IBM WebSphere Application Server, Version 5.1

Servers and Environment

Note

Before using this information, be sure to read the general information under "Notices" on page 89.

Compilation date: December 15, 2003

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. Welcome to Servers

Application servers

For an overview, refer to Chapter 2, “Welcome to Application Servers,” on page 3.

Chapter 2. Welcome to Application Servers

Overview

Application servers extend the ability of a Web server to handle Web application requests. An application server enables a server to generate a dynamic, customized response to a client request.

You can configure one or more application servers and enhance the operation of an application server, using:
- Transports
- Custom services
- Command-line information that passes to a server when it starts or initializes
- Settings that improve the use of the Java virtual machine (JVM).

See Chapter 3, “Configuring application servers,” on page 5.

Application servers use an Object Request Broker (ORB) for RMI/IIOP communication.

Asynchronous messaging

The product supports asynchronous messaging based on the Java Messaging Service (JMS) of a JMS provider that conforms to the JMS specification version 1.0.2 and supports the Application Server Facility (ASF) function defined within that specification.

For IBM WebSphere Application Server, the JMS functions (of the JMS provider) for an application server are served by the JMS server within the application server.

For more information, refer to the V5 InfoCenter at:
`http://publib.boulder.ibm.com/infocenter/wasinfo/index.jsp`

.

Chapter 3. Configuring application servers

An application server configuration provides settings that control how an application server provides services for running enterprise applications and their components.

This section describes how to create and configure application servers, and how to otherwise handle server configurations.

A WebSphere Application Server administrator can configure one or more application servers and perform tasks such as the following:
1. Create application servers.
2. Manage application servers.
3. Configure transports.
4. Develop custom services.
5. Define processes for the application server. As part of defining processes, you can define process execution statements for starting or initializing a UNIX process, monitoring policies to track the performance of a process, process logs to which standard out and standard error streams write, and name-value pairs for properties.
6. Use the Java virtual machine.

After preparing a server, deploy an application or component on the server. See "“Preparing to host applications” on page 40" for a sample procedure that you might follow in configuring the application server run-time and resources.

Application servers

Application servers extend a Web server’s capabilities 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, suppose--
1. A user at a Web browser on the public 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 a WebSphere Application Server product.
4. The WebSphere Application Server product 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.

The WebSphere Application Server product provides multiple application servers that can be either separately configured processes or nearly identical clones.

Creating application servers

You can create a new application server using the wsadmin tool or the Create New Application Server page of the administrative console.

Note: For the Base WebSphere Application Server product, although you can use the administrative console to create a new server definition, you cannot use it to start, stop or, in any way, control or manage that server. The Base product administrative console can only be used to create server definitions and, if necessary, adjust the server definitions that it creates. To manage Base application servers, use the wasadmin tool or the command line tools such as `startServer` and `stopServer`.

For the Base WebSphere Application Server product, you must manage each application server from the console that is hosted by that application server process.

The steps below describe how to use the Create New Application Server page.

1. Go to the Application Servers page and click **New**. This brings you to the Create New Application Server page.
2. Follow the instructions on the Create New Application Server page and define your application server.
   a. Select a node for the application server.
   b. Type in a name for the application server. The name must be unique within the node.
   c. Select whether the new server will have unique ports for each HTTP transport. By default, this option is enabled. If you select this option, you might need to update the alias list for the virtual host that you plan to use with this server to contain these new port values. If you deselect this option, ensure that the default port values do not conflict with other servers on the same physical machine.
   d. Select a template to be used in creating 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.
   e. If you create the new server using an existing application server as a model, select whether to map applications from the existing server to the new server. By default, this option is disabled.
3. To use multiple language encoding support in the administrative console, configure an application server with UTF-8 encoding enabled.

The new application server appears in the list of servers on the Application Servers page.

Note that the application server created has many default values specified for it. An application server has many properties that can be set and creating an application server on the Create New Application Server page specifies values for only a few of the important properties. To view all of the properties of your application server and to customize your application server further, click on the name of your application server on the Application Servers page and change the settings for your application server as needed.

Configuring application servers for UTF-8 encoding

To use multiple language encoding support in the administrative console, you must configure an application server with UTF-8 encoding enabled.

1. Create an application server or use an existing application server.
2. On the Application Server page, click on the name of the server you want enabled for UTF-8.
3. On the settings page for the selected application server, click **Process Definition**.
4. On the Process Definition page, click **Java Virtual Machine**.
5. On the Java Virtual Machine page, specify `-Dclient.encoding.override=UTF-8` for **Generic JVM Arguments** and click **OK**.
6. Click **Save** on the console taskbar.
7. Restart the application server.

Note that the autoRequestEncoding option does not work with UTF-8 encoding enabled. The default behavior for WebSphere Application Server is, first, to check if charset is set on content type header. If it is, then the product uses content type header for character encoding; if it is not, then the product uses character encoding set on server using the system property default.client.encoding. If charset is not present and the system property is not set, then the product uses ISO-8859-1. Enabling autoRequestEncoding on a Web module changes the default behavior: if charset it not present on an incoming request header, the product checks the Accept-Language header of the incoming request and does encoding using the first language found in that header. If there is no charset on content type header and no Accept language header, then the product uses character encoding set on server using the system property default.client.encoding. As with the default behavior, if charset is not present and the system property is not set, then the product uses ISO-8859-1.

Managing application servers

To view information about an application server, use the Application Servers of the administrative console..

Network Deployment and z/OS WebSphere Application Server

For the Network Deployment and z/OS products, you can use the Application Servers page of the administrative console to manage application servers. Or, if you prefer, you can also use the wasadmin tool or command line tools such as `startServer` and `stopServer` for managing application servers.

Base WebSphere Application Server

For the Base WebSphere Application Server product, although you can use the administrative console to create a new server definition, you cannot use it to start, stop or, in any way, control or manage that server. The Base product administrative console can only be used to create server definitions and, if necessary, adjust the server definitions that it creates. To manage Base application servers, use the wasadmin tool or the command line tools such as `startServer` and `stopServer`.

For the Base WebSphere Application Server product, you must manage each application server from the console that is hosted by that application server process.

1. Access the Application Servers page. Click **Servers > Application Servers** in the console navigation tree.
2. View information about application servers. The Application Servers page lists application servers in the cell and the nodes holding the application servers.

   To view additional information about a particular application server or to further configure an application server, click on the application server name under **Name**. This accesses the settings page for an application server.

   To view product information for an application server:

   a. Verify that the application server is running.
   b. Display the **Runtime** tab on the settings page for an application server.
   c. Click **Product Information**.

   The Product Information page displayed lists the WebSphere Application Server products installed for the application server, the version and build levels for the products, the build dates, and any interim fixes applied to the application server.
3. Create an application server. Click **New** and follow the instructions on the Create New Application Server page.
4. Monitor the running of application servers.
5. Delete an Application Server.
   a. Click **Servers > Application Servers** in the console navigation tree to access the Application Servers page.
   b. Place a checkmark in the check box beside an application server to delete it.
   c. Click **Delete**.
   d. Click **OK** to confirm the deletion.

Server collection

Use this page to view information about and manage application and JMS servers.

Application Servers

The Application Servers page lists application servers in the cell and the nodes holding the application servers.

The Network Deployment product also shows the status of the application servers. The status indicates whether a server is running, stopped, or encountering problems.

To view this administrative console page, click **Servers > Application Servers**.

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**.

Related concepts

"Application servers" on page 5

Cells
Cells are arbitrary, logical groupings of one or more nodes in a WebSphere Application Server distributed network.

Node
A node is a logical grouping of managed servers.

Related tasks

Chapter 3, "Configuring application servers," on page 5

Managing JMS servers on Application Server
Use this task to manage JMS servers on WebSphere Application Server.

Related reference

Administrative console buttons
This page describes the button choices that are available on various pages of the administrative console, depending on which product features you have enabled.

Administrative console filter settings
Use the Filter settings to specify how to filter entries shown in a collection view.

Administrative console preference settings
Use the Preference settings to specify how you would like information to be displayed on an administrative console page.

Administrative console page features
This topic provides information about the basic elements of an administrative console page, such as the various tabs one can expect to encounter.

Object names

Administrative console scope settings
Use Scope settings to filter the contents of an administrative console collection table to a particular cell, node, or server. Changing the value for Scope allows you to see other variables that apply to an object and might change the contents of the collection table.

Related information

An overview of WebSphere asynchronous messaging using JMS

Name

Specifies a logical name for the server. Server names must be unique within a node.

Node

Specifies the name of the node for the application server.

Status

Indicates whether the application 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.
To view this administrative console page, click **Servers > Application Servers >***server_name*.

The **Configuration** tab provides editable fields and the **Runtime** tab provides 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.

Data type	String
Default	server1

Initial State:

Specifies the component execution state requested when the server is first started.

Data type	String
Default	Started

Application Class loader Policy:

Specifies whether to use a single class loader to load all applications or to use a different class loader for each application.

The options are SINGLE and MULTIPLE. The default is to use a separate class loader for each application (MULTIPLE).

Data type	String
Default	MULTIPLE

Application Classloading 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 class loaders is PARENT_FIRST. By specifying PARENT_LAST, your application can override classes contained in the parent class loader, but this action can potentially result in ClassCastException or LinkageErrors if you have mixed use of overridden classes and non-overridden classes.

The options are PARENT_FIRST and PARENT_LAST. The default is to search in the parent class loader before searching in the application class loader to load a class.

Data type	String
Default	PARENT_FIRST

Short name:

Specifies the short name of the server.

The name is 1-8 characters, alpha-numeric or national language. It cannot start with a numeric.

The system assigns a cell-unique, default short name.

Unique Id:

Specifies the unique ID of this server.

The unique ID property is read only. The system automatically generates the value.

Process ID:

Specifies a string identifying the process.

Data type	String

Cell Name:

Specifies the name of the cell for the application server.

Data type	String
Default	*host_name*Network

Node Name:

Specifies the name of the node for the application server.

Data type	String

State:

Indicates whether the application server is started or stopped.

Data type	String
Default	Started

End point collection:

Use this page to view and manage communication end points used by run-time components running within a process. End points provide host and port specifications for a server.

To view this administrative console page, click **Servers > Application Servers >***server_name* **> End Points**.

Note that this page displays only when you are working with end points for application servers.

End Point Name:

Specifies the name of an end point. Each name must be unique within the server.

End point settings:

Use this to view and change the configuration for a communication end point used by run-time components running within a process. An end point provides host and port specifications for a server.

To view this administrative console page, click **Servers > Application Servers >***server_name* **> End Points >***end_point_name*

End Point Name:

Specifies the name of the end point. The name must be unique within the server.

Note that this field displays only when you are defining an end point for an application server.

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`.

Data type	String
Default	*

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.

Data type	Integer
Default	None

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.

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* **> Custom Properties** >*property_name*

Name:

Specifies the name (or key) for the property.

Data type	String

Value:

Specifies the value paired with the specified name.

Data type	String

Description:

Provides information about the name-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* **> 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* **> 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

Thread pool settings:

Use this page to configure a group of threads that an application server uses. Requests are sent to the server through any of the HTTP transports. A thread pool enables components of the server to reuse threads to eliminate the need to create new threads at run time. Creating new threads expends time and resources.

