gc configuration setting. Enter the following values:

  ```
  -XX:NewSize (lower bound)
  -XX:MaxNewSize (upper bound)
   -XX:SurvivorRatio=NewRatioSize
  ```

  The default values are:`NewSize=2m MaxNewSize=32m SurvivorRatio=2`. However, if you have a JVM with more than 1 GB heap size, you should use the values: `-XX:newSize=640m -XX:MaxNewSize=640m -XX:SurvivorRatio=16`, or set 50 to 60% of total heap size to a new generation pool.
- **-Xminf**

You can use **-Xminf** to specify the minimum free heap size percentage. The heap grows if the free space is below the specified amount. In reset enabled mode, this option specifies the minimum percentage of free space for the middleware and transient heaps. This is a floating point number, 0 through 1. The default is .3 (30%).

- **-server | -client**

  Java HotSpot Technology in the Sun-based Java 2 Standard Edition (J2SE) 5 uses an adaptive JVM containing algorithms for optimizing byte code execution over time. The JVM runs in two modes, **-server** and **-client**. If you use the default **-client** mode, there will be a faster startup time and a smaller memory footprint, but lower extended performance. You can enhance performance by using **-server** mode if a sufficient amount of time is allowed for the HotSpot JVM to warm up by performing continuous execution of byte code. In most cases, use **-server** mode, which produces more efficient run-time execution over extended periods. You can monitor the process size and the server startup time to check the difference between **-client** and **-server**.
- **-Dcom.ibm.CORBA.RequestTimeout=*timeout_interval***

  You can use **-Dcom.ibm.CORBA.RequestTimeout=*timeout_interval*** to specify the timeout period for responding to requests sent from the client. This argument uses the -D option. *timeout_interval* is the timeout period in seconds. If your network experiences extreme latency, specify a large value to prevent timeouts. If you specify a value that is too small, an application server that participates in workload management can time out before it receives a response.

  Be careful specifying this property; it has no recommended value. Set it only if your application is experiencing problems with timeouts.
- **-Dcom.ibm.websphere.wlm.unusable.interval=*interval***

  You can use the **-Dcom.ibm.websphere.wlm.unusable.interval=*interval*** argument to change the value for the com.ibm.websphere.wlm.unusable.interval property if the workload management state of the client is refreshing too soon or too late. This property specifies the time interval that the workload management client run time waits after it marks a server as unavailable before it attempts to contact the server again. This argument uses the -D option. *interval* is the time in seconds between attempts. The default value is 300 seconds. If the property is set to a large value, the server is marked as unavailable for a long period of time. This prevents the workload management refresh protocol from refreshing the workload management state of the client until after the time period has ended.
- **-Xshareclasses:none**

  You can use the **-Xshareclasses:none** argument to disable the share classes option for a process. The share classes option, which is available with Java 2 Standard Edition (J2SE) 5, lets you share classes in a cache. Sharing classes in a cache can improve startup time and reduce memory footprint. Processes, such as application servers, node agents, and deployment managers, can use the share classes option.

  If you use this option, you should clear the cache when the process is not in use. To clear the cache, either call the app_server_root/bin/clearClassCache.bat/sh utility or stop the process and then restart the process.

  ▶ Solaris ▶ HP-UX

  Important: ▶ Solaris ▶ HP-UX The IBM J2SE 5 is currently not used on Solaris, HP, and i5/OS.

Data type	String
Units	Java command line arguments

Executable JAR file name

Specifies a full path name for an executable JAR file that the JVM code uses.

Data type	String
Units	Path name

Disable JIT

Specifies whether to disable the just in time (JIT) compiler option of the JVM code.

If you disable the JIT compiler, throughput decreases noticeably. Therefore, for performance reasons, keep JIT enabled.

Data type	Boolean
Default	false (JIT enabled)
Recommended	JIT enabled

Operating system name

Specifies JVM settings for a given operating system.

For the base WebSphere Application Server and the WebSphere Application Server - Express products, when the process starts, the process uses the JVM settings specified for the server as the JVM settings for the operating system.

Data type	String

Configuring JVM sendRedirect calls to use context root

If the com.ibm.websphere.sendredirect.compatibility property is not set and your application servlet code has statements such as *sendRedirect("/home.html")*, your Web browser might display messages such as *Error 404: No target servlet configured for uri: /home.html*.

transition: The com.ibm.websphere.sendredirect.compatibility property is deprecated. You should modify your applications to redirect non-relative URLs (those starting with a "/") relative to the servlet container (*web_server_root*) instead of relative to the Web application context root.

To instruct the server not to use the Web server's document root and to use instead the Web application's context root for sendRedirect() calls, configure the JVM by setting the com.ibm.websphere.sendredirect.compatibility property to a `true` or `false` value.

