WebSphere® Business Integration Server Foundation, Version 5.1

Server and environment

Note
Before using this information, be sure to read the general information under "Notices" on page 135.

Compilation date: March 25, 2004

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

Application servers provide the core functionality of the WebSphere Application Server product family. They extend the ability of a Web server to handle Web application requests, and much more. An application server enables a server to generate a dynamic, customized response to a client request.

For additional overview, refer to Chapter 2, "Welcome to Application Servers," on page 3.

Clusters

Workload management optimizes the distribution of client processing tasks. Incoming work requests are distributed to the application servers that can most effectively process the requests. Workload management also provides failover when servers are not available, improving application availability.

Clusters are sets of application servers that are managed together and participate in workload management. The servers that are members of a cluster can be on different host machines, as opposed to the servers that are part of the same node and must be located on the same host machine.

For additional overview, refer to Chapter 5, "Welcome to Clusters," on page 69.

Chapter 2. Welcome to Application Servers

Overview

An application server is a Java Virtual Machine (JVM) running user applications. The application server collaborates with the Web server to return a dynamic, customized response to a client's request. Application code, including servlets, JSPs, EJBs and their supporting classes, run in an application server. In keeping with the Java 2 platform, Enterprise Edition (J2EE) component architecture, servlets and JSPs run in a Web container, and EJBs run in an EJB container.

To begin creating and managing an application server, seeChapter 3, "Configuring application servers," on page 5.

You can define multiple application servers, each running its own JVM. Enhance the operation of an application server by using the following options:
- "Configuring transports" on page 22
- "Custom services" on page 27
- Command-line information that passes to a server when it starts or initializes
- "Tuning application servers" on page 48
- Chapter 4, "Managing Object Request Brokers," on page 51

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 Network Deployment and Enterprise Extensions, the JMS functions (of JMS providers) within the administration domain are served by one or more JMS servers. 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.

For more information, refer to Using JMS and messaging in applications.

Chapter 3. Configuring application servers

An application Sserver 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
   - 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 43″ 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. On the Network Deployment product, you can also create a new application server when you add a cluster member to a server cluster.

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.

   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. Start your application server.
5. Monitor the running of application servers.
6. Stop your application server.
7. 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**.

Name
Specifies a logical name for the server. For WebSphere Application Server, server names must be unique within a node. For WebSphere Application Server for z/OS, this is sometimes called the long name. Server names must be unique within a node. If you have multiple nodes in a cluster, the server names must also be unique within the cluster. You cannot use the same server name within two nodes that are part of the same cluster. WebSphere uses the server name for administrative actions, such as referencing the server in scripting.

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. For WebSphere Application Server, server names must be unique within a node. For WebSphere Application Server for z/OS, this is sometimes called the long name. Server names must be unique within a node. If you have multiple nodes in a cluster, the server names must also be unique within the cluster. You cannot use the same server name within two nodes that are part of the same cluster. WebSphere uses the server name for administrative actions, such as referencing the server in scripting.

Data type	String
Default	server1

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

Data type	String
Default	Started

Note: The default setting for **Initial State** is *Started*. It is important to note that if you change the **Initial State** setting for this subcomponent to *Stopped*, and then start its associated server, this subcomponent will remain in the *Stopped*

state. For example, if you set the **Initial State** property of the Application Server subcomponent to *Stopped*, and then start its associated server, only the server and other services such as administration, naming, and security will start. The application services, such as the Web container, EJB container, and applications, will not start.

If you are using clusters, note also that the **Initial State** property is not intended to be used to control the state of individual servers in the cluster at the time the cluster is started. It is intended only as a way to control the state of the Application Server subcomponent of a server. It is best to start and stop the individual servers of a cluster using the Server options of the Administrative Console or command line commands (**startServer** and **stopServer**).

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. For WebSphere Application Server for z/OS, the server short name must be unique within a cell. The short name identifies the server to the native facilities of the operating system, such as Workload Manager (WLM), Automatic Restart Manager, SAF (for example, RACF), started task control, and others.

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 one of the following paths:
- **Servers > Application Servers >***server_name* **> End Points >***end_point_name*
- **Servers > JMS Servers >***server_name* **> Security Port Endpoint**
- **Servers > JMS 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

The following end points apply to WebSphere Application Server for z/OS only:

End point	Description
JMSSERVER Queued Address	Specifies the host and port number used to configure the WebSphere JMS Provider topic connection factory. Since the queue manager and queue broker for the WebSphere JMS Provider are configured outside the administrative console, changes to this endpoint require corresponding configuration changes to the queue manager and queue broker. The default port number is 5558.
JMSSERVER Direct Address	Specifies the host and port number used to configure the WebSphere JMS Provider topic connection factory. Since the queue manager and queue broker for the WebSphere JMS Provider are configured outside the administrative console, changes to this endpoint require corresponding configuration changes to the queue manager and queue broker. The default port number is 5559.

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

Range	1-65536

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 one of the following paths:
- **Servers > Application Servers >***server_name* **> Custom Properties >***property_name*
- **Servers** > **JMS 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.

Starting servers

This topic describes how to start application servers.

The node agent for the node on which the application server resides must be running before you can start the application server.

If you are running multiple servers, be sure the debug port is properly set. If multiple servers on the same node have the same debug port set, the servers could fail to start. See "Java virtual machine settings" on page 37 for more information on how to change the debug port.

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

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.

Note: After you start an application server, other processes might not discover the running application server immediately. Application servers are discovered by the node agent. Node agents are discovered by the deployment manager. Node agents usually can discover local application servers quickly but it can take a deployment manager from 2 to 60 seconds to discover a node agent.

Note: If you are using clusters, note that the **Initial State** property of the Application Server subcomponent (**Servers** > **Application Server** > *server_name* > **Server Components** > **Application Server**) is not intended to be used to control the state of individual servers in the cluster at the time the cluster is started. It is intended only as a way to control the state of the Application Server subcomponent of a server. It is best to start and stop the individual servers of a cluster using the Server options of the Administrative Console or command line commands (**startServer** and **stopServer**).

There are several options for starting an application server:

- Use the **startServer** command to start an application server from the command line.

  If the node agent for the node on which the application server resides is not running, run the **startNode** command and then run the **startServer** command.
- Start an application server using the administrative console.

1. 1. Click **Servers > Application Servers** from the administrative console navigation tree to go to the Application Server page.
2. 2. If the node agent for the node on which the application server resides is not running, click **Restart** or **Restart all Servers on Node** on the Node Agents page to start the node agent.

   If the node agent does not start, run the **startNode** command and then run the **startServer** command. Once a node agent completely stops running and remains stopped, you cannot remotely start the node agent from the Node Agents page. You must run the **startNode** command to start the node agent on the node where it runs.
3. 4. Select the check box beside the application server, then click **Start**.
4. 5. View the **Status** value and any messages or logs to see whether the application server starts.

- 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 an Application Server from a non-root user and the nodeagent from root

By default, each base WebSphere Application Server node on a Linux and UNIX platform uses the root user to run Application Servers. However, you can use a non-root user to run Application Servers. This task describes how to configure an Application Server to run as non-root while letting the nodeagent process and the jmsserver process run as root.

If global security is enabled, it is not recommended that the Local OS be used for user registry. In general, using the Local OS user registry requires that all processes run as root. Refer to Local operating system user registries for details.

Using a non-root user ID to run Application Servers can be done by setting all the Application Servers to run under the same operating system group. If you are running the JMS provider that WebSphere Application Server provides, add the `jmsserver` server to the `mqm` group to allow `jmsserver` to start the message queue. If you are not running `jmsserver`, you can use a group other than `mqm` in the following steps.