To view this administrative console page, click **Servers > Manage Application Servers >***server_name* **> ORB Service > Thread Pool**. (You can reach this page through more than one navigational route.)

Minimum size:

Specifies the minimum number of threads to allow in the pool.

Data type	Integer
Default	10

Maximum size:

Specifies the maximum number of threads to allow in the pool.

If your Tivoli Performance Viewer shows the Percent Maxed metric to remain consistently in the double digits, consider increasing the Maximum size. The Percent Maxed metric indicates the amount of time that the configured threads are used. If there are several simultaneous clients connecting to the server-side ORB, increase the size to support up to 1000 clients.

Data type	Integer
Default	50
Recommended	50 (25 on Linux systems)

Thread inactivity timeout:

Specifies the number of milliseconds of inactivity that should elapse before a thread is reclaimed. A value of 0 indicates not to wait and a negative value (less than 0) means to wait forever.

Note: The administrative console does not allow you to set the inactivity timeout to a negative number. To do this you must modify the value directly in the *config.xml* file.

Data type	Integer
Units	Milliseconds
Default	3500

Is Growable:

Specifies whether the number of threads can increase beyond the maximum size configured for the thread pool.

Data type	Boolean
Default	Not enabled (false)

Range	Valid values are Allow thread allocation beyond maximum thread size or Not enabled.

WebSphere internal JMS server settings

The JMS functions on a node within the WebSphere Application Server administration domain are served by the JMS server on that node. Use this panel to view or change the configuration properties of the selected JMS server.
You can use this panel to configure a general set of JMS server properties, which add to the default values of properties configured automatically for the embedded WebSphere JMS provider.

Note: In general, the default values of WebSphere MQ properties are adequate for WebSphere internal JMS servers. However, if you are running high messaging loads, you may need to change some WebSphere MQ properties; for example, properties for log file locations, file pages, and buffer pages. For more information about configuring WebSphere MQ properties, see the *WebSphere MQ System Administration* book, SC33-1873, which is available from the IBM Publications Center or from the WebSphere MQ collection kit, SK2T-0730.

Name:

The name by which the JMS server is known for administrative purposes within IBM WebSphere Application Server.

This name should not be changed.

Data type	String
Units	Not applicable
Default	WebSphere Internal JMS Server
Range	Not applicable

Description:

A description of the JMS server, for administrative purposes within IBM WebSphere Application Server.

This string should not be changed.

Data type	String
Default	WebSphere Internal JMS Server

Number of threads:

The number of concurrent threads to be used by the publish/subscribe matching engine

The number of concurrent threads should only be set to a small number.

Data type	Integer
Units	Threads
Default	1
Range	Greater than or equal to 1.

Queue Names:

The names of the queues hosted by this JMS server. Each queue name must be added on a separate line.

Each queue listed in this field must have a separate queue administrative object with the same administrative name. To make a queue available to applications, define a WebSphere queue and add its name to this field on the JMS Server panel for the host on which you want the queue to be hosted.

Data type	ASCII
Units	Queue name
Default	Not applicable
Range	Each entry in this field is a queue name of up to 45 characters, which must match exactly (including use of upper- and lowercase characters) the WebSphere queue administrative object defined for the queue.

Initial state:

The state that you want the JMS server to have when it is next restarted.

Data type	Enum
Units	Not applicable
Default	Started
Range	**Started** The JMS server is started automatically. **Stopped** The JMS server is not started automatically. If any deployed enterprise applications are to use JMS server functions provided by the JMS server, the system administrator must start the JMS server manually or select the Started value of this property then restart the JMS server. To restart the JMS server for an application server, stop then restart that application server.

Starting servers

Starting a server starts a new server process based on the process definition settings of the current server configuration.

Note: If you created a new server definition using a Base WebSphere Application Server, you cannot start, stop, or manage the new server using the original base application server.

There are several options for starting an application server:

- Use the **startServer** command to start an application server from the command line.
- Start an application server for tracing and debugging.

To start the application server with standard Java debugging enabled:

1. 1. Click **Servers > Application Servers** from the administrative console navigation tree. Then, click the application server whose processes you want to trace and debug, **Process Definition**, and **Java Virtual Machine**.
2. 2. On the Java Virtual Machine page, place a checkmark in the check box for the **Debug Mode** setting to enable the standard Java debugger. If needed, set debug arguments. Then, click **OK**.
3. 3. Save the changes to a configuration file.
4. 4. Stop the application server.
5. 5. Start the application server again as described previously.

Running Application Servers from a non-root user

By default, each base WebSphere Application Server node on a Linux and UNIX platform uses the root user ID to run all Application Server processes. 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 global security is enabled, the user registry must not be Local OS. Using the Local OS user registry requires the Application Server to run as root. Refer to Local operating system user registries for details.

For the following steps, assume that:

- `was1` is the user to run the Application Server
- `mqm` is the primary user group for user was1
- `wasgroup` is another user group for user was1
- `mqm` and `mqbrkrs` are user groups associated with the Java Message Service (JMS) provider that WebSphere Application Server provides
- `wasnode` is the node name
- `server1` is the Application Server
- `/opt/WebSphere/Appserver` is the installation root

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. The user group, `mqm`, is one of the required user groups for the JMS provider that WebSphere Application Server provides. Do not change this name.
3. If you are using the JMS provider that WebSphere Application Server provides, add `was1` to groups `mqm` and `mqbrkrs`.

   The user group, `mqbrkrs`, is one of the required user groups for the WebSphere messaging provider. Do not change this name.

   The user group, `wasgroup`, is an example. You can name this user group something else.
4. Log off and back on as root.
5. Start server1 as root. Run the `startServer.sh` script from the `/bin` directory of the installation root:

   ```
   startServer.sh server1
   ```
6. 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. Click **Servers > Application Servers > server1 > Process Definition > Process Execution** and change these values:

Property	Value
Run As User	was1
Run As Group	wasgroup
UMASK	002

c. Click **OK**.

d. Save the configuration.

7. Stop the Application Server. Use the `stopServer.sh` script from the `/bin` directory of the installation root:

```
stopServer.sh server1
```

8. 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/config
chgrp -R wasgroup /opt/WebSphere/AppServer/logs
chgrp -R wasgroup /opt/WebSphere/AppServer/wstemp
chgrp -R wasgroup /opt/WebSphere/AppServer/installedApps
chgrp -R wasgroup /opt/WebSphere/AppServer/temp
chgrp -R wasgroup /opt/WebSphere/AppServer/tranlog
chgrp -R wasgroup /opt/WebSphere/AppServer/cloudscape
chgrp -R wasgroup /opt/WebSphere/AppServer/bin/DefaultDB
chmod g+w /opt/WebSphere
chmod g+w /opt/WebSphere/AppServer
chmod -R g+w  /opt/WebSphere/AppServer/config
chmod -R g+w /opt/WebSphere/AppServer/logs
chmod -R g+w /opt/WebSphere/AppServer/wstemp
chmod -R g+w /opt/WebSphere/AppServer/installedApps
chmod -R g+w /opt/WebSphere/AppServer/temp
chmod -R g+w /opt/WebSphere/AppServer/tranlog
chmod -R g+w /opt/WebSphere/AppServer/cloudscape
chmod -R g+w /opt/WebSphere/AppServer/bin/DefaultDB
```

9. If you are running the JMS provider that WebSphere Application Server provides, delete the default queue manager for the Application Server. Run the `deletemq.sh` script as root from the `/bin` directory of the installation root directory. For example, assuming that the node name is `wasnode`:

```
deletemq.sh wasnode wasnode server1
```

10. Log on to the Application Server system as `was1`.

11. If you are running the JMS provider that WebSphere Application Server provides, create the queue manager and the broker for the JMS provider that WebSphere Application Server provides. Run the `createmq.sh` script as `was1` from the `/bin` directory of the installation root. For example, assuming that the node name is `wasnode`:

```
createmq.sh /opt/WebSphere/AppServer wasnode wasnode server1
```

12. Start server1 as `was1`. Run the `startServer.sh` script from the `/bin` directory of the installation root:

```
startServer.sh server1
```

13. If running the JMS provider that WebSphere Application Server provides, verify that the MQ queue is running. Run the **dspmq** command from the `/bin` directory of the installation root:

```
dspmq
```

The name of the queue is `WAS_wasnode_server1` because the JMS provider that WebSphere Application Server provides is running on server1.

14. 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 run-time components

You must monitor the status of run-time components to ensure that, once started, they remain operational as needed.

1. Regularly examine the status of run-time components. Browse messages displayed under **Websphere Runtime Messages** in the WebSphere status area at the bottom of the console. The run-time event messages marked with a red **X** provide detailed information on event processing. You can also use the Logging and Tracing page of the administrative console to monitor the status of run-time components. Click **Troubleshooting > Logs and Trace** in the console navigation tree to access the page.
2. If an application stops running when it should be operational, examine the application's status on an Applications page and try restarting the application.
3. If the run-time components do not restart, re-examine the messages and read information on problem determination to help you to restart the components.

Stopping servers

Stopping an application server stops a server process based on the process definition settings in the current application server configuration.

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.

Transports

A transport is 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. When a user at a Web browser requests an application, the request is passed to the Web server, then along the transport to the Web container.

Transports define the characteristics of the connections between a Web server and an application server, across which requests for applications are routed. Specifically, they define the connection between the Web server plug-in and the Web container of the application server.

Administering transports is closely related to administering WebSphere Application Server plug-ins for Web servers. Indeed, without a plug-in configuration, a transport configuration is of little use.

The internal transport

The internal HTTP transport allows HTTP requests to be routed to the application server directly or indirectly through a Web server plug-in. Logging is provided for debug purposes. Prior to WebSphere Application Server Version 5.0.2, the HTTP transport functionality existed only as a means of accepting HTTP requests forwarded by an HTTP plug-in that was connected to a Web server. In Version 5.0.2, HTTP transport functionality is now a supported internal Web server. By default, the internal HTTP transport listens for HTTP requests on port 9080 and for HTTPS requests on port 9443.

For example, use the URL `http://localhost:9080/snoop` to send requests to the snoop servlet on the local machine over HTTP and `https://localhost:9443/snoop` to send requests to the snoop servlet on the local machine over HTTPS.

The transport configuration is a part of the Web container configuration. You can configure the internal transport to use ports other than 9080 and 9443, . However, you must also adjust your virtual host alias and what you type into the Web browser.

Configuring transports

A transport is 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. To define the characteristics of the connections between that plug-in and the Web container, you must specify:

- How the transport is to handle a set of connections. For example, you must specify the number of concurrent requests that is to be allowed.
- Whether to secure the connections with SSL.
- The Host and IP information for the transport participants.

1. Create an HTTP transport.
   a. Ensure that virtual host aliases include port values for the new transport.
   b. Go to the HTTP Transports page and click **New**.
   c. On the settings page for an HTTP transport, specify values such as the transport's host name and port number, then click **OK**.
2. Change the configuration for an existing transport.
   a. Ensure that virtual host aliases include port values for the transport your are changing.
   b. Go to the HTTP Transports page and click on the transport under **Host** whose configuration you want to change.
   c. On the settings page for an HTTP transport, which might have the page title DefaultSSLSettings, change the specified values as needed, then click **OK**.
   d. Custom properties page, add and set any custom properties you want to use.
3. Regenerate the WebSphere plug-in for the Web server.
4. Stop the Application Server and start it again. You must stop the Application Server and start it again before the configuration changes you made take affect.

If the Web server is located on a machine remote from the Application Server, copy the `plugin-cfg.xml` file to the remote Web server and replace the file that is there. See *IBM WebSphere Application Server Version 5.1: Installation* for information about copying the `plugin-cfg.xml` and binary plug-in module to a remote Web server and configuring the Web server to use the files.

HTTP transport collection

Use this page to view or manage HTTP transports. Transports provide request queues between WebSphere 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.

To view this administrative console page, click **Servers > Application Servers >***server_name* **> 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.

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.

To view this administrative console page, click **Servers > Application Servers >***server_name* **> 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 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 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.

You can set these properties on either the Web Container or HTTP Transport **Custom Properties** pages. 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 values for these custom properties for a specific transport on the HTTP Transport **Custom Properties** page:

1. In the console navigation tree, click **Servers > Application Servers >***server_name* **> Web Container > HTTP Transport**

To specify a custom property:

1. Click on the **HOST** whose properties you want to set.
2. Under **Additional Properties** select **Custom Properties**.
3. On the Custom Properties page, click **New**.
4. Enter the property you want to configure in the **Name** field and the value you want to set it to in the **Value** field.
5. Select **Apply**.

Following is a list of custom properties provided with the Application Server. These properties that are not shown in the settings page for an HTTP transport.

ConnectionIOTimeOut

Specifies the maximum number of seconds to wait when trying to read or process data during a request.seconds.

Data type	Integer
Default	5

ConnectionKeepAliveTimeout

Specifies the maximum number of seconds to wait for the next request on a keep alive connection.

Data type	Integer

MaxConnectBacklog

Specifies 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 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

Specifies 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

Specifies 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.

Data type	Integer
Default	100 requests

KeepAliveEnabled

Specifies whether or not to keep connections alive

Data type	Boolean.
Default	true

Configuring error logging for internal Web server HTTP transport

In order to debug potential problems with using HTTP transport as an internal Web server, you may use the following error logging capabilities.

1. To turn error logging on, add the following property to the transport section of the `server.xml` file and set the value to **false**:
   ```
   Property name: ErrorLogDisable
   Value: True/False
   Default: Error log is disabled by default
   Scope: Virtual/Global
   ```
2. To specify your own error log file, add the following property to the transport section of the `server.xml` file
   ```
   Property name: ErrorLog
   Value: <filename>
   Default: logs/<server instance>/http.log
   ```

   The error log property is used to specify where to place the error log. For example:<properties xmi:id="WebContainer_Property_6" name="ErrorLog" value="logs/<server instance>/http.log"/>

   Note: The error log should appear in each instance of the server.
3. Add the LogLevel property to the transport section of the `server.xml` file to specify the level of messages to log in error log file.
   ```
   Property name: LogLevel
   Value:  <level> (Levels include: debug, info, warn, error, crit)
   Default:  warn (warn includes error and crit; debug includes all levels)
   Scope:  Virtual/Global
   ```

   Log levels specify what type of message appears in the error log. The warn, error, and crit messages are logged by default.
4. To disable error logging, add the ErrorLogDisable property to the transport section of the `server.xml` file and set the value to **true**:
   ```
   Property name: ErrorLogDisable
   Value: True/False
   Default: Error log is disabled by default
   Scope:  Virtual/Global
   ```
5. Restart the server.

If you have enabled error logging and encounter an error, there should be an error log message in the error log file you specified.

Configuring access logging for internal Web server HTTP transport

In order to debug potential problems with using HTTP transport as an internal Web server, you may use the following access logging capabilities.

1. To turn access logging on, add the following property to the transport section of the `server.xml` file, and set the value to **false**:
   ```
   Property name: AccessLogDisable
   Values: True/False
   Default: Access log is disabled by default
   Scope:  Virtual/Global
   ```

2. To specify your own access log file, add the following property to the transport section of the `server.xml` file:

   ```
   Property name: AccessLog
   Value: <filename>
   Default Value: logs/<server instance>/http_access.log
   ```

   The default access log file is `logs/`*`<server_instance>`*`/http_access.log`. Access log entries should have the format: `<hostname or IP> <user agent> [<local time> -<status code>] <thread id> <http request> <status code> <bytecount>`.
3. To turn access logging off, add the following property to the transport section of the `server.xml` file and set the value to **true**:

   ```
   Property name: AccessLogDisable
   Values: True/False
   Default: Access log is disabled by default
   Scope:  Virtual/Global
   ```
4. Restart the server.

If you have enabled access logging, there will be an access log in the location you specified.

Custom services

A custom service provides the ability to plug into a WebSphere application server to define a hook point that runs when the server starts and shuts down.

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 run-time calls their initialize methods.

Developing custom services

To define a hook point to be run when a server starts and shuts down, you develop a custom service class and then use the administrative console to configure a custom service instance for an application server. When the application server starts, the custom service starts and initializes.

The following restrictions apply to the WebSphere 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 (OUW) 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. Execution of standard J2EE code (client code, servlets, enterprise beans) is not supported.

- The 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.

Note that 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 run-time to the initialize method can include one for an external file containing configuration information for the service (retrieved with externalConfigURLKey). In addition, the 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 throw an exception, although no specific exception subclass is defined. If an exception is thrown, the run-time logs it, disables the custom service, and proceeds with starting the server.
2. On the Custom Service page of the administrative console, click **New**. Then, on the settings page for a custom service instance, create a custom service configuration for an existing application server, supplying the name of the class implemented. If your custom services 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.
3. Stop the application server and then restart the server.
4. Check the application server to ensure that the initialize method of the custom service ran as intended. Also ensure that the shutdown method performs as intended when the server stops.

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***> 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.

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* **> Custom Services >***custom_service_name*.

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. To enable the service, place a checkmark in the check box.

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

Process definition

A process definition specifies the run-time characteristics of an application server process.

A process definition can include characteristics such as JVM settings, standard in, error and output paths, and the user ID and password under which a server runs.

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 run-time properties such as the program to run, arguments to run the program, the working directory.

1. Go to the settings page for a process definition in the administrative console. Click **Servers > Application Servers** in the console navigation tree, click on an application server name and then **Process Definition**.

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 statements for starting or initializing a UNIX 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.
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 view or change settings for a process definition, which provides command-line information for starting or initializing a process.

To view this administrative console page, click **Servers > Application Servers >***server_name* **> Process Definition**.

Start Command

Specifies the platform-specific command to launch the server process.

Control process

Data type	String
Format	START *control JCL procedure name*
Example	START BBO5ACR

Servant process

For the servant process, the value on the start command specifies the procedure name that Workload Manger (WLM) uses to start the servant process. This value is used only if the WLM Dynamic Application Environment feature is installed.

Data type	String
Format	START *servant JCL procedure name*
Example	START BBO5ASR

Start Command Args

Specifies any additional arguments required by the start command.

Control process

Data type	String
Format	JOBNAME=*server short name*,ENV=*cell short name.node short name.server short name*
Example	JOBNAME=BBOS001,ENV=SY1.SY1.BBOS001

Servant process

Data type	String
Format	JOBNAME=*server short name*S,ENV=*cell short name.node short name.server short name*
Example	JOBNAME=BBOS001S,ENV=SY1.SY1.BBOS001

Stop Command

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*
Example	STOP BBOS001;CANCEL BBOS001

Stop Command Args

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 **Note:** In this example, Stop has no arguments. Immediate Stop has the argument ARMRESTART. A semicolon precedes ARMRESTART.

Terminate Command

Specifies the platform-specific command to terminate the server process

Data type	String
Format	FORCE *server short name*
Example	FORCE BBOS001

Terminate Command Args

Specifies any additional arguments required by the terminate command.
The default is an empty string.

Data type	String
Format	*terminate command arg string*
Example	ARMRESTART

Executable Name

Specifies the executable name that is invoked to start the process.

Data type	String

Executable Arguments

Specifies the arguments that are passed to the executable when starting the process.
For example, the executable target program might expect three arguments: *arg1 arg2 arg3*.

Data type	String
Units	Java command-line arguments

Working Directory

Specifies the file system directory that the process uses as its current working directory.
This directory is used to determine the locations of input and output files with relative path names.

Passivated enterprise beans are placed in the current working directory of the application server on which the beans are running. Make sure the working directory is a known directory under the root directory of the WebSphere Application Server product.

Data type	String

Process execution settings

Use this page to view or change the process execution settings for the server process.
To view this administrative console page, click **Servers > Application Servers >***server_name* **> 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. For OS/400, additional steps are required in order to run as a userid other than QEJBSVR. See the Security section of the WebSphere Application Server for iSeries online documentation for more information.

Data type	String

For OS/400, additional steps are required in order to run as a userid other than QEJBSVR. For more information, see the Security section of the WebSphere Application Server for iSeries online documentation. Go to http://www-1.ibm.com/servers/eserver/iseries/software/websphere/wsappserver/index.html and navigate to the WebSphere Application Server for iSeries Security information.

Run As Group:

Specifies the group that the process is a member of and runs as.

On OS/400, the Run As Group setting is ignored.

Data type	String

Run In Process Group:

Specifies a specific process group for the process. This process group is useful for such things as processor partitioning. A system admininistor can assign a process group to run on, for example, 6 of 12 processors. The default (0) is not to assign the process to any specific group.

On OS/400, the Run In Process Group setting is ignored.

Data type	Integer
Default	0

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, click **Servers > Application Servers >***server_name* **> 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* **> Process Definition > 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

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

Automatic Restart:

Specifies whether the process should restart automatically if it fails. The default is to restart the process automatically.

Data type	Boolean
Default	true

Node Restart State:

Specifies the desired state for the process after the node completely shuts down and restarts. The options are: *STOPPED*, *RUNNING*, *PREVIOUS*. The default is *STOPPED*.

Data type	String
Default	STOPPED
Range	Valid values are STOPPED, RUNNING, or PREVIOUS.

Java virtual machines (JVMs)