1. Access the settings page for a property of the JVM.
   a. Click **Servers > Application Servers** in the console navigation tree.
   b. On the Application Server page, click on the name of the server whose JVM settings you want to configure.
   c. On the settings page for the selected application server, under Server Infrastructure, click **Java and Process Management > Process Definition**.
   d. On the Process Definition page, click **Java Virtual Machine**.
   e. On the Java Virtual Machine page, click **Custom Properties**.
   f. On the Custom Properties page, click **New**.
2. On the settings page for a property, specify a name of `com.ibm.websphere.sendredirect.compatibility` and either `true` or `false` for the value, then click **OK**.
3. Click **Save** on the console task bar.
4. Stop the application server and then restart the application server.

Setting custom JVM properties

You can use the administrative console to change the values of JVM custom properties.

To set custom properties, connect to the administrative console and navigate to the Java virtual machine custom properties panel.

Application server	**Servers** > **Application Servers** > *server1* > **Java and Process Management** > **Process Definition** > **Java Virtual Machine** > **Custom Properties**
Deployment manager	**System Administration** > **Deployment Manager** > **Java and Process Management** > **Process definition** > > **Java Virtual Machine** > **Custom Properties**
Node agent	**System Administration** >**Node Agent** > *nodeagent* > **Java and Process Management** > **Process definition** > **Java Virtual Machine** > **Custom Properties**

If the custom property is not present in the list of already defined custom properties, create a new property, and enter the property name in the Name field and a valid value in the Value field. Restart the server to complete your changes.

com.ibm.websphere.network.useMultiHome

Set this property in a multihomed environment where WebSphere Application Server is restricted to listen only on a specific IP address for Discovery and SOAP messages. The settings for the **com.ibm.websphere.network.useMultiHome** property are as follows:

- Setting this property to `false` specifies that WebSphere Application Server will listen on all IP addresses on the host for Discovery and SOAP messages.
- Setting this property to `true` specifies that WebSphere Application Server will only listen on the configured host name for Discovery and SOAP messages. If you set this property to `true`, you should have a host name configured on WebSphere Application Server that resolves to a specific IP address.
- Setting this property to `null` specifies that WebSphere Application Server will only listen on the default IP address only.

If you cannot contact the server, check the setting for **com.ibm.websphere.network.useMultihome** to ensure it is correct. You can change the value through the administrative console. Modify the defaults by setting the value for the server, deployment manager, and node agent. In order for these changes to take place, you must restart the server.

com.ibm.websphere.deletejspclasses

Deletes JavaServer Pages classes for all applications after those applications have been deleted or updated. By default, the value of this property is `true`.

com.ibm.websphere.deletejspclasses.delete

Deletes JavaServer Pages classes for all applications after those applications have been deleted, but not after they have been updated. By default, the value of this property is `true`.

com.ibm.websphere.deletejspclasses.update

Deletes JavaServer Pages classes for all applications after those applications have been updated, but not after they have been deleted. By default, the value of this property is `true`.

invocationCacheSize

The invocationCacheSize custom property is used to control the size of the invocation cache. The invocation cache holds information for mapping request URLs to servlet resources. A cache of the requested size is created for each worker thread that is available to process a request. The default size of the invocation cache is 50. If more than 50 unique URLs are actively being used (each JavaServer Page is a unique URL), you should increase the size of the invocation cache.

A larger cache uses more of the Java heap, so you might also need to increase the maximum Java heap size. For example, if each cache entry requires 2KB, maximum thread size is set to 25, and the URL invocation cache size is 100; then 5MB of Java heap are required.

You can specify any number higher than 0 for the cache size. Setting the value to zero disables the invocation cache.

Preparing to host applications

Rather than use the default application server provided with the product, you can configure a new server and set of resources.

The default application server and a set of default resources are available to help you begin quickly. You can choose instead to configure a new server and set of resources. Here is what you need to do in order to set up a runtime environment to support applications.

1. Create an application server.
2. Create a virtual host.
3. Configure a Web container. See the *Administering applications and their environment* PDF for more information.
4. Configure an EJB container. See the *Administering applications and their environment* PDF for more information.
5. Create resources for data access. See the *Administering applications and their environment* PDF for more information.
6. Create a JDBC provider and data source. See the *Administering applications and their environment* PDF for more information.
7. Create a URL and URL provider. See the *Administering applications and their environment* PDF for more information.
8. Create a JavaMail session. See the *Administering applications and their environment* PDF for more information.
9. Create resources for session support. See the *Administering applications and their environment* PDF for more information.
10. Configure a Session Manager. See the *Administering applications and their environment* PDF for more information.

Java memory tuning tips

Enterprise applications written in the Java language involve complex object relationships and utilize large numbers of objects. Although, the Java language automatically manages memory associated with object life cycles, understanding the application usage patterns for objects is important.