1. Log on to the Application Server system as root.
2. Create the `was1` user that you can use to run the Application Server.
3. Add users `root` and `was1` to the `mqm` group.
4. Log off and back on.
5. Log on to the Network Deployment system as root.
6. If it is not started, start the deployment manager process with the `startManager.sh` script from the `/bin` directory of the installation root:

   ```
   startManager.sh
   ```

7. Configure Application Server properties for the `root` and `was1` users. Use the administrative console on the deployment manager to complete the following steps:
   a. Define the nodeagent to run as a root process. Click **System Administration> Node Agents > nodeagent** *(for the node)***> Process Definition > Process Execution** and change these values:

Property	Value
Run As User	root
Run As Group	mqm
UMASK	002

   b. Define each Application Server to run as a `was1` process. Substitute the name of each server for `server1`. Click **Servers > Application Servers > server1 > Process Definition > Process Execution** and change these values:

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

   c. If running the JMS provider that WebSphere Application Server provides, define the `jmsserver` process to run as a `root` process. Click **Servers > JMS Servers > jmsserver** *(for the node)***> Process Definition > Process Execution** and change these values:

Property	Value
Run As User	root
Run As Group	mqm
UMASK	002

   d. Save and synchronize all nodes.
8. Log on to the Application Server system as root.
9. Ensure that all servers on the Application Server system are stopped, including the server1 and jmsserver processes. Use the `stopServer.sh` script from the `/bin` directory of the installation root:

```
stopServer.sh server1
stopServer.sh jmsserver
```

10. Ensure that the nodeagent process is stopped. Use the `stopNode.sh` script from the `/bin` directory of the installation root:

```
stopNode.sh
```

11. 5.1 + As root, use operating system tools to change file permissions in the following subdirectories under $WBI_HOME:

```
chgrp -R wasgroup $WBI_HOME/BRBeans
chmod -R g+w $WBI_HOME/BRBeans
chgrp -R wasgroup $WBI_HOME/ProcessChoreographer
chmod -R g+w $WBI_HOME/ProcessChoreographer
chgrp -R wasgroup $WBI_HOME/Scheduler
chmod -R g+w $WBI_HOME/Scheduler
chgrp -R wasgroup $WBI_HOME/WSGW
chmod -R g+w $WBI_HOME/WSGW
```

```
chgrp -R wasgroup $WBI_HOME/eclipse
chmod -R g+w $WBI_HOME/eclipse
chgrp -R wasgroup $WBI_HOME/recoveryLogs
chmod -R g+w $WBI_HOME/recoveryLogs
```

Note: You might have to change the permissions on additional $WBI_HOME subdirectories if your Integration Server application encounters permission problems during execution.

12. Start the nodeagent process from root. Use the `startNode.sh` script from the `/bin` directory of the installation root:

    ```
    startNode.sh
    ```

13. Start the jmsserver process from root. Use the `startServer.sh` script from the `/bin` directory of the installation root:

    ```
    startServer.sh jmsserver
    ```

14. Log on to the Application Server system as the `was1` user.
15. Start all Application Servers from the `was1` user. Use the `startServer.sh` script from the `/bin` directory of the installation root:

    ```
    startServer.sh server1
    ```

16. If you are running the JMS provider that WebSphere Application Server provides, verify that the queue manager is running: Run the `dspmq.sh` script from the `/bin` directory of the installation root:

    ```
    dspmq.sh
    ```

    The name of the queue is `WAS_wasnode_jmsserver`.

You can start an Application Server from a non-root user and run the nodeagent and jmsserver as root.

Running an Application Server and nodeagent from a non-root user

By default, each base Application Server node on a Linux, UNIX, or z/OS platform uses the root user ID to run the nodeagent process, the jmsserver process, and all Application Server processes. However, you can run the nodeagent, the jmsserver, and all Application Server processes under the same non-root user and user group. If you do run the nodeagent process with a non-root user ID, you must run the jmsserver and all Application Server processes that the node agent controls, under the same non-root user ID.

If global security is enabled, the user registry must not be Local OS. Using the Local OS user registry requires the nodeagent to run as root. Refer to Local operating system user registries for details.

Using the same non-root user and user group gives the nodeagent process the operating system permissions to start all other server processes. If using the JMS provider that WebSphere Application Server provides, the user group must be `mqm` for the jmsserver to start the message queue. If you are not using the JMS provider that WebSphere Application Server provides, you can specify a user group other than `mqm`.

For the steps that follow, assume that:
- `wasadmin` is the user to run all servers
- `wasnode` is the node name
- `wascell` is the cell name
- `mqm` and `mqbrkrs` are user groups associated with the JMS provider that WebSphere Application Server provides

- server1 is the Application Server
- /opt/WebSphere/Appserver is the installation root for the base node
- jmsserver exists because you are using the JMS provider that WebSphere Application Server provides

To configure a user ID to run the nodeagent and all server processes, complete the following steps.

1. Log on to the Application Server system as root.
2. Create user wasadmin with primary group mqm. Also add user wasadmin to group mqbrkrs if you are running the JMS provider that WebSphere Application Server provides.
3. Log off and back on.
4. Log on to the Network Deployment system as root.
5. If the deployment manager process is not started, start it with the startManager.sh script from the /bin directory of the installation root:

   ```
   startManager.sh
   ```
6. Start the administrative console.
7. Define the nodeagent to run as a wasadmin process using the administrative console of the deployment manager. Click **System Administration> Node Agents > nodeagent** *(for the node)***> Process Definition > Process Execution** and change these values:

Property	Value
Run As User	wasadmin
Run As Group	mqm
UMASK	002

8. Define each Application Server to run as a wasadmin process. Substitute the name of each server for server1. Click **Servers > Application Servers > server1 > Process Definition > Process Execution** and change these values:

Property	Value
Run As User	wasadmin
Run As Group	mqm
UMASK	002

9. If you are running the JMS provider that WebSphere Application Server provides, define the jmsserver process to run as a wasadmin process. Click **JMS Servers > jmsserver** *(for the node)***> Process Definition > Process Execution** and change these values:

Property	Value
Run As User	wasadmin
Run As Group	mqm
UMASK	002

10. Save and synchronize all nodes.
11. Log on to the Application Server system as root.
12. Ensure that all servers are stopped, including the server1 and jmsserver processes. Use the stopServer.sh script from the /bin directory of the installation root:

```
stopServer.sh server1
stopServer.sh jmsserver
```

13. Ensure that the node agent is stopped. Use the `stopNode.sh` script from the `/bin` directory of the installation root:

    ```
    stopNode.sh
    ```

14. 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:

    ```
    deletemq.sh wascell wasnode jmsserver
    ```

15. 5.1 + Change file permissions as the root user. Use operating system tools to change file permissions in the following subdirectories under $WBI_HOME:

    ```
    chgrp -R wasgroup $WBI_HOME/BRBeans
    chmod -R g+w $WBI_HOME/BRBeans
    chgrp -R wasgroup $WBI_HOME/ProcessChoreographer
    chmod -R g+w $WBI_HOME/ProcessChoreographer
    chgrp -R wasgroup $WBI_HOME/Scheduler
    chmod -R g+w $WBI_HOME/Scheduler
    chgrp -R wasgroup $WBI_HOME/WSGW
    chmod -R g+w $WBI_HOME/WSGW
    chgrp -R wasgroup $WBI_HOME/eclipse
    chmod -R g+w $WBI_HOME/eclipse
    chgrp -R wasgroup $WBI_HOME/recoveryLogs
    chmod -R g+w $WBI_HOME/recoveryLogs
    ```

    Note: You might have to change the permissions on additional $WBI_HOME subdirectories if your Integration Server application encounters permission problems during execution.