The Java virtual machine (JVM) is an interpretive computing engine responsible for executing 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.

Using the JVM

As part of configuring an application server, you might define settings that enhance your system's use of the Java virtual machine (JVM).

To view and change the JVM configuration for an application server's process, use the Java Virtual Machine page of the console or use wsadmin to change the configuration through scripting.

1. Access the Java Virtual Machine page.
   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, click **Process Definition**.
   d. On the Process Definition page, click **Java Virtual Machine**.
2. On the Java Virtual Machine page, specify values for the JVM settings as needed and click **OK**.
3. Click **Save** on the console taskbar.
4. Restart the application server.

"“Configuring application servers for UTF-8 encoding” on page 7" 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.

"“Example: Configuring JVM sendRedirect calls to use context root” on page 38" 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 for the application server's process.

To view this administrative console page, click **Servers > Application Servers** >*server_name* **> 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.

In general, increasing the size of the Java heap improves throughput until the heap no longer resides in physical memory. After the heap begins swapping to disk, Java performance suffers drastically.

Data type	Integer
Default	96 for OS/400, 50 for all other platforms

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.

In general, increasing the size of the Java heap improves throughput until the heap no longer resides in physical memory. After the heap begins swapping to disk, Java performance suffers drastically. Set the maximum heap size low enough to contain the heap within physical memory.

Data type	Integer

Default	0 for OS/400, 256 for all other platforms. 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.
Debug arguments are only required if the Debug Mode property is set to true.

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 use by entering them into the **General JVM Arguments** field.

Note: 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%.

The **-Xquickstart** option is not supported on OS/400.

- **-Xverify:none:** When using this value, the class verification stage is skipped during class loading . By using **-Xverify:none** with the just in time (JIT) compiler enabled, startup time is improved by 10-15%.

  The **-Xverify:none** option is not supported on OS/400.
- **-Xnoclassgc:** You can use this value to disable class garbage collection, which 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=`*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.

  The **-Xgcthreads** option is not supported on OS/400.
- **-Xnocompactgc:** This value disables 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 this value 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 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 this value 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 Development Kit (JDK) Version 1.3Version 1.4.1 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)`, and `-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.

  The **-XX** option is not supported on OS/400.

- **-server | -client:** Java HotSpot Technology in the Sun-based Java Development Kit (JDK) Version 1.3Version 1.4.1 introduces 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**.

  The **-server | -client** option is not supported on OS/400.

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. When started, the process uses the JVM settings for the operating system of the node.

Data type	String

Example: 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*. 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, click **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 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 taskbar.
4. Stop the application server and then restart the application server

Tuning Java virtual machines

The application server, being a Java process, requires a Java virtual machine (JVM) to run, and to support the Java applications running on it. As part of configuring an application server, you can fine-tune settings that enhance system use of the JVM. In addition to the following tuning parameters, see also Java memory tuning tips.

Use the following JVM parameters, including garbage collection options for IBM Developer Kit 1.3.1IBM Developer Kit 1.4.1, to tune the Java virtual machine. For instructions on view and change the JVM configuration , go to Using the JVM For information on specifying any of the following settings, go to Java virtual machine settings.

- **Specify any or all of the following generic JVM arguments.** These optional command line arguments are passed to the Java virtual machine code that starts the application server process.
  - Quickstart (-Xquickstart)
  - Avoiding class verification (-Xverify:none)
  - Class garbage collection (-Xnoclassgc)
  - Garbage collection threads (-Xgcthreads)
  - Garbage collection policy (-Xgcpolicy)
  - Sun JDK Generational Garbage Collection (-XX)

    You can find more information about generational garbage collection at http://java.sun.com/docs/hotspot/gc/index.html.
  - Sun Java Development Kit 1.31.4.1 HotSpot JVM warm-up (-server)
  - Heap compaction (-Xnocompactgc)
  - Initial system heap size (-Xinitsh)
- **Set the initial heap size.**
- **Set the maximum heap size.**
- **Disable just in time (JIT) compiler.**

Preparing to host applications

The default application server and a set of default resources are available to help you begin quickly. Suppose you choose instead to configure a new server and set of resources. Here is what you need to do in order to set up a run-time environment to support applications.

1. Create an application server.
2. Create a virtual host.
3. Configure a Web container.
4. Configure an EJB container.
5. Create resources for data access.
6. Create a JDBC provider and data source.
7. Create a URL and URL provider.
8. Create a JMS destination, connection, and provider.
9. Create a JavaMail session.
10. Create resources for session support.
11. Configure a Session Manager.

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.

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, especially when running on symmetric multiprocessing (SMP) server machines. The Java virtual machine (JVM) uses a parallel garbage collector to fully exploit an SMP during most garbage collection cycles where the Sun HotSpot 1.3.1 JVM has a single-threaded garbage collector.

The garbage collection gauge

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.

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 a description of the IBM verbose:gc output and more information about the IBM garbage collector, see Performance: Resources for learning.

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.

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.

Tivoli Performance Viewer can help find memory leaks. For 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.

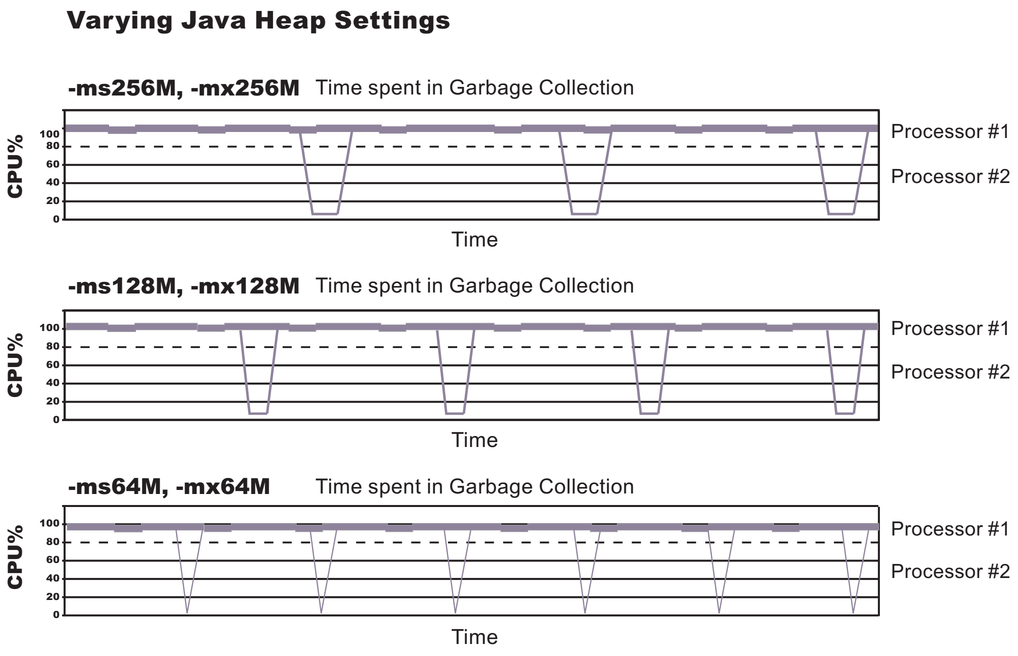
To avoid heap fragmentation, turn on the -Xcompactgc flag in the JVM advanced settings command line arguments. The -Xcompactgc function verifies that each garbage collection cycle eliminates fragmentation. However, compaction is a relatively expensive operation.

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.

For performance analysis, the initial and maximum heap sizes should be equal.

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 NT or Windows 2000 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.

For current information available from IBM Support on known problems and their resolution, see the IBM Support page.

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.

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

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 follow steps describe various tuning tasks that may improve your application server performance. You can choose to implement any of these application server settings.

- **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 tune the ORB with the following parameters:
  - **Pass by reference (com.ibm.CORBA.iiop.noLocalCopies)**
  - **Connection cache minimum (com.ibm.CORBA.MaxOpenConnections)**
  - Thread pool **Maximum size**
  - **com.ibm.CORBA.ServerSocketQueueDepth**
  - **com.ibm.CORBA.FragmentSize**

  The Object Request Broker tuning guidelines offer tips on using these parameters to tune the ORB.
- **Tune the XML parser definitions.**
  - **Description:** Facilitates server startup by adding XML parser definitions to the `jaxp.properties` and `xerxes.properties` files in the ${*install_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
- **Tune the dynamic cache service.** Using the dynamic cache service can improve performance. See Improving performance through the dynamic cache service for information about using the dynamic cache service and how it can affect your application server performance.
- **Tune the Web container.** One of the parts of each WebSphere Application Server is a Web container. To route servlet requests from the Web server to the Web containers, the product establishes a transport queue between the Web server plug-in and each Web container. The Web container is initially created with

default property values suitable for simple Web applications. However, these values might not be appropriate for more complex Web applications. Using the following parameters, you can tune the Web container to fit the specific needs of your application server.
  - Set the thread pool **Maximum size**.
  - Specify a **growable thread pool**.
  - Set **MaxKeepAliveConnections** and **MaxKeepAliveRequests** on the HTTP transport custom settings page.
- **Tune the EJB container.** An 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**.
  - Set the **Cache size**.
  - **Break CMP enterprise beans into several enterprise bean modules** while Assembling EJB modules.

  See also EJB queue tips.
- **Tune the session management.** The installed default settings for session management are optimal for performance. See Tuning session management and Tuning parameter settings for more information about tuning session management.
- **Tune the data sources.** A data source is used to access data from the database. The following parameters reveal how the number of physical connections within a connection pool can change performance.
  - Review information on **connection pooling**.
  - Set **Maximum connection pool**.
  - Set **Minimum connection pool** .
  - Set **Statement cache size**.

Chapter 4. Welcome to Environment

The environment of the product applies to the configuring of Web server plug-ins, variables, and objects that you want to be consistent throughout a cell.

Web servers

For an overview, refer to Chapter 5, “Welcome to Web servers,” on page 49.

Cell-wide settings

For an overview, refer to Chapter 7, “Welcome to Cell-wide settings,” on page 75.

Variables

For an overview, refer to “Configuring WebSphere variables” on page 82

Chapter 5. Welcome to Web servers

In the WebSphere Application Server product, an Application Server works with a Web server to handle requests for Web applications. The Application Server and Web server communicate using a WebSphere HTTP plug-in for the Web server.

If you specify a Web server when installing the WebSphere Application Server product, the installation program changes the Web server configuration file automatically to establish a plug-in.

You do not need a Web server plug-in or Web server to start the application server or the administrative console. In a test or development environment, you can use the internal HTTP transport instead of the Web server plug-in and Web server. But, for performance reasons, you must use a Web server plug-in and Web server in a production environment.

The WebSphere Application Server documentation provides information on plug-ins but does not provide information on administering Web servers. To learn how to administer your Web server, refer to documentation for your Web server. For tips on tuning your Web server see Web server tuning parameters.

Chapter 6. Configuring Web server plug-ins

A WebSphere application server works with a Web server to handle requests for Web applications. The Web server and application server communicate using a WebSphere HTTP plug-in for the Web server.

The installation program for WebSphere Application Server modifies the Web server configuration file automatically to establish a plug-in, provided that you specify a Web server during installation.

Plug-ins are the preferred method of communication between the Web server and the application server. A plug-in offers the following advantages:
- XML-based configuration file
- Standard protocol recognized by firewall products
- Security using HTTPS, replacing proprietary OSE over SSL

A Web server plug-in and Web server are not required in order to start the application server or the administrative console. In a test or development environment, you can use the internal HTTP transport instead of the Web server plug-in and Web server. But, for performance reasons, you must use a Web server plug-in and Web server in a production environment. An HTTP transport facilitates the connection between the Web server plug-in and a Web container of an application server.

1. Administer your Web server. Refer to your Web server documentation for information on administering your Web server.

   Ensure that Web servers are 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 httpd.conf for 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 IBM HTTP Server:

   HTTP method POST is not supported by this URL.
2. If you encounter problems starting your Web server, check the http_plugin.log file in the WebSphere logs directory for information on what portion of the plug-in configuration file contained the 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. A frequent reason for a Web server not starting is an improper entry in a plug-in configuration file.
3. Install the plug-in to a specific location.
4. Check the version of your IBM HTTP Server installation.
5. Regenerate the plug-in configuration or manually edit the plug-in configuration file.. After changing configurations to plug-ins, transports or virtual hosts, you must regenerate your Web server plug-in for the changes to take effect.
6. **5.1 +** Tune your Web server with Web server tuning parameters.

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.

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`.

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 allows the users to 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.

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/AppServer50/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.
- 5.1+ 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.

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.

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"/>
</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"/>
</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"/>
</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"/>
</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 10 million bytes.

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.

CloneID (zero or one attribute for each Server)

If this unique ID is present in the HTTP cookie header of a request (or the URL if using URL rewriting), the plug-in routes the request to this particular server, provided all other routing rules are met. If a CloneID is not specified in the Server, then session affinity is not enabled for this server.

This attribute is used in conjunction with session affinity. When this attribute is set, the plug-in checks the incoming cookie header or URL for **JSESSIONID**. If **JSESSIONID** is found then the plug-in looks for one or more clone IDs. If clone IDs are found, and a match is made to the value specified for this attribute, then the request is sent to this server rather than load balanced across the cluster.

If you are not using session affinity then it is best to remove these clone IDs from the configuration because there is added request processing in the plug-in when these are set. If clone IDs are not in the plug-in then it is assumed that session affinity is not on and the request is load balanced across the cluster.

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)

The weight associated with this server when the plug-in does weighted Round Robin load balancing. The algorithm for this attribute decrements all weights within the server cluster until all weights reach zero. Once a particular server's weight 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.

5.1 + 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 IHS Web server.
- Each node starts 2 processes.
- The MaxConnections attribute is set to 50.

The application server could potentially get up to 500 connections (5 nodes x 2 processes= 50 connections).

This attribute is not necessary on the z/OS platform. The z/OS control region, 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.

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.

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.

If a request comes in that does not have affinity established, the plug-in routes it to the ClusterAddress, if defined. If affinity has been established, then the plug-in routes the request directly to the clone, bypassing the ClusterAddress entirely. If no ClusterAddress is defined for the ServerCluster, then the plug-in load balances across the PrimaryServers list.

PrimaryServers (zero or one element for each ServerCluster)
Lists defined servers to which the plug-in routes requests for this cluster. If a list of PrimaryServers is not specified, the plug-in routes requests to servers defined for the ServerCluster.

BackupServers (zero or one element for each ServerCluster)
Lists servers to which requests should be sent to if all servers specified in the PrimaryServers list are unavailable. The plug-in does not load balance across the BackupServers list 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 * 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 * for the host name, a * for the port, or a * for both. A * 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 to a particular clone. The default value is **JSESSIONID**. (See the description of the CloneID attribute for additional session affinity information.)

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" newBehavior="false"
   rmEnabled="false" traceLevel="PERF_DEBUG">
```