In particular, verify the following:
- The application is not over utilizing objects
- The application is not leaking objects
- The Java heap parameters are set properly to handle a given object usage pattern

Understanding the effect of garbage collection is necessary to apply these management techniques.

IBM Support has documents that can save you time gathering information needed to resolve this problem. Before opening a PMR, see the IBM Support page.

The garbage collection bottleneck

Examining Java garbage collection gives insight to how the application is utilizing memory. Garbage collection is a Java strength. By taking the burden of memory management away from the application writer, Java applications are more robust than applications written in languages that do not provide garbage collection. This robustness applies as long as the application is not abusing objects. Garbage collection normally consumes from 5% to 20% of total execution time of a properly functioning application. If not managed, garbage collection is one of the biggest bottlenecks for an application.

Monitoring garbage collection

You can use garbage collection to evaluate application performance health. By monitoring garbage collection during the execution of a fixed workload, you gain insight as to whether the application is over-utilizing objects. Garbage collection can even detect the presence of memory leaks.

For more information about monitoring garbage collection, see:

- The description of the DMPJVM command in the i5/OS Information Center. This command dumps JVM information for a specific job.

You can monitor garbage collection statistics using object statistics in the Tivoli Performance Viewer, or using the **verbose:gc** JVM configuration setting. The **verbose:gc** format is not standardized between different JVMs or release levels.

For this type of investigation, set the minimum and maximum heap sizes to the same value. Choose a representative, repetitive workload that matches production usage as closely as possible, user errors included.

To ensure meaningful statistics, run the fixed workload until the application state is steady. It usually takes several minutes to reach a steady state.

Detecting over-utilization of objects

You can use the Tivoli Performance Viewer to check if the application is overusing objects, by observing the counters for the JVM runtime. You have to set the **-XrunpmiJvmpiProfiler** command line option, as well as the JVM module maximum level in order to enable the Java virtual machine profiler interface (JVMPI) counters.

The best result for the average time between garbage collections is at least 5-6 times the average duration of a single garbage collection. If you do not achieve this number, the application is spending more than 15% of its time in garbage collection.

If the information indicates a garbage collection bottleneck, there are two ways to clear the bottleneck. The most cost-effective way to optimize the application is to implement object caches and pools. Use a Java profiler to determine which objects to target. If you can not optimize the application, adding memory, processors and clones might help. Additional memory allows each clone to maintain a reasonable heap size. Additional processors allow the clones to run in parallel.

Detecting memory leaks

Memory leaks in the Java language are a dangerous contributor to garbage collection bottlenecks. Memory leaks are more damaging than memory overuse, because a memory leak ultimately leads to system instability. Over time, garbage collection occurs more frequently until the heap is exhausted and the Java code fails with a fatal out-of-memory exception. Memory leaks occur when an unused object has references that are never freed. Memory leaks most commonly occur in collection classes, such as Hashtable because the table always has a reference to the object, even after real references are deleted.

High workload often causes applications to crash immediately after deployment in the production environment. This is especially true for leaking applications where the high workload accelerates the magnification of the leakage and a memory allocation failure occurs.

Memory leak testing

The goal of memory leak testing is to magnify numbers. Memory leaks are measured in terms of the amount of bytes or kilobytes that cannot be garbage collected. The delicate task is to differentiate these amounts between expected sizes of useful and unusable memory. This task is achieved more easily if the numbers are magnified, resulting in larger gaps and easier identification of inconsistencies. The following list contains important conclusions about memory leaks:

- **Long-running test**

  Memory leak problems can manifest only after a period of time, therefore, memory leaks are found easily during long-running tests. Short running tests can lead to false alarms. It is sometimes difficult to know when a memory leak is occurring in the Java language, especially when memory usage has seemingly increased either abruptly or monotonically in a given period of time. The reason it is hard to detect a memory leak is that these kinds of increases can be valid or might be the intention of the developer. You can learn how to differentiate the delayed use of objects from completely unused objects by running applications for a longer period of time. Long-running application testing gives you higher confidence for whether the delayed use of objects is actually occurring.
- **Repetitive test**

  In many cases, memory leak problems occur by successive repetitions of the same test case. The goal of memory leak testing is to establish a big gap between unusable memory and used memory in terms of their relative sizes. By repeating the same scenario over and over again, the gap is multiplied in a very progressive way. This testing helps if the number of leaks caused by the execution of a test case is so minimal that it is hardly noticeable in one run.

  You can use repetitive tests at the system level or module level. The advantage with modular testing is better control. When a module is designed to keep the private module without creating external side effects such as memory usage, testing for memory leaks is easier. First, the memory usage before running the module is recorded. Then, a fixed set of test cases are run repeatedly. At the end of the test run, the current memory usage is recorded and checked for significant changes. Remember, garbage collection must be suggested when recording the actual memory usage by inserting System.gc() in the module where you want garbage collection to occur, or using a profiling tool, to force the event to occur.
- **Concurrency test**

  Some memory leak problems can occur only when there are several threads running in the application. Unfortunately, synchronization points are very susceptible to memory leaks because of the added complication in the program logic. Careless programming can lead to kept or unreleased references. The incident of memory leaks is often facilitated or accelerated by increased concurrency in the system. The most common way to increase concurrency is to increase the number of clients in the test driver.