16. Log in as `wasadmin` on the application server system.
17. From `wasadmin`, run the `startServer.sh` script from the `/bin` directory of the installation root to start the JMS server and all Application Servers:

    ```
    startServer.sh jmsserver
    startServer.sh server1
    ```

18. If you are running the JMS provider that WebSphere Application Server provides, create the queue manager and broker for the Application Server. Run the `createmq.sh` script as `wasadmin` from the `/bin` directory of the installation root:

    ```
    createmq.sh /opt/WebSphere/AppServer wascell wasnode jmsserver
    ```

19. If you are running the JMS provider that WebSphere Application Server provides, verify that the MQ queue is running. As user `wasadmin`, run the `dspmq.sh` script from the `/bin` directory of the installation root:

    ```
    dspmq.sh
    ```

    The name of the queue is `WAS_wasnode_jmsserver`.

You can start an Application Server, the jmsserver, and the nodeagent 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. If messages indicate that a server has stopped running, use the Application Servers page to try restarting the server. If a cluster of servers has stopped running, use the Server Cluster page to try restarting the cluster. If the status of an application server is *Unavailable*, the node agent is not running in that node and you must restart the node agent before you can start the server.
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.

There are two options for stopping an application server:

- Use the **stopServer** command to stop an application server from the command line.
- Use the administrative console to stop an application server:
  1. 1. Click **Servers > Application Servers** from the administrative console navigation tree to go to the Application Server page.
  2. 2. Place a checkmark in the check box beside the application server that you want stopped and click **Stop**.
  3. 3. Confirm that you want to stop the application server.
  4. 4. View the **Status** value and any messages or logs to see whether the application server stops.

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 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 ″Manually configuring supported Web servers″ in the information center for the base product 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. On the settings page, enter the property you want to configure in the **Name** field and the value you want to set it to in the **Value** field.
5. Click **Apply** or **OK**.
6. Click **Save** on the console taskbar to save your configuration changes.
7. Restart the server.

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

ConnectionIOTimeOut
Use the `ConnectionIOTimeOut` property to specify the maximum number of seconds to wait when trying to read or process data during a request.seconds.

Data type	Integer
Default	5 seconds

ConnectionKeepAliveTimeout
Use the `ConnectionKeepAliveTimeout` property to specify the maximum number of seconds to wait for the next request on a keep alive connection.

Data type	Integer

MaxConnectBacklog
Use the `MaxConnectBacklog` property to specify the maximum number of outstanding connect requests that the operating system will buffer while it waits

for the application server to accept the connections. If a client attempts to connect when this operating system buffer is full, the connect request will be rejected.

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

Use the `MaxKeepAliveConnections` property to specify the maximum number of concurrent keep alive (persistent) connections across all HTTP transports. To make a particular transport close connections after a request, you can set MaxKeepAliveConnections to `0` (zero) or you can set KeepAliveEnabled to `false` on that transport.

The Web server plug-in keeps connections open to the application server as long as it can. However, if the value of this property is too small, performance is negatively impacted because the plug-in has to open a new connection for each request instead of sending multiple requests through one connection. The application server might not accept a new connection under a heavy load if there are too many sockets in TIME_WAIT state. If all client requests are going through the Web server plug-in and there are many TIME_WAIT state sockets for port 9080, the application server is closing connections prematurely, which decreases performance. The application server closes the connection from the plug-in, or from any client, for any of the following reasons:

- The client request was an HTTP 1.0 request when the Web server plug-in always sends HTTP 1.1 requests.
- The maximum number of concurrent keep-alives was reached. A keep-alive must be obtained only once for the life of a connection, that is, after the first request is completed, but before the second request can be read.
- The maximum number of requests for a connection was reached, preventing denial of service attacks in which a client tries to hold on to a keep-alive connection forever.
- A time out occurred while waiting to read the next request or to read the remainder of the current request.

Data type	Integer
Default	90% of the maximum number of threads in the Web container thread pool. This prevents all of the threads from being held by keep alive connections so that there are threads available to handle new incoming connect requests.

MaxKeepAliveRequests

Use the `MaxKeepAliveRequests` property to specify the maximum number of requests which can be processed on a single keep alive connection. This parameter can help prevent denial of service attacks when a client tries to hold on to a keep-alive connection. The Web server plug-in keeps connections open to the application server as long as it can, providing optimum performance.

Data type	Integer
Default	100 requests

KeepAliveEnabled

Use the `KeepAliveEnabled` property to specify 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. 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 service class requires a configuration file, specify the fully-qualified path name to that configuration file in the **externalConfigURL** field. This file name is passed into your custom service class.

   To invoke a native library from the custom service, provide the path name in the **Classpath** field in addition to the path names that are used to locate the classes and JAR files for the custom service. Doing this adds the path name to the WAS extension classloader, which allows the custom service to locate and correctly load the native library.
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 or nodeagent 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. For WebSphere Application Server, this page provides command-line information for starting or initializing a process. For WebSphere Application Server for z/OS, this page provides command-line information for starting, initializing, or stopping a process.

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

Start Command (startCommand)

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 (startCommandArgs)

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 (stopCommand)

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 (stopCommandArgs)

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 (terminateCommand)

Specifies the platform-specific command to terminate the server process

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

Terminate Command Args (terminateCommandArgs)
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 >***process type* **> 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.

Data type	String
Default	STOPPED
Range	Valid values are STOPPED, RUNNING, or PREVIOUS. If you want the process to return to its current state after the node restarts, use PREVIOUS.

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.

″"Configuring JVM sendRedirect calls to use context root" on page 41″ 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. For WebSphere ApplicationServer, 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	For WebSphere Application Server for z/OS, the default for the Control region is 48, and for the Servant region it is 128. For WebSphere Application Server for iSeries, the default is 96. For all other platforms, the default is 50.

Maximum Heap Size

Specifies the maximum heap size available to the JVM code, in megabytes.

Increasing the heap size can improve startup. For WebSphere Application Server, 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. If you enable debugging on multiple application servers on the same node, make sure that the servers are using different `address` arguments, which define the port for debugging. For example, if you enable debugging on two servers and leave the default debug port for each server as `address=7777`, the servers could fail to start properly.

Data type	String
Units	Java command-line arguments

Generic JVM Arguments

Specifies command line arguments to pass to the Java virtual machine code that starts the application server process.

The following are optional command line arguments that you can 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

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

Example: Setting Custom JVM Properties

In the WebSphere Application Server administrative console, you can change the values of the following custom JVM properties:

com.ibm.websphere.bean.delete.sleep.time

Specifies the time between sweeps to check for timed out beans. The value is entered in seconds. For example, a value of 120 would be 2 minutes. This property also controls the interval in the Servant process that checks for timed out beans still visible to the EJB container.

The default value is 4200 (70 minutes). The minimum value is 60 (1 minute). The value can be changed through the administrative console. To apply this property, you must specify the value in both the Control and Servant JVM Custom Properties.

Steps for this task

1. Connect to the administrative console and navigate to the Java Virtual Machine Custom Properties panel. For example, **Servers > Application Servers >***server1* **> Process Definition >***Control* **> Java Virtual Machine > Custom Properties** to define the property in the Control, or **Application Server >***server1* **> Process Definition >***Servant* **> Java Virtual Machine > Custom Properties** to define the property in the Servant.
2. If the **com.ibm.websphere.bean.delete.sleep.time** property is not present in the list, create a new property name.
3. Enter the name and value.

com.ibm.websphere.network.useMultiHome

Set this property in a multihomed environment where WebSphere Application Server is restricted to listen only on a specific IP address for Discovery and SOAP messages.

By default, the value of the property is true and the application server listens on all IP addresses on the host for Discovery and SOAP messages. If the property is set to false, then WebSphere Application Server will only listen on the configured host name for Discovery and SOAP messages. If you set the property to false, you should have a host name configured on WebSphere Application Server that resolves to a specific IP address.

You can change the value through the administrative console. Modify the defaults by setting the value for the server, deployment manager, and node agent. In order for these changes to take place, you must restart the server.