armEnabled (zero or one attribute for RequestMetrics)

This attribute is currently ignored by the plug-in. It is for the future usage.

newBehavior (zero or one attribute for RequestMetrics)

This attribute is currently ignored by the plug-in. It is for the future usage.

rmEnabled (exactly one attribute for RequestMetrics)

This attribute indicates whether or not the request metrics is enabled in the plug-in. When it is `true`, the plug-in request metrics will look at the filters and log the request trace record in the plug-in log file. This is performed if a request passes the filters.

traceLevel (exactly one attribute for RequestMetrics)

When rmEnabled is `true`, this attribute indicates how much information is logged. When it is `NONE`, there is no request logging. When it is not `NONE`, 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.

Related tasks

"Manually editing the plug-in configuration" on page 68

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.

You have the option to install WebSphere Application Server plug-ins for your Web servers when the application server is installed.

The plug-ins use a configuration file to determine whether a request should be handled by the Web server or the application server.

Installing plug-ins to specific locations

Sometimes, you might want to install the WebSphere Application Server plug-ins for Web servers to locations other than their default locations.

Installing Web server plug-ins to non-default locations

Depending on the operating system, when you run the WebSphere Application Server installation program and select to install a Web server plug-in, the plug-in might install silently if the Web server is installed in a standard location for that Web server brand. In such a case, the WebSphere Application Server installation program does not prompt you for the configuration file location.

If you have installed two Web servers of the same brand, on the same machine, the plug-in operates properly for the server installed at the standard location, but the plug-in does not operate properly for the server that is in the non-standard location.

For example, suppose IBM HTTP Server ″installation 1″ is installed in the standard directory in which the Web server is installed on Solaris, /opt/IBMHTTPD. When the plug-in for this first Web server is installed, it silently modifies the httpd.conf file in /opt/IBMHTTPD/conf, which is appropriate.

But now suppose that you installed IBM HTTP Server ″installation 2″ in /opt/IHS2. When the Web server plug-in for this second Web server is installed, it too is installed in httpd.conf in /opt/IBMHTTPD/conf.

One way around this is to manually edit the httpd.conf file of the second IBM HTTP Server installation, rather than installing the plug-in. You could copy the modified configuration file from the first Web server and make it the configuration file of the second server. Of course, this approach assumes that you have not customized the configuration file of the second server, and find it acceptable for its configuration to match that of the first server.

Another way is to install both Web servers in some non-standard place, such as /opt/IHS1 and /opt/IHS2. When the WebSphere Application Server installation program cannot find either httpd.conf file, you are prompted for the one to edit. Specify the location of one of the Web server files. Then repeat the plug-in installation, specifying the other location this time.

Installing WebSphere Application Server plug-ins on non-default IIS servers

When the product installs the WebSphere Application Server plug-in for Microsoft Internet Information Server (IIS), it assumes the user wants to attach the plug-in to the default IIS server. The following instructions explain how to attach the plug-in to an IIS server other than the default.

The procedure might be necessary for a user implementing multiple Web sites that separate the pages they serve along some logical boundary, such as security level. Follow the steps to allow a newly defined site (using an IIS instance other than the default) to exercise servlets in conjunction with an existing site or sites.

The instructions apply to IIS Versions 4.x and 5.x.

1. Use the Internet Service Manager (from Microsoft IIS) to create a new site with a name, port number, and base subdirectory.
2. Go to the WebSphere Application Server administrative console and configure the virtual host to contain an alias for the port number used by the site.
3. Create a virtual directory for the new site.
   a. Open the Internet Service Manager for IIS.
   b. Select the new site in the tree view.
   c. Right-click to display a menu. Select **New > Virtual Directory**.
   d. Ensure that the values are set appropriately:
      - The name of the virtual directory should be set to SEPLUGINS (using all capital letters).
      - The physical path should be set to the WebSphere Application Server bin directory (such as c:\WebSphere\AppServer\bin on Windows NT).

        Note, the directory must have the EXECUTE permission set, but setting any other permissions (or allowing it to inherit other permissions) is a security risk.
4. Start the new site.
5. Issue a browser request to verify that the configuration works, such as:
   ```
   http://mymachine:8080/servlet/snoop for WebSphere Application Server V4
   or
   http://mymachine:8080/snoop for WebSphere Application Server V5
   ```

   for
6. The administrator must ensure that the SEPLUGINS retains its EXECUTE permission. It is possible for virtual directories under a site to inherit properties from the site. Ensure that the SEPLUGINS virtual directory does not inherit permissions from changes to the site, if those changes involve withdrawing the EXECUTE permission.

Checking your IBM HTTP Server version

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

1. Change directory to the installation root of the Web server. For example, this is /opt/IBMHTTPD on a Solaris machine.
2. Find the subdirectory that contains apache.exe (on a Windows platforms) or apachectl (on a UNIX-based platforms, such as z/OS, Solaris, Linux, and HP-UX)
3. On a Windows platform, issue:
   ```
   apache.exe -V
   ```
4. On a UNIX-based platforms issue:
   ```
   ./apachectl -V 4
   ```

The version is shown in the "Server version:" field and will looks something like the following:

```
Server version: IBM_HTTP_Server/2.0.47 Apache/2.0.47
Server built: Oct 2 2003 20:38:36
Server's Module Magic Number: 20020903:4.
```

Web server tuning parameters

WebSphere Application Server provides plug-ins for several Web server brands and versions. Each Web server operating system combination has specific tuning parameters that affect the application performance.

- **IBM HTTP Server**

  The IBM HTTP Server 1.3.28 is a multi-process, single-threaded server.
  - **Access logs**
    - **Description:** Collects all incoming HTTP requests. Logging degrades performance because IO operation overhead causes logs to grow significantly in a short time.
    - **How to view or set:**
      1. Open the IBM HTTP Server `httpd.conf` file, located in the directory *IBM_HTTP_Server_root_directory*/conf.
      2. Search for a line with the text **CustomLog**.
      3. Comment out this line by placing # in front of the line.
      4. Save and close the `httpd.conf` file.
      5. Stop and restart the IBM HTTP Server.
    - **Default value:** Logging of every incoming HTTP request is enabled.
    - **Recommended value:** Disable the access logs.
  - **MaxClients**
    - **Description:** The MaxClients directive controls the maximum number of simultaneous connections or users that the web server can service at any one time. If, at peak usage, your web server needs to support 200 active users at once, you should set MaxClients to 220 (200 plus an extra 10% for load growth). Setting MaxClients too low could cause some users to believe the web server is not responding. You should have sufficient RAM in your web server machines to support each connected client. For IBM HTTP Server 1.3 on Unix, you should allocate around 1.5MB MaxClients of RAM for use by the IBM HTTP Server. For IBM HTTP Server 1.3 on Windows, you should allocate around 300KB MaxClients of RAM for use by the IBM HTTP Server. Some third party modules can significantly increase the amount of RAM used per connected client.
    - **How to view or set:** Edit the MaxClients directive in the IBM HTTP Server `httpd.conf` file, located in the directory *IBM_HTTP_Server_root_directory*/conf.
    - **Default value:** 150

        - **Recommended value:** The maximum number of users normally simultaneously connected to your web server, plus an additional 10% for buffer. Note: The KeepAliveTimeout setting can affect how long a user is connected to the webserver.
    - **MinSpareServers, MaxSpareServers, and StartServers**
        - **Description:** Pre-allocates and maintains the specified number of processes so that few processes are created and destroyed as the load approaches the specified number of processes. Specifying similar values reduces the CPU usage for creating and destroying HTTPD processes. Adjust this parameter if the time waiting for IBM HTTP Server to start more servers, so that it can handle HTTP requests, is not acceptable.
        - **How to view or set:** Edit the MinSpareServers, MaxSpareServers and StartServers directives in the `httpd.conf` file located in the *IBM_HTTP_Server_root_directory*/conf directory.
        - **Default value:** MinSpareServers 5, MaxSpareServers 10, StartServers 5
        - **Recommended value:** For optimum performance, specify the same value for the MinSpareServers and the StartServers parameters. If MaxSpareServers is set to less than MinSpareServers, IBM HTTP Server resets MaxSpareServer=MinSpareServer+1. Setting the StartServers too high can cause swapping if memory is not sufficient, degrading performance.
    - **ListenBackLog**
        - **Description:** Sets the length of a pending connections queue. When several clients request connections to the IBM HTTP Server, and all threads used, a queue exists to hold additional client requests. However, if you use the default Fast Response Cache Accelerator (FRCA) feature of IHS 1.3.28 on Windows, the ListenBackLog directive is not used since FRCA has its own internal queue.
        - **How to view or set:** For non-FRCA: Edit the IBM HTTP Server `httpd.conf` file. Then, add or view the ListenBackLog directive.
        - **Default value:** For HTTP Server 1.3.28: 1024 with FRCA enabled, 511 with FRCA disabled
        - **Recommended value:** Use the defaults.
- **IBM HTTP Server - Linux**
    - **MaxRequestsPerChild**
        - **Description:** Sets the limit on the number of requests that an individual child server process handles. After the number of requests reaches the value set for the MaxRequestsPerChild parameter, the child process dies. Adjust this parameter if destroying and creating child processes is degrading your Web server performance.
        - **How to view or set:**
            1. Edit the IBM HTTP server `httpd.conf` file located in the *IBM_HTTP_Server_root_directory*/conf directory.
            2. Change the value of the parameter.
            3. Save the changes and restart the IBM HTTP server.
        - **Default value:** 500
        - **Recommended value:** Should normally be set to 0. Non-zero settings can be useful if child memory usage is observed to steadily increase over time. Memory leaks have occasionally been observed in third party modules and various OS runtime libraries used by the IBM HTTP Server.
- **IBM HTTP Server - Windows 2000 and Windows 2003**
    - **ThreadsPerChild**
        - **Description:** Sets the number of concurrent threads running at any one time within the IBM HTTP Server.
        - **How to view or set:** Edit the IBM HTTP Server file `httpd.conf` file located in the directory *IBM_HTTP_Server_root_directory*/conf. Change the value of the parameter. Save the changes and restart the IBM HTTP Server.