  Consider the following points when choosing which test cases to use for memory leak testing:

  – A good test case exercises areas of the application where objects are created. Most of the time, knowledge of the application is required. A description of the scenario can suggest creation of data spaces, such as adding a new record, creating an HTTP session, performing a transaction and searching a record.

  – Look at areas where collections of objects are used. Typically, memory leaks are composed of objects within the same class. Also, collection classes such as Vector and Hashtable are common places where references to objects are implicitly stored by calling corresponding insertion methods. For example, the get method of a Hashtable object does not remove its reference to the retrieved object.

You can use the Tivoli Performance Viewer to help find memory leaks.

For the best results, repeat experiments with increasing duration, like 1000, 2000, and 4000 page requests. The Tivoli Performance Viewer graph of used memory should have a sawtooth shape. Each drop on the graph corresponds to a garbage collection. There is a memory leak if one of the following occurs:

- The amount of memory used immediately after each garbage collection increases significantly. The sawtooth pattern looks more like a staircase.
- The sawtooth pattern has an irregular shape.

Also, look at the difference between the number of objects allocated and the number of objects freed. If the gap between the two increases over time, there is a memory leak.

Heap consumption indicating a possible leak during a heavy workload (the application server is consistently near 100% CPU utilization), yet appearing to recover during a subsequent lighter or near-idle workload, is an indication of heap fragmentation. Heap fragmentation can occur when the JVM can free sufficient objects to satisfy memory allocation requests during garbage collection cycles, but the JVM does not have the time to compact small free memory areas in the heap to larger contiguous spaces.

Another form of heap fragmentation occurs when small objects (less than 512 bytes) are freed. The objects are freed, but the storage is not recovered, resulting in memory fragmentation until a heap compaction has been run.

Heap fragmentation can be reduced by forcing compactions to occur, but there is a performance penalty for doing this. Use the Java **-X** command to see the list of memory options.

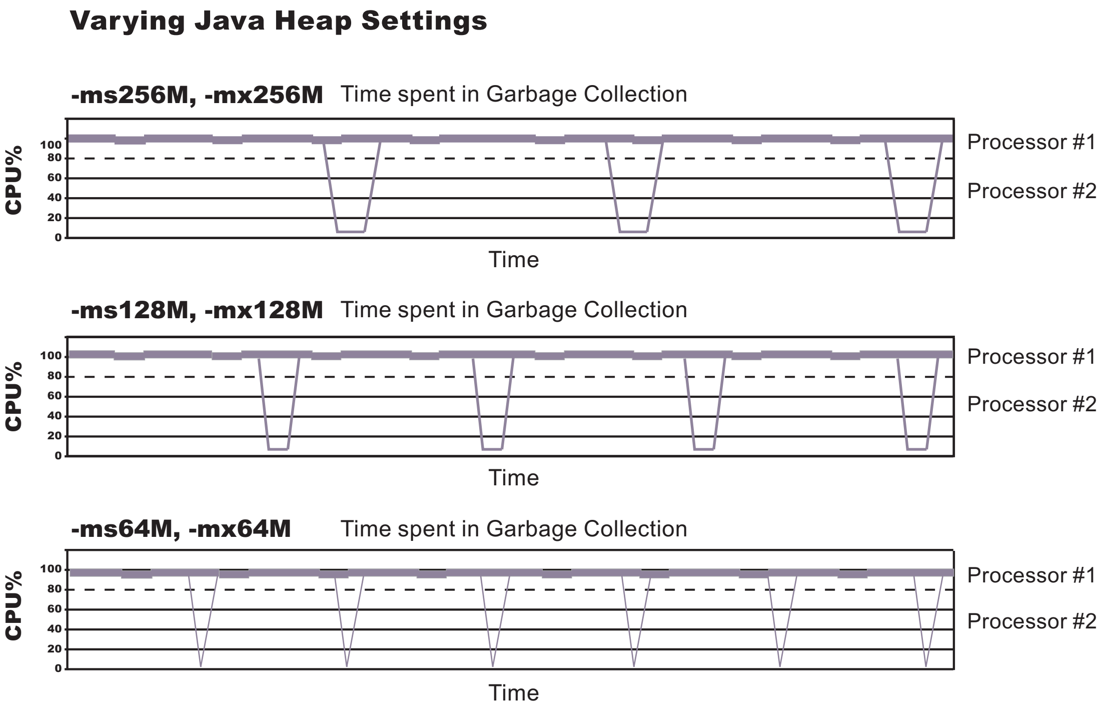
Java heap parameters

The Java heap parameters also influence the behavior of garbage collection. Increasing the heap size supports more object creation. Because a large heap takes longer to fill, the application runs longer before a garbage collection occurs. However, a larger heap also takes longer to compact and causes garbage collection to take longer. See the *Tuning guide* PDF for more information about heap settings.