Steps for this task

1. To set this property, connect to the administrative console and navigate to the indicated page.

Application server	**Servers > Application Servers >***server1* **> Process Definition >***Control* **> Java Virtual Machine > Custom Properties**
Deployment manager	**System Administration > Deployment Manager > Process definition >** *Control* **> Java Virtual Machine > Custom Properties**
Node agent	**System Administration >Node Agent >** *nodeagent* **> Process definition >***Control* **> Java Virtual Machine > Custom Properties**

2. If the **com.ibm.websphere.network.useMultiHome** property is not present in the list, create a new property name and indicate its value.
3. Restart the server.

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. For more information about garbage collection in a Solaris operating environment see Performance: Resources for learning.

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. See Heap compaction (-Xnocompactgc) for more information.

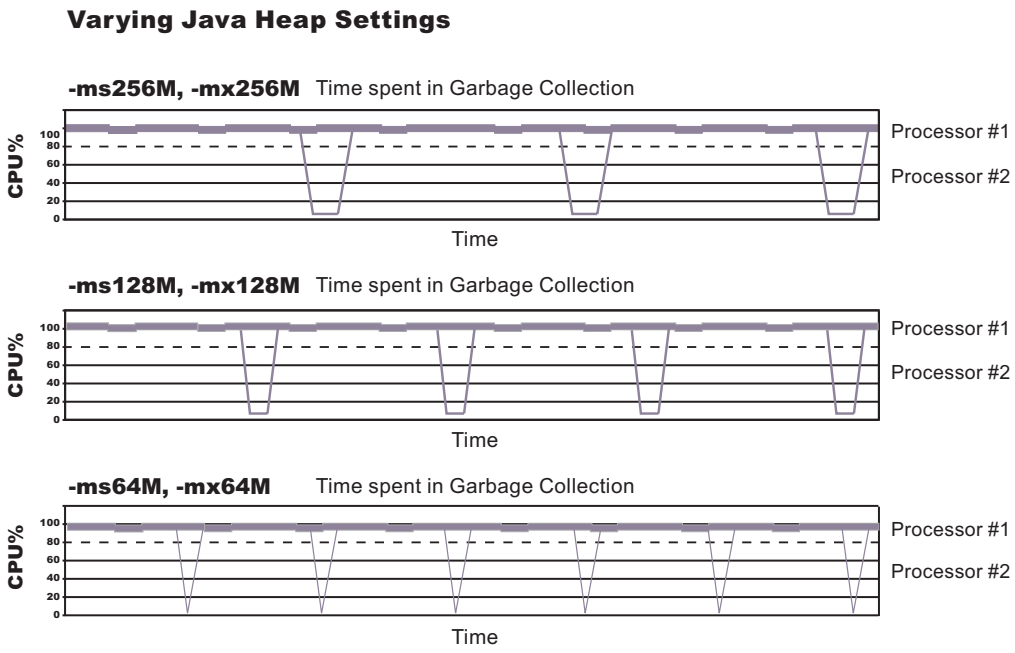
Java heap parameters

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. SeeHeap compaction for more information.

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 more information about garbage collection see Performance: Resources for learning.

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

Programming instructions and examples
- WebSphere Application Server education, http://www.software.ibm.com/wsdd/education/enablement/curriculum/cur_webappsrvadm.html
- Listing of all IBM WebSphere Application Server Redbooks, http://publib-b.boulder.ibm.com/Redbooks.nsf/Portals/WebSphere

Programming specifications
- The Java™ Virtual Machine Specification, Second Edition, http://java.sun.com/docs/books/vmspec/
- Sun's technology forum for the Java™ Virtual Machine Specification, http://forum.java.sun.com/forum.jsp?forum=37

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.

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 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. Managing Object Request Brokers

Default property values are set when the product is started and the Java Object Request Broker (ORB) service is initialized. These properties control the run-time behavior of the ORB and can also affect the behavior of product components that are tightly integrated with the ORB, such as security. It might be necessary to modify some ORB settings under certain conditions.

In every request/response exchange, there is a client-side ORB and a server-side ORB. It is important that the ORB properties be set for both sides as necessary.

After an ORB instance has been established in a process, changes to ORB properties do not affect the behavior of the running ORB instance. The process must be stopped and restarted in order for the modified properties to take effect.

The following steps are to be performed only as needed.

1. Adjust timeout settings to improve handling of network failures. Before making these adjustments, be sure to read "ORB tuning guidelines."
2. Adjust thread-pool settings used by the ORB for handling IIOP connections.
3. Use ORB custom property settings to enable and configure the Logical Pool Distribution (LPD) mechanism to improve performance on requests that have shorter execution times. LPD provides a mechanism allow shorter execution time requests greater access to execution threads. The LPD defaults are usually sufficient. However, when activating the LPD, it may be necessary to increase the ORB maximum thread pool size by 33 to 200 percent. Testing may be required to determine the best overall values for the ORB maximum thread pool size. Results will vary among systems and applications.

   For more information about LPD, see "Logical Pool Distribution."
4. If problems with the ORB arise, determine the problem. For help in troubleshooting, look at the ORB communications trace.

Object Request Brokers

An Object Request Broker (ORB) manages the interaction between clients and servers, using the Internet InterORB Protocol (IIOP). It enables clients to make requests and receive responses from servers in a network-distributed environment.

The ORB provides a framework for clients to locate objects in the network and call operations on those objects as if the remote objects were located in the same running process as the client, providing location transparency. The client calls an operation on a local object, known as a stub. Then the stub forwards the request to the desired remote object, where the operation is run and the results are returned to the client.

The client-side ORB is responsible for creating an IIOP request that contains the operation and any required parameters, and for sending the request on the network. The server-side ORB receives the IIOP request, locates the target object, invokes the requested operation, and returns the results to the client. The client-side ORB demarshals the returned results and passes the result to the stub, which, in turn, returns to the client application, as if the operation had been run locally.

This product uses an ORB to manage communication between client applications and server applications as well as communication among product components. During product installation, default property values are set when the ORB is initialized. These properties control the run-time behavior of the ORB and can also affect the behavior of product components that are tightly integrated with the ORB, such as security. This product does not support the use of multiple ORB instances.

Logical Pool Distribution (LPD)

The Logical Pool Distribution (LPD) thread pool mechanism implements a strategy for improving the performance of requests that have shorter execution times.

The need for LPD is indicated by a mixture of EJB requests where the execution times vary across the request types, and the ORB thread pool must be constrained for performance reasons. In this case, longer execution time requests may tend to elongate the response times for shorter execution time requests by denying them adequate access to threads in the thread pool. LPD provides a mechanism allow shorter execution time requests greater access to execution threads.

LPD divides the Object Request Broker (ORB) thread pool into logical pools, as configured by the administrator using ORB custom properties starting with com.ibm.websphere.threadpool.strategy.*. The size of each pool is a percentage of the maximum number of ORB threads. The sum of the logical pool percentages should equal 100.

When LPD is active, incoming ORB requests are vectored to a pool based on historical execution time history for the request type. The request type is determined by the method which is qualified internally as unique across components. The LPD mechanism adjusts pool targets at runtime to optimize the distribution of requests across logical pools.

After it is enabled, the LPD mechanism can be tuned. Tuning exercises should be driven by response time and throughput measurements, as well as statistics produced by the LPD mechanism.

Object Request Broker tuning guidelines

The following options exist for improving the performance of the Object Request Broker (ORB). Tuning results will vary among systems and applications.

- **Logical Pool Distribution (LPD) mechanism**

  If you suspect that requests with longer execution times are elongating the response times for shorter execution time requests by denying them adequate access to threads in the thread pool, LPD provides a mechanism to allow the shorter requests greater access to execution threads.