There are two ways to find how many threads are used under load:

1. Use the Windows 2000 and Windows 2003 Performance Monitor under the desktop Start menu:
    a. Right-click the status bar on your desktop. Click **Task Manager**.
    b. Select the **Processes** tab.
    c. Click **View** > **Select Columns**.
    d. Select **Thread Count**.
    e. Click **OK**.

    The **Processes** tab shows the number threads for each process under the column name **Threads**, including Apache.
2. Use the IBM HTTP Server server-status (this choice works on all platforms, not just Windows):
    a. Edit the IBM HTTP Server `httpd.conf` file as follows: Remove the comment character # from the following lines: `#LoadModule status_module`, `modules/ApacheModuleStatus.dll`, `#<Location/server-status>`, `#SetHandler server-status`, and `#</Location>`.
    b. Save the changes and restart the IBM HTTP Server.
    c. In a Web browser, go to the URL: `http://`*`yourhost`*`/server-status`. Alternatively,
    d. Click **Reload** to update status.
    e. (Optional) If the browser supports refresh, go to `http://`*`your_host`*`/server-status?refresh=5` to refresh every five seconds. You will see five requests currently processing 45 idle servers.

- **Default value:** 50 for IBM HTTP Server 1.3.28.
- **Recommended value:** Set this value to prevent bottlenecks, allowing just enough traffic through to the application server.

– **Web server configuration reload interval**
  - **Description:** Tracks a variety of configuration information about WebSphere Application Server resources. The Web server needs to understand some of this information, such as Uniform Resource Identifiers (URIs) pointing to WebSphere Application Server resources. This configuration data is pushed to the Web server through the WebSphere Application Server plug-in at intervals specified by this parameter. Periodic updates add new servlet definitions without having to restart any of the WebSphere Application Server servers. However, the dynamic regeneration of this configuration information is costly in terms of performance. Adjust this parameter in a stable production environment.
  - **How to view or set:** This parameter, <config RefreshInterval=xxx>, where *xxx* is the time interval in seconds, is specified in the `plugin-cfg.xml` file located in the %WAS_HOME%\config\cells directory.
  - **Default value:** The default reload interval is 60 seconds.
  - **Recommended value:** Increase the reload interval to a value that represents an acceptable wait time between the servlet update and the Web server update.

  For more information about the `plugin-cfg.xml` file see the topic plugin-cfg.xml file.

- **Sun ONE Web server, Enterprise Edition (formerly iPlanet) - Solaris operating environment**

  The default configuration of the Sun ONE Web server, Enterprise Edition provides a single-process, multi-threaded server.

  – **Active threads**

  - **Description:** Specifies the current number of threads active in the server. After the server reaches the limit set with this parameter, the server stops servicing new connections until it finishes old connections. If this setting is too low, the server can become throttled, resulting in degraded response times. To tell if the Web server is being throttled, consult its perfdump statistics. Look at the following data:
    - **WaitingThreads count:** If WaitingThreads count is getting close to zero, or is zero, the server is not accepting new connections.
    - **BusyThreads count:** If the WaitingThreads count is close to zero, or is zero, BusyThreads is probably very close to its limit.
    - **ActiveThreads count:** If ActiveThreads count is close to its limit, the server is probably limiting itself.
  - **How to view or set:** Use the Maximum number of simultaneous requests parameter in the Enterprise Server Manager interface to control the number of active threads within Sun ONE Web server, Enterprise Edition. This setting corresponds to the RqThrottle parameter in the `magnus.conf` file.
  - **Default value:** 512
  - **Recommended value:** Increase the thread count until the active threads parameters show optimum behavior.
- **Microsoft Internet Information Server (IIS) - Windows NT and Windows 2000**
  - **IIS permission properties**
    - **Description:** The Web server has several properties that dramatically affect the performance of the application server. The default settings are usually acceptable. However, because other products can change the default settings without user knowledge, make sure to check the IIS settings for the Home Directory permissions of the Web server. The permissions should be set to Script and not to Execute. If the permissions are set to Execute, no error messages are returned, but the performance of WebSphere Application Server is decreased.
    - **How to view or set:** To check or change these permissions, perform the following procedure in the Microsoft management console:
      1. Select the Web site (usually default Web site).
      2. Right-click and select the **Properties** option.
      3. Click the **Home Directory** tab. To set the permissions of the Home Directory:
         a. In the **Application** settings, select the **Script** check box in the **Permissions** list and clear the **Execute** check box.
         b. (Optional) Check the permissions of the sePlugin:
            1) Expand the Web server.
            2) Right-click the sePlugin and select **Properties**.
            3) Confirm that the **Execute** permissions are set to **Execute**.
    - **Default value:** Script
    - **Recommended value:** Script
  - **Number of expected hits per day**
    - **Description:** Controls the memory that IIS allocates for connections.
    - **How to view or set:** Using the performance window, set the parameter to More than 100000 in the Web site properties panel of the Microsoft management console.
    - **Default value:** Fewer than 100000
    - **Recommended value:** More than 100000
  - **ListenBackLog parameter**
    - **Description:** Alleviates failed connections under heavy load conditions, if you are using IIS on Windows NT and Windows 2000. Failure typically occurs when you are using more than 100 clients. ListenBackLog increases

the number of requests that IIS keeps in its queue. Consider raising this value if you see intermittent `Unable to locate server errors` in the Netscape browser.

- **How to view or set:**
  1. Use a command prompt to issue the **regedit** command to access the operating system registry.
  2. In the registry window, locate the parameter in the HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\InetInfo\Parameters\ListenBackLog directory.
  3. Right-click the parameter to adjust the setting according to the server load.
- **Default value:** 25 (decimal)
- **Recommended value:** You can set the ListenBackLog parameter can be set as high as 200, without negative impact on performance and with an improvement in load handling.

Modifying the WebSphere plug-in to improve performance

You can improve the performance of IBM HTTP Server 1.3 (with the WebSphere Web server plug-in) by modifying the plug-in's RetryInterval configuration. The RetryInterval is the length of time to wait before trying to connect to a server that has been marked temporarily unavailable. Making this change can help the IBM HTTP Server 1.3 to scale higher than 400 users.

The plug-in marks a server temporarily unavailable if the connection to the server fails. Although a default value is 60 seconds, it is recommended that you lower this value in order to increase throughput under heavy load conditions. Lowering the RetryInterval is important for IBM HTTP Server 1.3 on UNIX operating systems that have a single thread per process, or for IBM HTTP Server 2.0 if it is configured to have fewer than 10 threads per process.

How can lowering the RetryInterval affect throughput? If the plug-in attempts to connect to a particular application server while the application server threads are busy handling other connections, which happens under heavy load conditions, the connection times out and the plug-in marks the server temporarily unavailable. If the same plug-in process has other connections open to the same server and a response is received on one of these connections, the server is marked again. However, when you use the IBM HTTP Server 1.3 on a UNIX operating system, there is no other connection since there is only one thread and one concurrent request per plug-in process. Therefore, the plug-in waits for the RetryInterval before attempting to connect to the server again.

Since the application server is not really down, but is busy, requests are typically completed in a small amount of time. The application server threads become available to accept more connections. A large RetryInterval causes application servers that are marked temporarily unavailable, resulting in more consistent application server CPU utilization and a higher sustained throughput.

Note: Although lowering the RetryInterval can improve performance, if all the application servers are running, a low value can have an adverse affect when one of the application servers is down. In this case, each IBM HTTP Server 1.3 process attempts to connect and fail more frequently, resulting in increased latency and decreased overall throughput.

For more information about tuning heavy-loaded Web servers, see Performance: Resources for learning

Related tasks

Tuning performance parameter index

Related reference

Performance: Resources for learning

"plugin-cfg.xml file" on page 51

Manually editing the plug-in configuration

Regenerating the plug-in configuration file does not guarantee a correct configuration for advanced configuration scenarios. If your use of the WebSphere Application Server product requires a plug-in configuration more advanced than what is provided with the default plug-in configuration, or if the plug-in does not behave as properly after the configuration file is regenerated, you can manually edit the configuration file to obtain the proper plug-in behavior for your environment.

1. Find the plug-in configuration files. By default, the working or active versions of the `plugin-cfg.xml` file reside in the directory *install_root*/config/cells.
2. Manually edit the plug-in configuration files. A plugin-cfg.xml file is created using either the Update Web Server Plug-in Configuration page in the Application Server administrative console, or by running the GenPluginCfg.sh script. If the file is created in a distributed platform environment, it will be in ASCII format, which is the required format for a distributed platform Web server plug-in.

   To edit a plugin-cfg.xml file, open the file in a text editor, change the plug-in settings as needed, and save the file.

   CAUTION: If the plug-in configuration is regenerated, your manual edits to the plug-in configuration file will be overwritten. Therefore, you should maintain a record of any manual changes you make for future reference.

Situations requiring manual editing of the plug-in configuration

Some situations require you to edit the `plug-in-cfg.xml` file.

The following situations require manual editing of the `plugin-cfg.xml` file:

- If the Web server and `plugin-cfg.xml` file are installed on a separate remote system, you must change the paths in `plugin-cfg.xml` file if:
  - The plug-in was generated on a Windows 32 system platform. Copy it to a remote Linux or UNIX system with an HTTP Server and a WebSphere Application Server Version 5 plug-in.
  - The plug-in was generated on a Linux or UNIX system. Copy it to a remote Windows platform with an HTTP Server and a WebSphere Application Server Version 5 plug-in.
  - The plug-in was generated on a Linux or UNIX system and needs to be copied to another, remote Linux or UNIX system that has a different configuration. For example, the plug-in was generated on a system having a single-server (Base) or Network Deployment installation on AIX in the default path, and the remote HTTP Server and plug-in are installed on a Solaris or Linux system with the plug-in installed in the default location.
- In a Network Deployment environment, if the single-server (base) product, Network Deployment product, plug-in, and HTTP Server are all installed on the same node:
  - During installation of the base product, you must copy the `plugin-cfg.xml` file to the `*install_root*/AppServer/config/cells` directory because that is

where the plug-in and the HTTP Server will look for it. (The plugin-cfg.xml file is initially located in the `install_root/DeploymentManager/config/cells` directory.)
  - In the plugin-cfg.xml file, you must change references to the `install_root/DeploymentManager/config/cells` path to `install_root/AppServer/config/cells`. The references to the `install_root/DeploymentManager/config/cells` path exist because the deployment manager runs the plugin-update procedure.