For performance analysis, the initial and maximum heap sizes should be equal.

Initial heap size

When tuning a production system where the working set size of the Java application is not understood, a good starting value for the initial heap size is 25% of the maximum heap size. The JVM then tries to adapt the size of the heap to the working set size of the application.



The illustration represents three CPU profiles, each running a fixed workload with varying Java heap settings. In the middle profile, the initial and maximum heap sizes are set to 128MB. Four garbage collections occur. The total time in garbage collection is about 15% of the total run. When the heap parameters are doubled to 256MB, as in the top profile, the length of the work time increases between garbage collections. Only three garbage collections occur, but the length of each garbage collection is also increased. In the third profile, the heap size is reduced to 64MB and exhibits the opposite effect. With a smaller heap size, both the time between garbage collections and the time for each garbage collection are shorter. For all three configurations, the total time in garbage collection is approximately 15%. This example illustrates an important concept about the Java heap and its relationship to object utilization. There is always a cost for garbage collection in Java applications.

Run a series of test experiments that vary the Java heap settings. For example, run experiments with 128MB, 192MB, 256MB, and 320MB. During each experiment, monitor the total memory usage. If you expand the heap too aggressively, paging can occur. Use the **vmstat** command or the Windows 2000/2003 Performance Monitor to check for paging. If paging occurs, reduce the size of the heap or add more memory to the system. When all the runs are finished, compare the following statistics:

- Number of garbage collection calls
- Average duration of a single garbage collection call
- Ratio between the length of a single garbage collection call and the average time between calls

If the application is not over utilizing objects and has no memory leaks, the state of steady memory utilization is reached. Garbage collection also occurs less frequently and for short duration.

If the heap free space settles at 85% or more, consider decreasing the maximum heap size values because the application server and the application are under-utilizing the memory allocated for heap.

Configuration update performance in a large cell configuration

In a large cell configuration, you might have to determine whether configuration update performance or consistency checking is more important. When configuration consistency checking is turned on, a large amount of time might be required to save a configuration change or to deploy a large number of applications. The following factors influence how much time is required:

- The more application servers or clusters there are defined in cell, the longer it takes to save a configuration change.
- The more applications there are deployed in a cell, the longer it takes to save a configuration change.

If the amount of time required to change a configuration change is unsatisfactory, you can add the config_consistency_check custom property to your JVM settings and set the value of this property to false.

1. In the administrative console, click **Servers > Application servers > server1**.
2. Under Server Infrastructure, select Java and Process Management, and then click **Process Definition**.
3. Under Additional Properties, click **Java Virtual Machine > Custom Properties > New**.
4. Enter config_consistency_check in the Name field and false in the Value field.
5. Click **OK** and then **Save** to apply these changes to the master configuration.
6. Restart the server.

If you are using the WebSphere Application Server wsadmin command wsadmin -conntype none in local mode, you must set the config_consistency_check property to false before issuing this command.

Tuning application servers

The WebSphere Application Server contains interrelated components that must be harmoniously tuned to support the custom needs of your end-to-end e-business application.

This group of interrelated components is known as the queuing network. The queuing network helps the system achieve maximum throughput while maintaining the overall stability of the system.

The following steps describe various tuning tasks that may improve your application server performance. You can choose to implement any of these application server settings. These steps can be performed in any order.

1. **Tune the object request broker.** An Object Request Broker (ORB) manages the interaction between clients and servers, using the Internet InterORB Protocol (IIOP). It supports client requests and responses received from servers in a network-distributed environment. You can use the following parameters to tune the ORB:
   - Set **Pass by reference (com.ibm.CORBA.iiop.noLocalCopies)** as described in the *Tuning guide* PDF.
   - Set the **Connection cache minimum (com.ibm.CORBA.MaxOpenConnections)** as described in the *Tuning guide* PDF.
   - Set **Maximum size** as described in Thread pool settings
   - Set **com.ibm.CORBA.ServerSocketQueueDepth** as described in the *Administering applications and their environment* PDF.
   - Set the **com.ibm.CORBA.FragmentSize** as described in the *Administering applications and their environment* PDF.
2. **Tune the XML parser definitions.**
   - **Description:** Facilitates server startup by adding XML parser definitions to the jaxp.properties and xerxes.properties files in the ${*app_server_root*}/jre/lib directory. The XMLParserConfiguration value might change as new versions of Xerces are provided.
   - **How to view or set:** Insert the following lines in both files:

     ```
     javax.xml.parsers.SAXParserFactory=org.apache.xerces.jaxp.SAXParserFactoryImpl
     javax.xml.parsers.DocumentBuildFactory=org.apache.xerces.jaxp.
               DocumentBuilderFactoryImpl
     org.apache.xerces.xni.parser.XMLParserConfiguration=org.apache.xerces.parsers.
              StandardParserConfiguration
     ```
   - **Default value:** None
   - **Recommended value:** None
3. **Tune the dynamic cache service.**

Using the dynamic cache service can improve performance. See the *Administering applications and their environment* PDF for information about using the dynamic cache service and how it can affect your application server performance.