  For more information, see “Logical Pool Distribution (LPD)”
- **ORB timeout**

  If Web clients that access Java applications running in the product environment are consistently experiencing problems with their requests, and the problem cannot be traced to other sources and addressed through other solutions, consider setting an ORB timeout value and adjusting it for your environment.
  - Web browsers vary in their language for indicating that they have timed out. Usually, the problem is announced as a connection failure or no-path-to-server message.
  - Aim to set an ORB timeout value to less than the time after which a Web client eventually times out. Because it can be difficult to tell how long Web

clients wait before timing out, setting an ORB timeout requires experimentation. Another difficulty is that the ideal testing environment features some simulated network failures for testing the proposed setting value.
  - Empirical results from limited testing indicate that 30 seconds is a reasonable starting value. Mainly, you need to ensure that the setting is not too low. To fine-tune the setting, find a value that is not too low. Then gradually decrease the setting until reaching the threshhold at which the value becomes too low. Set the value a little above the threshold.
  - When an ORB timeout value is set too low, the symptom is numerous CORBA 'NO_RESPONSE' exceptions, which occur even for some requests that should have been valid. If requests that should have been successful, for example, the server is not down, are being lost or refused, the value is likely to be too low.

  Note: Do not adjust an ORB timeout value unless experiencing a problem, because configuring a value that is inappropriate for the environment can itself create a problem. If you set the value, experimentation might be needed to find the correct value for the particular environment. Configuring an incorrect value can produce results worse than the original problem.

  You can adjust timeout intervals for the product's Java ORB through the following administrative settings:
  - **Request timeout**, the number of seconds to wait before timing out on most pending ORB requests if the network fails
  - **Locate request timeout**, the number of seconds to wait before timing out on a locate-request message
  - **com.ibm.CORBA.ConnectTimeout**, the maximum timethe client ORB waits before timing out while establishing an IIOP connection. See "Object Request Broker service settings" on page 57 for more information.
- **Determining the ORB message size**

  The ORB breaks apart messages into fragments to send over the ORB connection. You can configure this fragment size through the com.ibm.CORBA.FragmentSize parameter.

  To determine the size of the messages being transferred over the ORB and the number of fragments required to do so, you must first enable ORB tracing in the ORB Properties page in the webui and then enable ORBRas tracing from the logging and tracing page in the webui. You'll probably also want to bump up the trace file sizes as this can generate a lot of data. Restart the server and run at least one iteration (preferably several) of the case you wish to measure.

  Then look at the traceable file and do a search for ″Fragment to follow: Yes″. This indicates that the ORB transmitted a fragment, but it still has at least one remaining fragment to send before the entire message has arrived. If No is indicated instead of Yes, this means that that particular fragment is the last in the entire message. It may also be the first if the message fit entirely into one fragment.

  If you go to the spot where ″Fragment to follow: Yes″ is located, you will find a block that looks similar to this:

Fragment to follow: Yes
Message size: 4988 (0x137C)
--
Request ID: 1411

This indicates that the amount of data in the fragment is 4988 bytes and the Request ID is 1411. Then if you do a search for all occurrences of ″Request ID: 1411″, you will see the number of fragments used to send that particular message. If you add all the associated message sizes, you will have the total size of the message that's being send through the ORB.

Object Request Broker service settings in administrative console

Use this page to configure the Java Object Request Broker (ORB) service.

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

Several settings are available for controlling internal Object Request Broker (ORB) processing. You can use these settings to improve application performance in the case of applications containing enterprise beans. You can make changes to these settings for the default server or any application server configured in the administrative domain.

Request timeout

Specifies the number of seconds to wait before timing out on a request message.

For use in command-line scripting, the full name of this system property is com.ibm.CORBA.RequestTimeout.

Data type	int
Units	Seconds
Default	180
Range	0 to 300

Request retries delay

Specifies the number of milliseconds between request retries.

For use in command-line scripting, the full name of this system property is com.ibm.CORBA.requestRetriesDelay.

Data type	int
Units	Milliseconds
Default	0
Range	0 to 60

Connection cache maximum

Specifies the largest number of connections allowed to occupy the connection cache for the service. If there are many simultaneous clients connecting to the server-side ORB, this parameter can be increased to support the heavy load up to 1000 clients.

For use in command-line scripting, the full name of this system property is com.ibm.CORBA.MaxOpenConnections.

Data type	Integer
Units	Connections
Default	240

Connection cache minimum

Specifies the smallest number of connections allowed to occupy the connection cache for the service.

Data type	Integer
Units	Connections
Default	100

ORB tracing

Enables the tracing of ORB GIOP messages.

This setting affects two system properties: com.ibm.CORBA.Debug and com.ibm.CORBA.CommTrace. If you set these properties through command-line scripting, you must set both to `true` in order to enable the tracing of GIOP messages.

Data type	Boolean
Default	Not enabled (`false`)

Locate request timeout

Specifies the number of seconds to wait before timing out on a LocateRequest message.

For use in command-line scripting, the full name of this system property is com.ibm.CORBA.LocateRequestTimeout.

Data type	int
Units	Seconds
Default	180
Range	0 to 300

Force tunneling

Controls how the client ORB attempts to use HTTP tunneling.

For direct access, the full name of this property is com.ibm.CORBA.ForceTunnel.

Data type	String
Default	`NEVER`
Range	Valid values are `ALWAYS`, `NEVER`, or `WHENREQUIRED`.

Additional information about valid values follows:

ALWAYS
: Use HTTP tunneling immediately, without trying TCP connections first.

NEVER
: Disable HTTP tunneling. If a TCP connection fails, a CORBA system exception (COMM_FAILURE) is thrown.

WHENREQUIRED
: Use HTTP tunneling if TCP connections fail.

Tunnel agent URL

Specifies the URL of the servlet used to support HTTP tunneling.

This must be a properly formed URL, such as `http://w3.mycorp.com:81/servlet/com.ibm.CORBA.services.IIOPTunnelServlet` or, for applets, `http://applethost:port/servlet/com.ibm.CORBA.services.IIOPTunnelServlet`. This field is required if **HTTP tunneling** is set.

For use in command-line scripting, the full name of this system property is com.ibm.CORBA.TunnelAgentURL.

Pass by reference

Specifies how the ORB passes parameters. If enabled, the ORB passes parameters by reference instead of by value, which avoids making an object copy. If you do not enable pass by reference, the parameters are copied to the stack before every remote method call is made, which can be expensive.

If the EJB client and the EJB server are installed in the same WebSphere Application Server instance, and the client and server use remote interfaces, enabling Pass by reference can improve performance up to 50%. Pass by reference helps performance only where non-primitive object types are passed as parameters. Therefore, int and floats are always copied, regardless of the call model.

Enable this property with caution, because unexpected behavior can occur. If an object reference is modified by the remote method, the caller might change.

For use in command line scripting, the full name of this system property is com.ibm.CORBA.iiop.noLocalCopies.

Data type	Boolean
Default	Not enabled (false)

The use of this option for enterprise beans with remote interfaces violates the EJB Specification, Version 2.0 (see section 5.4). Object references passed to EJB methods or to EJB home methods are not copied and can be subject to corruption.

Consider the following example:

```
Iterator iterator = collection.iterator();
MyPrimaryKey pk = new MyPrimaryKey();
while (iterator.hasNext()) {
   pk.id = (String) iterator.next();
   MyEJB myEJB = myEJBHome.findByPrimaryKey(pk);
}
```

In this example, a reference to the same MyPrimaryKey object passes into WebSphere Application Server with a different ID value each time. Running this code with Pass by reference `enabled` causes a problem within the application server because multiple enterprise beans are referencing the same MyPrimaryKey object. To avoid this problem, set the system property com.ibm.websphere.ejbcontainer.allowPrimaryKeyMutation to `true` when Pass by reference is enabled. Setting Pass by reference to `true` causes the EJB container to make a local copy of the PrimaryKey object. As a result, however, a small portion of the performance advantage of setting Pass by reference is lost.

As a general rule, any application code that passes an object reference as a parameter to an enterprise bean method or EJB home method must be scrutinized to determine if passing that object reference results in loss of data integrity or other problems.