  The `install_root/AppServer` path is the default for the base product. If the base product is installed in a different location, change the paths to refer to the actual location.
- If the single-server (base) or Network Deployment product, or the plug-in, are installed in a non-default location, you must change the paths in `plugin-cfg.xml`. The plug-in generator assumes that the plug-in path is the same as the base product path structure. For example, if you install the base product in c:\myApps\WebSphere\AppServer50 and install the plug-in on a remote Windows system in c:\WebSphere\AppServer50, you must generate the plug-in using the administrative console on the base product and then edit the plug-in at c:\myApps\WebSphere\AppServer50\config. You must change all of the c:\myApps\WebSphere\AppServer50\... path structures to c:\WebSphere\AppServer50\...
- When the deployment manager is installed on a machine that is remote from the base WebSphere Application Server installation, implement one of the following solutions to allow the `plugin-cfg.xml` file to retain the `install_root/AppServer` directory structures, and to not assume those of the `install_root/DeploymentManager`, after a regeneration of the plug-in and full synchronization. The `plugin-cfg.xml` file is located in the `install_root/AppServer/config` directory.
  - **Command line**:

    At a command prompt, change to the `/bin` directory of the installation root of the deployment manager machine. Type GenPluginCfg -destination.root<*install_root*> on the machine where the Network Deployment product is installed. This creates or updates the `plugin-cfg.xml` file. This changes all directory specifications in the `plugin-cfg.xml` file to the `install_root\AppServer` directories. For example, run the `GenPluginCfg -destination.root "E:\WebSphere\AppServer"` command from the `\bin` directory of the installation root of the Network Deployment node.
  - **plugin-cfg.xml file**:

    Edit the `plugin-cfg.xml` file in the `/config/cells` directory of the deployment manager, to point to the correct directory structure for the log file, keyring, and stashfile. Perform a full synchronization so the `plugin-cfg.xml` file is replicated in all the WebSphere Application Server nodes. The deployment manager `plugin-cfg.xml` file can point to Application Server directories without any conflict.
- When you intend to allow Web servers to access the administrative console.

Regenerating Web server plug-in configurations

At times, you might need to instruct the WebSphere Web server plug-in to regenerate its configuration. You should regenerate the plug-in configuration after, for example, installing or removing an enterprise application, adding or removing servlets and mappings from a particular application, or changing the configuration for the plug-in, a virtual host or a transport. Failure to regenerate the plug-in after introducing a new application likely results in a *404 File Not Found* error when a user tries to access the new Web application.

CAUTION: Regenerating the plug-in configuration can overwrite manual configuration changes that you might want to preserve. Before performing this task, understand its implications as described in Chapter 6, “Configuring Web server plug-ins,” on page 51.

To regenerate the plug-in configuration, you can either use the Update Web Server Plug-in Configuration page in the administrative console, or issue the following command:

```
install_root/bin/GenPluginCfg.sh|bat
```

Both methods for regenerating the plug-in configuration create a `plugin-cfg.xml` file in ASCII format, which is the proper format for execution in a distributed environment.

To use the Update Web Server Plug-in Configuration page in the administrative console:

1. Go to the Update Web Server Plug-in page. Click **Environment > Update Web Server Plug-in** in the console navigation tree.
2. Click **OK**.
3. You might need to stop the application server and then start the application server again before the changes to the plug-in configuration go into effect.

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 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.

- **Command line**:

  At a command prompt, change to the `DeploymentManager/bin` directory and type on the machine where the Deployment Manager is installed. This creates or updates the `plugin-cfg.xml` file, and changes all of the directories in the `plugin-cfg.xml` file to directories.

For example, issue the following command from the `DeploymentManager/bin` directory.

- **plugin-cfg.xml file**:

  Edit `plugin-cfg.xml` file, located in the DeploymentManager/config/cells directory, to point to the correct directory structure for the log file, keyring, and stashfile.

  Perform a full synchronization so the `plugin-cfg.xml` file is replicated in all the WebSphere Application Server nodes. You can use scripting to perform a node synchronization or use the administrative console.

  The Deployment Manager `plugin-cfg.xml` file can point to the application server directories without any conflict.

After using the administrative console to make configuration changes that involve the served paths of Web applications, manually trigger the regeneration of the plug-in configuration (or manually edit the file if that is what you have been doing). The plug-in configuration regenerates. The Web server is aware of can access the new Web application configuration. If a user requests a servlet using the path specified by the new Web resource, the request should be successful.

GenPluginCfg command reference

Purpose

This topic describes the command-line syntax for the GenPluginCfg command. This command is used to regenerate the WebSphere Web server plug-in configuration file, plugin-cfg.xml. This file is located in the bin directory of the product installation root, and has the following command-line syntax:

CAUTION: Regenerating the plug-in configuration can overwrite manual configuration changes that you might want to preserve. Before performing this task, understand its implications as described in Chapter 6, “Configuring Web server plug-ins,” on page 51.

To regenerate the plug-in configuration, you can either use the Update Web Server Plug-in Configuration page in the administrative console, or issue the following command:

```
install_root/bin/GenPluginCfg.sh|bat
```

Both methods for regenerating the plug-in configuration create a `plugin-cfg.xml` file in ASCII format, which is the proper format for execution in a distributed environment.

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
- WebSphere Application Server V4.0 and V4.0.1 for zOS and OS/390: Configuring Web Applications
- Best Practice: Configuring Web Applications on WebSphere Application Server for z/OS and OS/390

Programming instructions and examples
- IBM HTTP Server documentation
- WebSphere Application Server education
- Listing of all IBM WebSphere Application Server Redbooks
- Redbook on IBM WebSphere V4.0 Advanced Edition Scalability and Availability

Web server plug-in tuning tips

During normal operation, the backlog of connections pending to an application server are bound to grow. Therefore, balancing workloads amongst application servers in a network fronted by a WebSphere Web server plug-in helps improve request response time.

In a distributed environment, you can use the MaxConnections server attribute in the Web server plug-in configuration file (`plugin-cfg.xml`) to define the maximum number of connections that can be pending to any of the Application Servers in the cluster. 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.

In a z/OS environment, WebSphere Application Server V5 uses native Workload Management (WLM) functionality to dynamically balance the workload of the Application Servers within a cluster. The MaxConnections attribute is not the optimal choice for load balancing in a z/OS environment.

The capacity of the Application Servers in the network determines the value you specify for the MaxConnections attribute in the`plugin-cfg.xml` file. The ideal scenario is for all of the Application Servers in the network to be optimally utilized. For example, if: you have the following environment:
- There are 10 WebSphere Application Server nodes in a cluster.
- All of these nodes host the same applications (that is, Application_1 and Application_2).
- This cluster of nodes is fronted by five IBM HTTP Servers.
- The IBM HTTP Servers get requests through a load balancer.
- Application_1 takes approximately 60 seconds to respond to a request
- Application_2 takes approximately 1 second to respond to a request.

Depending on the request arrival pattern, all requests to Application_1 might be forwarded to two of the nodes, say node_1 and node_2. If the arrival rate is faster than the processing rate, the number of pending requests to node_1 and node_2 can grow.

Eventually, node_1 and node_2 are busy and are not able to respond to future requests. It usually takes a long time to recover from this overloaded situation.

If you want to maintain 2500 connections, and optimally utilize the Application Servers in this example, set the MaxConnections attribute in the `plugin-cfg.xml` file to 50. (This value is arrived at by dividing the number of connections by the result of multiplying the number of Application Servers by the number of Web servers; in this example, 2500/(10x5)=50.)

The MaxConnections attribute works best with Web servers that follow the threading model instead of the process model, and only one process is started.

The IBM HTTP Server V1.3.x follows the process model. With the process model, a new process gets created to handle each connection from the Application Server, and typically, one process handles only one connection to the Application Server. Therefore, the MaxConnections attribute does not have much of an impact in restricting the number of concurrent requests to the Application Server.