4. **Tune the Web container.** The WebSphere Application Server Web container manages all HTTP requests to servlets, JavaServer Pages and Web services. Requests flow through a transport chain to the Web container. The transport chain defines the important tuning parameters for performance for the Web container. There is a transport chain for each TCP port that WebSphere Application Server is listening on for HTTP requests. For example, the default HTTP port 9080 is defined in Web container inbound channel chain. Use the following parameters to tune the Web container:
   - HTTP requests are processed by a pool of server threads. The minimum and maximum thread pool size for the Web container can be configured for optimal performance. Generally, 5 to 10 threads per server CPU will provide the best throughput. The number of threads configured does not represent the number of requests WebSphere can process concurrently. Requests are queued in the transport chain when all threads are busy. To specify the thread pool settings:
     a. Click **Servers > Application Servers >** *server_name* **Web Container Settings> Web Container > Web container transport chains**.
     b. Select the normal inbound chain for serving requests. This will usually be named WCInboundDefault, on port 9080.
     c. Click **TCP Inbound Channel (TCP_2)**.
     d. Set **Thread Pools** under Related Items.
     e. Select **WebContainer**.
     f. Enter values for **Minimum Size** and **Maximum Size**.
   - The HTTP 1.1 protocol provides a keep-alive feature to enable the TCP connection between HTTP clients and the server to remain open between requests. By default WebSphere Application Server will close a given client connection after a number of requests or a timeout period. After a connection is closed, it will be recreated if the client issues another request. Early closure of connections can reduce performance. Enter a value for the maximum number of persistent requests to (keep-alive) to specify the number of requests that are allowed on a single HTTP connection. Enter a value for persistent timeouts to specify the amount of time, in seconds, that the HTTP transport channel allows a socket to remain idle between requests. To specify values for Maximum persistent requests and Persistent timeout:
     a. Click **Servers > Application Servers >** *server_name* **Web Container Settings> Web Container > Web container transport chains**.
     b. Select the normal inbound chain for serving requests. This will usually be named WCInboundDefault, on port 9080.
     c. Click **HTTP Inbound Channel (HTTP_2)**.
     d. Enter values for **Maximum persistent requests** and **Persistent timeout**.
5. **Tune the EJB container.** An Enterprise JavaBeans (EJB) container is automatically created when you create an application server. After the EJB container is deployed, you can use the following parameters to make adjustments that improve performance.
   - Set the **Cleanup interval** and the **Cache size** as described in the *Administering applications and their environment* PDF.
   - **Break CMP enterprise beans into several enterprise bean modules** while assembling EJB modules.

   See also the *Tuning guide* PDF.
6. **Tune the session management.**

   The installed default settings for session management are optimal for performance. See the *Tuning guide* PDF for more information about tuning session management.
7. **Tune the data sources and associated connection pools.** A data source is used to access data from the database; it is associated with a pool of connections to that database.
   - Review information on connection pools, contained in the *Administering applications and their environment* PDF, to understand how the number of physical connections within a connection pool can change performance.

- Use the *Tuning guide* PDF as a reference for the data source and connection pool properties that most affect performance.

8. **Tune the URL invocation cache.**

   Each JavaServer Page is a unique URL. If you have more than 50 unique URLs that are actively being used, increase the value specified for the invocationCacheSize JVM custom property. This property controls the size of the URL invocation cache. See the *Administering applications and their environment* PDF for more information on how to change this property.

Web services client to Web container optimized communication

To improve performance, there is an optimized communication path between a Web services client application and a Web container that are located in the same application server process. Requests from the Web services client that are normally sent to the Web container using a network connection are delivered directly to the Web container using an optimized local path. The local path is available because the Web services client application and the Web container are running in the same process.

This direct communication eliminates the need for clients and web containers that are in the same process to communicate over the network. For example, a Web services client might be running in an application server. Instead of accessing the network to communicate with the Web container, the Web services client can communicate with the Web container using the optimized local path. This optimized local path improves the performance of the application server by allowing Web services clients and Web containers to communicate without using network transports.