Object Request Broker service settings

You can use these properties to set and monitor settings associated with the Java Object Request Broker (ORB) service.

Setting ORB properties through the administrative console

To view the administrative console page, click **Servers > Application Servers >** *serverName* **> ORB Service > Custom Properties**.

To add properties to the page, click **New** and enter at least a name (case-sensitive) and value for the property. Then click **Apply**. When you are finished entering properties, click **OK**.

Setting ORB Properties through the command line If you use `java` command, then you use the `-D` option, for example:

```
java -Dcom.ibm.CORBA.propname1=value1 -Dcom.ibm.CORBA.propname2=value2 ...
   application name
```

If you use the `launchclient` command, then you need to prefix the property with `-CC`, for example:

```
launchclient yourapp.ear -CCDcom.ibm.CORBA.propname1=value1 -CCDcom.ibm.CORBA.
   propname2=value2 ... optional application arguments
```

The custom properties page already might include Secure Sockets Layer (SSL) properties that were added during product setup. A list of additional properties associated with the Java ORB service follows.

com.ibm.CORBA.BootstrapHost

Specifies the DNS host name or IP address of the machine on which initial server contact for this client resides. This setting is deprecated and will be removed in a future release.

com.ibm.CORBA.BootstrapPort

Specifies the port to which the ORB connects for bootstrapping. In other words, the port of the machine on which the initial server contact for this client is listening. This setting is deprecated and will be removed in a future release.

Default 2809

com.ibm.CORBA.FragmentSize

Specifies the size of GIOP fragments used by the ORB. If the total size of a request exceeds the set value, the ORB breaks up and sends multiple fragments until the entire request is sent. This should also be set on the client side with a -D system property if using a stand-alone java application.

Consider adjusting `com.ibm.CORBA.FragmentSize` when the amount of data that is sent over IIOP exceeds one kilobyte. You should definitely adjust this setting if thread dumps show that most client side threads get stuck in a wait coming from

writeLock. Sometimes a low amount of fragmentation is allowed, so set this property so that most messages have few to no fragments. If you want to instruct the ORB not to break up any of the requests or replies it sends, set this property to zero. However, setting the value to zero does not prevent the ORB from receiving GIOP fragments in requests or replies sent by another existing ORB.

Units	Bytes.
Default	1024
Range	From 64 to largest value of Java integer type that is divisible by 8

com.ibm.CORBA.ListenerPort

Specifies the port on which this server listens for incoming requests. The setting of this property is valid only for client-side ORBs.

Default	Next available system-assigned port number
Range	0 to 2147483647

com.ibm.CORBA.LocalHost

Specifies the host name or IP address of the system on which the server ORB is running. The setting of this property is valid only for client-side ORBs. Otherwise, the ORB obtains a value at run time by calling InetAddress.getLocalHost().getHostAddress().

com.ibm.CORBA.ServerSocketQueueDepth

Corresponds to the length of the TCP/IP stack listen queue and prevents WebSphere Application Server from rejecting requests when there is not space in the listen queue. If there are several simultaneous clients connecting to the server-side ORB, you can increase this parameter to support up to 1000 clients.

Default	50
Range	From 50 to the largest value of Java int type

com.ibm.CORBA.ShortExceptionDetails

If set to any value, this specifies that the exception detail message that is returned whenever the server ORB encounters a CORBA system exception is to contain a short description of the exception as returned by the toString() method of java.lang.Throwable. Otherwise, the message contains the complete stack trace as returned by the printStackTrace() method of java.lang.Throwable.

com.ibm.websphere.threadpool.strategy. implementation

If set to com.ibm.ws.threadpool.strategy.LogicalPoolDistribution, this enables the Logical Pool Distribution (LPD) thread pool strategy the next time you start the application server.

Some requests have shorter execution times than others. LPD is a mechanism for allowing these shorter requests greater access to execution threads. For more information, see ″Logical Pool Distribution.″

com.ibm.websphere.threadpool.strategy. LogicalPoolDistribution.calcinterval

Specifies how often the Logical Pool Distribution (LPD) mechanism will readjust the pool execution target times. It cannot be turned off once this support is installed.

LPD must be enabled (see com.ibm.websphere.threadpool.strategy.implementation).

Data type	Integer
Units	Milliseconds
Default	30
Range	20,000 minimum

com.ibm.websphere.threadpool.strategy. LogicalPoolDistribution.lruinterval

Specifies how long the Logical Pool Distribution internal data is kept for inactive requests. The mechanism tracks several statistics for each request type received. Consider removing requests that have not been active for a while.

LPD must be enabled (see com.ibm.websphere.threadpool.strategy.implementation).

Data type	Integer
Units	Milliseconds
Default	300,000 (5 minutes)
Range	60,000 (1 minute) minimum

com.ibm.websphere.threadpool.strategy. LogicalPoolDistribution.outqueues

Specifies how many pools are created and how many threads are allocated to each pool in the Logical Pool Distribution mechanism.

The ORB parameter for max threads controls the total number of threads. The outqueues parameter is specified as a comma separated list of percentages that should add up to 100. For example, the list 25,25,25,25 will set up 4 pools, each allocated 25% of the available ORB thread pool. The pools are indexed left to right from 0 to n-1. Each outqueue is dynamically assigned a target execution time by the calculation mechanism. Target execution times are assigned to outqueues in increasing order so pool 0 gets the requests with the least execution time and pool n-1 gets requests with the highest execution times.

LPD must be enabled (see com.ibm.websphere.threadpool.strategy.implementation).

Data type	Integers in comma separated list
Default	25,25,25,25
Range	Percentages in list must total 100 percent

com.ibm.websphere.threadpool.strategy. LogicalPoolDistribution.statsinterval

If active, statistics will be dumped to stdout after this interval expires, but only if requests have been processed. This keeps the mechanism from filling the log files with redundant information. These stats are beneficial for tuning the Logical Pool Distribution mechanism.

LPD must be enabled (see com.ibm.websphere.threadpool.strategy.implementation).

Data type	Integer
Units	Milliseconds
Default	0 (meaning Off)
Range	30,000 (30 seconds) minimum

com.ibm.websphere.threadpool.strategy. LogicalPoolDistribution.workqueue

Specifies the size of a new queue where incoming requests wait for dispatching. Pertains to the Logical Pool Distribution mechanism.

LPD must be enabled (see com.ibm.websphere.threadpool.strategy.implementation).

Data type	Integer
Default	96
Range	10 minimum

com.ibm.CORBA.numJNIReaders

You can improve performance by setting the `com.ibm.CORBA.numJNIReaders` system property through a command-line script. This property specifies the number of threads to be shared for request handling when the native selector mechanism is enabled.

Valid Range	0-2147483647
Default	2

com.ibm.CORBA.ConnectTimeout

The `com.ibm.CORBA.ConnectTimeout` property specifies the maximum time in seconds that the client ORB waits before timing out when attempting to establish an IIOP connection with a remote server ORB. Generally, WebSphere client applications use this property. The property is not used by the application server by default. However, if necessary, you can specify the property for each individual application server through the administrative console.

Client applications specify the `com.ibm.CORBA.ConnectTimeout` property in the `orb.properties` file. You can add the property when running the `launchclient` script.

Begin by setting your timeout value to 20-30 seconds, but consider factors such as network congestion and application server load and capacity. Lower values can provide better failover performance, but may result in exceptions if the remote server does not have enough time to complete the connection.

Valid Range	0-300 (seconds)
Default	0 (the client ORB waits indefinitely)

Object Request Broker communications trace

The Object Request Broker (ORB) communications trace, typically referred to as *CommTrace*, contains the sequence of General InterORB Protocol (GIOP) messages sent and received by the ORB during application execution. It might be necessary to understand the low-level sequence of client-to-server or server-to-server interactions during problem determination. This article uses trace entries from log examples to explain the contents of the log and help you understand the interaction sequence. It focuses only in the GIOP messages and does not discuss in detail additional trace information that appears when intervening with the GIOP-message boundaries.