The IBM HTTP Server V2.0.x 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
```

Tuning Web servers

Web server tuning parameters lists tuning parameters specific to Web servers. The listed parameters may not apply to all of the supported Web servers. Check your Web server documentation before using any of these parameters.

Chapter 7. Welcome to Cell-wide settings

The configuration data for Version 5.0 of WebSphere Application Server is stored in files, and those files exist in one of several directories in the configuration repository tree. The directory in which a configuration file exists determines its scope, or how broadly or narrowly that data applies. Files in an individual server directory apply to that specific server only. Files in a node level directory apply to every server on that node. And files in the cell directory apply to every server on every node within the entire cell.

Cell-wide settings are configuration data that is stored in files in the cell directory and those files are replicated to every node in the cell. There are several different configuration settings that apply to the entire cell. These include the definition of virtual hosts, shared libraries, and any variables that you want to be consistent throughout the entire cell.

Chapter 8. Configuring the cell-wide environment

To assist in handling requests among Web applications, Web containers, and application servers, you can configure cell-wide settings for virtual hosts, variables and shared libraries.

1. Configure virtual hosts.
2. Configure variables.
3. If your deployed applications will use shared library files, define the shared library files needed.

Virtual hosts

A virtual host is a configuration enabling a single host machine to resemble multiple host machines. 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 used to request the servlet, for example yourHostName:80. When no port number is specified, 80 is assumed.

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 and serve the servlet. If no match is found, an error is returned to the browser.

An application server provides a default virtual host with some common aliases, such as the machine's IP address, short host name, and fully qualified host name. The alias comprises the first part of the path for accessing a resource such as a servlet. For example, it is localhost:80 in the request http://localhost:80/myServlet.

A virtual host is not associated with a particular node (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.

Why and when to use virtual hosting

Virtual hosts allow the administrator to isolate, and independently manage, multiple sets of resources on the same physical machine.

Suppose an Internet Service Provider (ISP) has two customers whose Internet sites it would like to host on the same machine. The ISP would like to keep the two sites isolated from one another, despite their sharing a machine. The ISP could associate the resources of the first company with VirtualHost1 and the resources of the second company with VirtualHost2.

Now suppose both company's sites offer the same servlet. Each site has its own instances of the servlet, which are unaware of the other site's instances. 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 will not be routed there.

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, even though VirtualHost2 is refusing to fill requests for the same servlet.

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 node (physical machine), you probably do not need any virtual hosts in addition to the default host.

How requests map to virtual host aliases

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 case insensitive, 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 due to 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 hostname 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 used to issue the request. A message states that the virtual host could not be found.

Configuring virtual hosts

Virtual hosts enable you to isolate, and independently manage, multiple sets of resources on the same physical machine.

1. Create a virtual host using the Virtual Hosts page of the administrative console. Click **Environment > Virtual Hosts** from the navigation tree of the console,

click **New** and, on the settings page for a virtual host that displays, specify an administrative name for the virtual host. When you create a virtual host, a default set of 90 MIME entries is created for the virtual host.

2. Determine whether you need a virtual host alias for each HTTP transport port. There must be a virtual host alias corresponding to each port used by an HTTP transport. There is one HTTP transport in each Web container, with one Web container in each application server.

   You must create a virtual host for each HTTP port in the following cases:
   - You are using the internal HTTP transport with a port other than the default of 9080, or for some reason the virtual host does not contain the usual entry for port 9080.
   - You have created multiple application servers (either stand-alone or in a cluster) that are using the same virtual host. Because each server must be listening on a different HTTP transport port, you need a virtual host alias for each one's transport port.

   If you determine that you need one or more virtual host aliases, on the HTTP Transports page, note the **Port** values, such as 9080 or 9082.
3. If necessary, create a virtual host alias for each HTTP transport port. From the Virtual Hosts page, click on your virtual host and, on the settings page for a virtual host, click **Host Aliases**. For each virtual host alias that you need, on the Host Aliases page, click **New**; then, on the settings page for a virtual host alias, specify a host name and port. Configure the virtual host to contain an alias for the port number. For example, specify an alias of `*:9082` if 9082 is the port number in use by the transport.
4. When you enter the URL for the application into a Web browser, include the port number in the URL. For example, if 9082 is the port number, specify a URL such as `http://localhost:9082/wlm/SimpleServlet`
5. If MIME entries are not specified at the Web module level, define MIME object types and their file name extensions. For each needed MIME entry, on the MIME Types page, click **New**; then, on the settings page for a MIME type, specify a MIME type and extension.
6. After you configure a virtual host alias or change a configuration, you must regenerate the Web server plug-in configuration and restart WebSphere Application Server.

Virtual host collection

Use this page to manage virtual hosts.

To view this administrative console page, click **Environment > Virtual Hosts**.

Name

Specifies a logical name for configuring Web applications to a particular host name. The default virtual host is suitable for most simple configurations.
A virtual host configuration lets a single host machine resemble multiple host machines. Resources associated with one virtual host cannot share data with resources associated with another virtual host, even when virtual hosts share the same machine.

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, JSP, or HTML page). For example, the host alias name is `myhost` in a DNS name of `myhost:8080`.

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, 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 host alias name is `myhost`, the DNS name is `myhost:8080`, where *8080* is the port. A URL refers to this DNS as: http://*myhost:8080*/servlet/snoop.

For existing instances, the default reflects the value specified at product setup. For new instances, the default can be `*` to allow any value or no specification.

Data type	String
Default	*

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`, `9080` or a similar value.

Data type	Integer
Default	80

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 a virtual host. Virtual host MIME entries apply when you do not specify MIME entries at the Web module level.

To view this administrative console page, click **Environment > Virtual Hosts >***virtual_host_name* **> MIME Types**.

MIME Type:

Specifies a MIME type, which can be text, image, or application. An example value for MIME type is `text/html`.

Extensions:

Lists file extensions of files that map the MIME type. Example extensions for a `text/html` MIME type include `.htm`, `.html`, and `.txt`.

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 text, image, or application.

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:

Lists file extensions of files that map the MIME type.

Example file extensions for a `text/html` MIME type include `.htm`, `.html`, and `.txt`. A default value appears only if you are viewing the configuration for an existing instance.

Data type	String

Variables

A variable is a configuration property that can be used to provide a parameter for any value in the system. A variable has a name and a value to be used in place of that name wherever the variable name is located within the configuration files.

Variables have a scope, which is the range of locations in the WebSphere Application Server network where the variable is applicable. A variable with a cell-wide scope applies across the entire WebSphere Application Server cell. A variable with node-level scope applies only on the node for which it is defined. If a node-level variable has the same name as a cell-wide variable, the node-level variable value takes precedence. A server variable only applies to the one server process, and takes precedence over any wider scoped variable with the same name.

When you use variables in configuration values such as file system path settings, the following syntax refers to the variable, using the variable's name:

`${`*`variable_name`*`}`

If the value of a variable contains a reference to another variable, the value of the variable is computed by substituting the value of the referenced variable recursively. For example:

Variable name	**Variable value**
ROOT_DIR	`/`
HOME_DIR	`${ROOT_DIR}home`
USER_DIR	`${HOME_DIR}/myuserdir`

In this example, the variable reference `${USER_DIR}` resolves to the value `/home/myuserdir`.

Variables are often useful where configured paths cannot contain *and*.

Configuring WebSphere variables

You can define a WebSphere Application Server variable to provide a parameter for a value in the system. After you define the name and value for a variable, the value is used in place of that name wherever the variable name is located within the configuration files. The scope of a variable can be cell-wide, node-wide, or applicable to only one server process.

1. Click **Environment > Manage WebSphere Variables** in the console navigation tree. On the WebSphere Variables page, click **New**.
2. Specify the scope of the new variable. Indicate if new variable should be for the **Cell**, **Node**, or **Server** and click **Apply**. The scope control enables you to choose the variable level in which you wish to work. Choosing to apply to a cell, node or server allows you to put the variable on all servers at that particular level. If you specify the same variable on a cell or node as on a server, the server level variable overrides the cell or node variable. Similarly, a variable at the node level overrides the instance specified at the cell level.
3. On the Variable page, specify a name and value for the variable. So other people can understand what the variable is used for, also specify a description for the variable. Then click **OK**.
   - Use cell-wide customization values.
   - You can use WebSphere variables to modify the daemon configuration. By appending a server custom property onto a daemon tag, you can designate

that variable specifically for that daemon. Enter `DAEMON_<server custom property>` in the **Name** field. For example, if you enter `DAEMON_ras_trace_outputlocation` in the Name field and `SYSOUT` in the Value field, you can direct that particular daemon's trace output to SYSPRINT.

- WebSphere variables support substitution. The name of a variable can be formed by substituting the value of another variable. If you enter `${<variable name>}` in the **Name** field, the value of `<variable name>` will be the name of your new WebSphere variable. For example `${JAVA_HOME}` will create a WebSphere variable with a name that is equal to the Java home directory.
- There are also WebSphere internal variables. The application server uses these variables for its own purposes. The prefixes that indicate that a variable is WebSphere internal are `WAS_DAEMON_<server custom property>`, `WAS_DAEMON_ONLY_<server custom property>`, and `WAS_SERVER_ONLY_<server custom property>`. Any variables with these tags are not intended for your use. They are reserved exclusively for use by the server runtime. Modifying these variables may cause unexpected errors.

4. Verify that the variable is shown in the list of variables.
5. Save your configuration.
6. To have the configuration take effect, stop the server and then start the server again.

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 > Manage WebSphere Variables**.

For information on a variable, click the variable and read the value in the **Description** field.

Name

Specifies a symbolic name that represents a file path, Web address, or other value.

Value

Specifies the value that the symbolic name represents, such as an absolute file path.

Scope

Specifies whether the symbolic name applies across a cell, node, or server.

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 > Manage WebSphere Variables >***WebSphere_variable_name*.

Name:

Specifies a symbolic name that represents a file path, Web address, or other value.

WebSphere Application Server substitutes the symbolic name wherever its value appears in the system.

For example, ″CONFIG_ROOT″ is the symbolic name representing the configuration directory path ″C:\WebSphere\AppServer\Config″ for the base

WebSphere Application Server product on a Windows system.

Data type	String

Value:

Specifies the value that the symbolic name represents, such as absolute file path.

For example, the value might be the absolute file path, ″C:\WebSphere\AppServer\Config,″ in the base WebSphere Application Server product on a Windows system. The corresponding symbolic name might be ″CONFIG_ROOT.″

Data type	String

Description:

Documents the purpose of a variable.

Data type	String

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 400** 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 400 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/iseries/v5r1/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 2.0. 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 5.0 on platforms other than iSeries, use the **JTOpen** version of the Toolbox JDBC driver.

Configure and use 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 work station such as *C:\JDBC_Drivers\Toolbox*.
2. Open the administrative console.
3. Select **Environment**.
4. Select **Managed WebSphere Variables**.
5. Set the managed variable *OS400_TOOLBOX_JDBC_DRIVER_PATH* at the **Node** level.
6. Double click **OS400_TOOLBOX_JDBC_DRIVER_PATH**.
7. 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 == "C:\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 classpath, 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

If your deployed applications use shared library files, set variables for the library files and associate the files with specific applications or with an Application Server, which associates the files with all applications on the server. Use the Shared Libraries page to define new shared library files to the system and remove them.

1. Identify library files and their classpaths.
   a. Click **Environment > Shared Libraries** in the console navigation tree to access the Shared Libraries page.
   b. Change the scope of the collection table to see what shared libraries are in a cell, node, or server. Select the cell, a node, or a server and click **Apply**.

c. Click **New**.

d. On the settings page for a shared library, specify the name, classpath, and any other variables for the library file that are needed.

e. Click **Apply**.

Repeat this step until you define a shared library instance for each library file that your applications need.

2. Associate shared library files with an application that must use one or more shared libraries.

   Use this step to associate a file with an application or perform the next step to associate a file with an Application Server, which associates the file with every application on the server.

   a. Click **Applications > Enterprise Applications** in the console navigation tree.
   b. Click on the installed application that uses the shared libraries.
   c. Click **Libraries** to access the Library Ref page.
   d. Click **Add**.
   e. On the settings page for a library reference, specify variables for the library reference as needed.
   f. Click **Apply**.

   Repeat this step until you define a library reference instance for each library file that your application requires.

3. Click **Servers > Application Servers >***server_name* to associate a shared library with an Application Server for the run-time environment.

   Use this step to associate a file with an Application Server, which associates the file with every application on the server, or perform the previous step to associate a file with an application.

   a. Set the application class-loader (sometimes referred to as classloader) policy and application class-loader mode on the settings page for an application server as described in *IBM WebSphere Application Server Network Deployment, Version 5.1: Applications*.
   b. Click **Libraries** on the settings page for a class loader.
   c. Click **Add** from the Library Ref page.
   d. Specify variables for the library reference as needed on the settings page for a library reference and click **OK**.

   Repeat this step until you define an Application Server for each library file that your application needs.

4. Remove a library file from the collection of shared library files by placing a checkmark beside the library you want removed on the Shared Libraries page and clicking **Delete**.

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**.

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.

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*.

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.

Data type	String
Units	Class path

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 **Applications > Enterprise Applications >***application_name* **> Libraries**.

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 **Applications > Enterprise Applications >***application_name* **> Libraries >***library_reference_name*. A shared library must be defined to view this page.

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 cell-wide 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

Related tasks

Chapter 8, “Configuring the cell-wide environment,” on page 77

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 Licensing
IBM Corporation
500 Columbus Avenue
Thornwood, New York 10594 USA

Trademarks and service marks

The following terms are trademarks of IBM Corporation in the United States, other countries, or both:

- AIX
- CICS
- Cloudscape
- DB2
- DFSMS
- Everyplace
- iSeries
- IBM
- IMS
- Informix
- iSeries
- Language Environment
- MQSeries
- MVS
- OS/390
- RACF
- Redbooks
- RMF
- SecureWay
- SupportPac
- ViaVoice
- VisualAge
- VTAM
- WebSphere
- z/OS
- zSeries

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, and service names may be trademarks or service marks of others.