In a clustered environment, there is typically an HTTP server (such as IBM HTTP server) that handles incoming client requests, distributing them to the correct application server in the cluster. The HTTP server uses information about the requested application and the defined virtual hosts to determine which application server receives the request. The Web services client also uses the defined virtual host information to determine whether the request can be served by the local Web container. You must define unique values for the host and port on each application server. You cannot define the values of host and port as wild cards denoted by the asterisk symbol (*) when you enable the optimized communication between the Web services application and the Web container. Using wild cards indicate that the local Web container can handle Web services requests for all destinations.

The optimized local communication path is disabled by default. You can enable the local communication path with the enableInProcessConnections custom property. Before configuring this custom property, make sure that you are not using wild cards for host names in your Web container end points. Set this property to **true** in the Web container to enabled the optimized local communication path. When disabled, the Web services client and the Web container communicate using network transports.

For information about how to configure the enableInProcessConnections custom property, see the *Administering applications and their environment* PDF.

When the optimized local communication path is enabled, logging of requests through the local path uses the same log attributes as the network channel chain for the Web container. To use a different log file for in process requests than the log file for network requests, use a custom property on the HTTP Inbound Channel in the transport chain. Use the inProcessLogFilenamePrefix custom property to specify a string that is added to the beginning of the network log file name to create a file name that is unique. Requests through the local process path are logged to this specified file. For example, if the log filename is ../httpaccess.log for a network chain, and the inProcesslLogFilenamePrefix custom property is set to “local” on the HTTP channel in that transport chain, the local log file name for requests to the host associated with that chain is /localhttpaccess.log.

Application servers: Resources for learning

Use the following links to find relevant supplemental information about configuring application servers. 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 instructions and examples
- Programming specifications
- Administration

Programming instructions and examples

- WebSphere Application Server education

Programming specifications

- The Java^{TM} Virtual Machine Specification, Second Edition
- Sun's technology forum for the Java^{TM} Virtual Machine Specification

Administration

- Listing of all IBM WebSphere Application Server Redbooks

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 - Express.

Default product locations when the root user or an administrator user installs the product

The root user or administrator user (on a Windows system) is capable of registering shared products and installing into system-owned directories. The following default directories are system-owned directories.

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 - Express products or components, of course, require multiple locations.

app_server_root - the install_root for WebSphere Application Server

The following list shows default installation root directories for WebSphere Application Server - Express:

AIX /usr/IBM/WebSphere/AppServer

HP-UX /opt/IBM/WebSphere/AppServer

Linux /opt/IBM/WebSphere/AppServer

Solaris /opt/IBM/WebSphere/AppServer

Windows C:\Program Files\IBM\WebSphere\AppServer

profile_root

The following list shows the default directory for a profile named *profile_name* on each distributed operating system:

AIX /usr/IBM/WebSphere/AppServer/profiles/*profile_name*

HP-UX /opt/IBM/WebSphere/AppServer/profiles/*profile_name*

Linux /opt/IBM/WebSphere/AppServer/profiles/*profile_name*

Solaris /opt/IBM/WebSphere/AppServer/profiles/*profile_name*

Windows C:\Program Files\IBM\WebSphere\AppServer\profiles\profile_name

plugins_root

The following default installation root is for the Web server plug-ins for WebSphere Application Server:

AIX /usr/IBM/HTTPServer/Plugins

HP-UX /opt/IBM/HTTPServer/Plugins

Linux /opt/ibm/HTTPServer/Plugins

Solaris /opt/IBM/HTTPServer/Plugins

Windows C:\Program Files\IBM\HTTPServer\Plugins

web_server_root

The following default installation root directories are for the IBM HTTP Server:

AIX /usr/IBM/HTTPServer

HP-UX /opt/IBM/HTTPServer

Linux /opt/ibm/HTTPServer

Solaris /opt/IBM/HTTPServer

Windows C:\Program Files\IBM\HTTPServer

gskit_root

The following list shows the default installation root directories for Version 7 of the IBM Global Security Kit (GSKit):

AIX /usr/ibm/gsk7

HP-UX /opt/ibm/gsk7

Linux /opt/ibm/gsk7

Solaris /opt/ibm/gsk7

Windows C:\Program Files\IBM\GSK7

app_client_root

The following default installation root directories are for the WebSphere Application Client:

AIX /usr/IBM/WebSphere/AppClient (J2EE Application client only)

HP-UX /opt/IBM/WebSphere/AppClient (J2EE Application client only)

Linux /opt/IBM/WebSphere/AppClient (J2EE Application client only)

Solaris /opt/IBM/WebSphere/AppClient (J2EE Application client only)

Windows C:\Program Files\IBM\WebSphere\AppClient

updi_root

The following list shows the default installation root directories for the Update Installer for WebSphere Software:

AIX /usr/IBM/WebSphere/UpdateInstaller

HP-UX /opt/IBM/WebSphere/UpdateInstaller

Linux /opt/IBM/WebSphere/UpdateInstaller

Solaris /opt/IBM/WebSphere/UpdateInstaller

Windows C:\Program Files\IBM\WebSphere\UpdateInstaller

cip_app_server_root

The following list shows the default installation root directories for a customized installation package (CIP) produced by the Installation Factory.