Location

When ORB tracing is enabled, this information is placed in *install_root*/logs/trace.

Usage notes

- Is this file read-only?

  Yes
- Is this file updated by a product component?

  This file is updated by the administrative function.
- How and when are the contents of this file used?

  You use this file to localize and resolve ORB-related problems.

How to interpret the output

The following sections refer to sample log output found later in this topic.

Identifying information

The start of a GIOP message is identified by a line which contains either ″OUT GOING:″ or ″IN COMING:″ depending on whether the message is sent or received by the process that is being traced.

Following the identifying line entry is a series of items, formatted for convenience, with information extracted from the raw message that identify the endpoints in this particular message interaction. See lines 3-13 in both examples. The formatted items include the following:

- GIOP message type (line 3)
- Date and time that message was recorded (line 4)
- Information useful in uniquely identifying the thread in execution when the message was recorded, with other thread-specific information (line 5, broken for publication in the reply example)
- Local and remote TCP/IP ports used for the interaction (lines 6 through 9)
- GIOP version, byte order, whether the message is a fragment, and message size (lines 10 through 13)

Request ID, response expected and reply status

Following the introductory message information, the request ID is an integer generated by the ORB. It is used to identify and associate each request with its corresponding reply. This is necessary because the ORB can receive requests from multiple clients and must be able to associate each reply with the corresponding originating request.

- Lines 15-17 in the request example show the request ID, followed by an indication to the receiving endpoint that a response is expected (CORBA allows sending of one-way requests for which a response is not expected.)
- Line 15 in the reply example shows the request ID; line 33 shows the reply status received after completing the previously sent request.

Object Key

Lines 18-20 in the request example show the object key, the internal representation used by the ORB during execution to identify and locate the target object intended to receive the request message. Object keys are not standardized.

Operation

Line 21 in the request example shows the name of the operation to be executed by the target object in the receiving endpoint. In this example, the specific operation requested is named `_get_value`.

Service context information

The service contexts in the message are also formatted for convenience. Each GIOP message might contain a sequence of service contexts sent/received by each endpoint. Service contexts, identified uniquely with an ID, contain data used in the specific interaction, such as security, character codeset conversion, and ORB version information. The content of some of the service contexts is standardized and specified by OMG, while other service contexts are proprietary and specified by each vendor. IBM-specific service contexts are identified with IDs that begin with `0x4942`.

Lines 22-41 in the request example illustrate typical service context entries. There are three service contexts in the request message, as shown in line 22. The ID, length of data, and raw data for each service context is printed next. Lines 23-25 show an IBM-proprietary context, as indicated by the ID `0x49424D12`. Lines 26-41 show two standard service contexts, identified by ID `0x6` (line 26) and `0x1` (line 39).

Lines 16-32 in the the reply example illustrate two service contexts, one IBM-proprietary (line 17) and one standardized (line 20).

For the definition of the standardized service contexts, see the CORBA specification. Service context 0x1 (CORBA::IOP::CodeSets) is used to publish the character codesets supported by the ORB in order to negotiate and determine the codeset used to transmit character data. Service context 0x6 (CORBA::IOP::SendingContextRunTime) is used by RMI-IIOP to provide the receiving endpoint with the IOR for the SendingContextRuntime object. IBM service context 0x49424D12 is used to publish ORB PartnerVersion information in order to support release-to-release interoperability between sending and receiving ORBs.

Data offset

Line 42 in the request example shows the offset, relative to the beginning of the GIOP message, where the remainder body of the request or reply message is located. This portion of the message is specific to each operation and varies from operation to operation. Therefore, it is not formatted, as the specific contents are not known by the ORB. The offset is printed as an aid to quickly locating the operation-specific data in the raw GIOP message dump, which follows the data offset.

Raw GIOP message dump

Starting at line 45 in the request example and line 36 in the reply example, a raw dump of the entire GIOP message is printed in hexadecimal format. Request messages contain the parameters required by the given operation

and reply messages contain the return values and content of output parameters as required by the given operation. For brevity, not all of the raw data has been included in the figures.

Sample Log Entry - GIOP Request

```
1.  OUT GOING:

3.  Request Message
4.  Date:          April 17, 2002 10:00:43 PM CDT
5.  Thread Info:   P=842115:O=1:CT
6.  Local Port:    1243 (0x4DB)
7.  Local IP:      jdoe.austin.ibm.com/192.168.1.101
8.  Remote Port:   1242 (0x4DA)
9.  Remote IP:     jdoe.austin.ibm.com/192.168.1.101
10.  GIOP Version:  1.2
11. Byte order:    big endian
12. Fragment to follow: No
13. Message size:  268 (0x10C)
--
15. Request ID:        5
16. Response Flag:     WITH_TARGET
17. Target Address:     0
18. Object Key:        length = 24 (0x18)
                       4B4D4249 00000010 BA4D6D34 000E0008
                       00000000 00000000
21. Operation:         _get_value
22. Service Context:   length = 3 (0x3)
23. Context ID:  1229081874 (0x49424D12)
24. Context data:  length = 8 (0x8)
                       00000000 13100003
26. Context ID:  6 (0x6)
27. Context data:  length = 164 (0xA4)
                       00000000 00000028 49444C3A 6F6D672E
                       6F72672F 53656E64 696E6743 6F6E7465
                       78742F43 6F646542 6173653A 312E3000
                       00000001 00000000 00000068 00010200
                       0000000E 3139322E 3136382E 312E3130
                       310004DC 00000018 4B4D4249 00000010
                       BA4D6D69 000E0008 00000000 00000000
                       00000002 00000001 00000018 00000000
                       00010001 00000001 00010020 00010100
                       00000000 49424D0A 00000008 00000000
                       13100003
39. Context ID:  1 (0x1)
40. Context data:  length = 12 (0xC)
                       00000000 00010001 00010100
42. Data Offset:       118


45. 0000: 47494F50 01020000 0000010C 00000005   GIOP............
46. 0010: 03000000 00000000 00000018 4B4D4249   ............KMBI
47. 0020: [remainder of message body deleted for brevity]
```

Sample Log Entry - GIOP Reply

```
1.  IN COMING:

3.  Reply Message
4.  Date:          April 17, 2002 10:00:47 PM CDT
5.  Thread Info:   RT=0:P=842115:O=1:com.ibm.rmi.transport.TCPTransportConnection
5a.   remoteHost=192.168.1.101 remotePort=1242 localPort=1243
6.  Local Port:    1243 (0x4DB)
7.  Local IP:      jdoe.austin.ibm.com/192.168.1.101
8.  Remote Port:   1242 (0x4DA)
9.  Remote IP:     jdoe.austin.ibm.com/192.168.1.101
10. GIOP Version:  1.2
```

```
11. Byte order:    big endian
12. Fragment to follow: No
13. Message size:  208 (0xD0)
--
15. Request ID:        5
16. Service Context:   length = 2 (0x2)
17. Context ID:  1229081874 (0x49424D12)
18. Context data:  length = 8 (0x8)
                       00000000 13100003
20. Context ID:  6 (0x6)
21. Context data:  length = 164 (0xA4)
                       00000000 00000028 49444C3A 6F6D672E
                       6F72672F 53656E64 696E6743 6F6E7465
                       78742F43 6F646542 6173653A 312E3000
                       00000001 00000000 00000068 00010200
                       0000000E 3139322E 3136382E 312E3130
                       310004DA 00000018 4B4D4249 00000010
                       BA4D6D34 000E0008 00000001 00000000
                       00000002 00000001 00000018 00000000
                       00010001 00000001 00010020 00010100
                       00000000 49424D0A 00000008 00000000
                       13100003
33. Reply Status:      NO_EXCEPTION


36. 0000: 47494F50 01020001 000000D0 00000005   GIOP............
37. 0010: 00000000 00000002 49424D12 00000008   ........IBM.....
38. 0020: [remainder of message body deleted for brevity]
```

Client-side programming tips for the Java Object Request Broker service

This article includes programming tips for applications that communicate with the client-side Object Request Broker (ORB) that is part of the Java ORB service.