A CIP is a WebSphere Application Server - Express product bundled with one or more maintenance packages, an optional configuration archive, one or more optional enterprise archive files, and other optional files and scripts:

AIX /usr/IBM/WebSphere/AppServer/cip/*cip_uid*

HP-UX /opt/IBM/WebSphere/AppServer/cip/*cip_uid*

Linux /opt/IBM/WebSphere/AppServer/cip/*cip_uid*

Solaris /opt/IBM/WebSphere/AppServer/cip/*cip_uid*

Windows C:\Program Files\IBM\WebSphere\AppServer\cip\cip_uid

The *cip_uid* variable is the CIP unique ID generated during creation of the build definition file. You can override the generated value in the Build definition wizard. Use a unique value to allow multiple CIPs to install on the system.

component_root

The component installation root directory is any installation root directory described in this topic. Some programs are for use across multiple components. In particular, the Update Installer for WebSphere Software is for use with WebSphere Application Server - Express, Web server plug-ins, the Application Client, and the IBM HTTP Server. All of these components are part of the product package.

Default product locations when a non-root user or a non-administrator user installs the product

The non-root user or non-administrator user (on a Windows system) is not capable of registering shared products and installing into system-owned directories. The following default directories are user-owned directories in the home directory of the non-root installer as opposed to being globally shared resources that are available to all users.

app_server_root

The following list shows the default installation directories for non-root installation of WebSphere Application Server - Express:

AIX *user_home*/IBM/WebSphere/AppServer

HP-UX *user_home*/IBM/WebSphere/AppServer

Linux *user_home*/IBM/WebSphere/AppServer

Solaris *user_home*/IBM/WebSphere/AppServer

Windows C:\IBM\WebSphere\AppServer

profile_root

The following list shows the default directories for creating profiles:

AIX *user_home*/IBM/WebSphere/AppServer/profiles/

HP-UX *user_home*/IBM/WebSphere/AppServer/profiles/

Linux *user_home*/IBM/WebSphere/AppServer/profiles/

Solaris *user_home*/IBM/WebSphere/AppServer/profiles/

Windows C:\IBM\WebSphere\AppServer\profiles\

web_server_root

The following default installation root directories are for the IBM HTTP Server:

AIX *user_home*/IBM/HTTPServer

HP-UX *user_home*/IBM/HTTPServer

Linux *user_home*/ibm/HTTPServer

Solaris *user_home*/IBM/HTTPServer

Windows C:\IBM\HTTPServer

plugins_root

The following list shows the default installation root directories for the Web server plug-ins for WebSphere Application Server:

AIX *user_home*/IBM/HTTPServer/Plugins

HP-UX *user_home*/IBM/HTTPServer/Plugins

Linux *user_home*/ibm/HTTPServer/Plugins

Solaris *user_home*/IBM/HTTPServer/Plugins

Windows C:\IBM\HTTPServer\Plugins

app_client_root

The following list shows the default installation root directories for the WebSphere Application Client:

AIX *user_home*/IBM/WebSphere/AppServer/AppClient (J2EE Application client only)

HP-UX *user_home*/IBM/WebSphere/AppClient (J2EE Application client only)

Linux *user_home*/IBM/WebSphere/AppClient (J2EE Application client only)

Solaris *user_home*/IBM/WebSphere/AppClient (J2EE Application client only)

Windows C:\IBM\WebSphere\AppClient

updi_root

The following list shows the default installation directories for non-root installation of WebSphere Application Server - Express:

AIX *user_home*/IBM/WebSphere/UpdateInstaller

HP-UX *user_home*/IBM/WebSphere/UpdateInstaller

Linux *user_home*/IBM/WebSphere/UpdateInstaller

Solaris *user_home*/IBM/WebSphere/UpdateInstaller

Windows C:\Program Files\IBM\WebSphere\UpdateInstaller

cip_app_server_root

The following list shows the default installation root directories for a WebSphere Application Server - Express product CIP:

AIX *user_home*/IBM/WebSphere/AppServer/cip/*cip_uid*

HP-UX *user_home*/IBM/WebSphere/AppServer/cip/*cip_uid*

Linux *user_home*/IBM/WebSphere/AppServer/cip/*cip_uid*

Solaris *user_home*/IBM/WebSphere/AppServer/cip/*cip_uid*

Windows C:\IBM\WebSphere\AppServer\cip\cip_uid

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

Trademarks and service marks

For trademark attribution, visit the IBM Terms of Use Web site (http://www.ibm.com/legal/us/).