Resolution of initial references to services

Client applications can use the properties *ORBInitRef* and *ORBDefaultInitRef* to configure the network location that the Java ORB service uses to find a service such as naming. Once set, these properties are included in the parameters used to initialize the ORB, as follows:

```
org.omg.CORBA.ORB.init(java.lang.String[] args,
                       java.util.Properties props)
```

You can set these properties in client code or by command-line argument. It is possible to specify more than one service location by using multiple ORBInitRef property settings (one for each service), but only a single value for ORBDefaultInitRef may be specified. For more information about the two properties and the order of precedence that the ORB uses to locate services, read the CORBA/IIOP specification, cited in "Resources for learning."

For setting in client code, these properties are com.ibm.CORBA.ORBInitRef.*service_name* and com.ibm.CORBA.ORBDefaultInitRef, respectively. For example, to specify that the naming service (NameService) is located in sample.server.com at port 2809, set the com.ibm.CORBA.ORBInitRef.NameService property to `corbaloc::sample.server.com:2809/NameService`.

For setting by command-line argument, these properties are `-ORBInitRef` and `-ORBDefaultInitRef`, respectively. To locate the same naming service specified previously, use the following Java command (split here for publication only):

```
java program -ORBInitRef
     NameService=corbaloc::sample.server.com:2809/NameService
```

After these properties have been set for services supported by the ORB, J2EE applications obtain the initial reference to a given service by calling the resolve_initial_references function on the ORB as defined in the CORBA/IIOP specification.

Preferred API for obtaining an ORB instance

For J2EE applications, you can use either of the following approaches. However, it is strongly recommended that you use the JNDI approach to ensure that the same ORB instance is used throughout the client application; you will avoid the unintended inconsistencies that might occur when different ORB instances are used.

JNDI approach: For J2EE applications (including enterprise beans, J2EE clients and servlets), you can obtain an ORB instance by creating a JNDI InitialContext object and looking up the ORB under the name `java:comp/ORB`, as follows:

```
javax.naming.Context ctx = new javax.naming.InitialContext();
org.omg.CORBA.ORB orb =
   (org.omg.CORBA.ORB)javax.rmi.PortableRemoteObject.narrow(ctx.lookup("java:comp/ORB"),
                                                            org.omg.CORBA.ORB.class);
```

The ORB instance obtained using JNDI is a singleton object, shared by all J2EE components running in the same Java virtual machine process.

CORBA approach: Because thin-client applications do not run in a J2EE container, they cannot use JNDI interfaces to look up the ORB. In this case, you can obtain an ORB instance by using CORBA programming interfaces, as follows:

```
java.util.Properties props = new java.util.Properties();
java.lang.String[] args = new java.lang.String[0];
org.omg.CORBA.ORB orb = org.omg.CORBA.ORB.init(args, props);
```

In contrast to the JNDI approach, the CORBA specification requires that a new ORB instance be created each time the ORB.init method is called. If necessary to change the ORB's default settings, you can add ORB property settings to the Properties object that is passed in the ORB.init() call.

The use of com.ibm.ejs.oa.EJSORB.getORBinstance(), supported in previous releases of this product, has been deprecated.

API restrictions with sharing an ORB instance among J2EE application components

For performance reasons, it often makes sense to share a single ORB instance among components in a J2EE application. As required by the J2EE Specification, Version 1.3, all web and EJB containers provide an ORB instance in the JNDI namespace as `java:comp/ORB`. Each container can share this instance among application components but is not required to. For proper isolation between application components, application code must comply with the following restrictions:

- Do not call the ORB shutdown or destroy methods

- Do not call org.omg.CORBA_2_3.ORB methods register_value_factory or unregister_value_factory

In addition, an ORB instance should not be shared among application components in different J2EE applications.

Required use of rmic and idlj shipped with the IBM Developer Kit

The Java Runtime Environment (JRE) used by this product includes the tools **rmic** and **idlj**. You use the tools to generate Java language bindings for the CORBA/IIOP protocol.

During product installation, the tools are installed in the directory *installation_root*/java/ibm_bin, where *installation_root* is the installation directory for the product. Versions of these tools included with Java development kits in $JAVA_HOME/bin other than the IBM Developer Kit installed with this product are incompatible with this product.

When you install this product, the directory *installation_root*/java/ibm_bin is included in the $PATH search order to enable use of the rmic and idlj scripts provided by IBM. Because the scripts are in *installation_root*/java/ibm_bin instead of the JRE standard location *installation_root*/java/bin, it is unlikely that you will overwrite them when applying maintenance to a JRE not provided by IBM.

In addition to the rmic and idlj tools, the JRE also includes Interface Definition Language (IDL) files. The files are based on those defined by the Object Management Group (OMG) and can be used by applications that need an IDL definition of selected ORB interfaces. The files are placed in the *installation_root*/java/ibm_lib directory.

Before using either the rmic or idlj tool, ensure that the *installation_root*/java/ibm_bin directory is included in the proper PATH variable search order in the environment. If your application will use IDL files in the *installation_root*/java/ibm_lib directory, also ensure that the directory is included in the PATH variable.

Character codeset conversion support for the Java Object Request Broker service

The CORBA/IIOP specification defines a framework for negotiation and conversion of character codesets used by the Java Object Request Broker (ORB) service. This product supports the framework and provides the following system properties for modifying the default settings:

com.ibm.CORBA.ORBCharEncoding
Specifies the name of the native codeset that the ORB is to use for character data (referred to as *NCS-C* in the CORBA/IIOP specification). By default, the ORB uses `UTF8`. (In contrast, the default value for versions 3.5.x and 4.0.x of this product was `ISO8859_1`, also known as Latin-1.) Valid codeset values for this property are shown in the table that follows this list; values that are valid only for ORBWCharDefault are indicated.

com.ibm.CORBA.ORBWCharDefault
Specifies the default codeset that the ORB is to use for transmission of wide character data when no codeset for wide character data is found in the tagged component in the Interoperable Object Reference (IOR) or in the GIOP service context. If no codeset for wide character data is found and

this property is not set, the ORB raises an exception, as specified in the CORBA specification. There is no default value set for this property. The only valid codeset values for this property are `UCS2` or `UTF16`.

The CORBA codeset negotiation/conversion framework specifies the use of codeset registry IDs as defined in the Open Software Foundation (OSF) codeset registry. The ORB translates the Java `file.encoding` names shown in the following table to the corresponding OSF registry IDs. These IDs are then used by the ORB in the IOR Codeset tagged component and GIOP Codeset service context as specified in the CORBA/IIOP specification.

Java name	OSF registry ID	Comments
ASCII	0x00010020	
ISO8859_1	0x00010001	
ISO8859_2	0x00010002	
ISO8859_3	0x00010003	
ISO8859_4	0x00010004	
ISO8859_5	0x00010005	
ISO8859_6	0x00010006	
ISO8859_7	0x00010007	
ISO8859_8	0x00010008	
ISO8859_9	0x00010009	
ISO8859_15_FDIS	0x0001000F	
Cp1250	0x100204E2	
Cp1251	0x100204E3	
Cp1252	0x100204E4	
Cp1253	0x100204E5	
Cp1254	0x100204E6	
Cp1255	0x100204E7	
Cp1256	0x100204E8	
Cp1257	0x100204E9	
Cp943C	0x100203AF	
Cp943	0x100203AF	
Cp949C	0x100203B5	
Cp949	0x100203B5	
Cp1363C	0x10020553	
Cp1363	0x10020553	
Cp950	0x100203B6	
Cp1381	0x10020565	
Cp1386	0x1002056A	
EUC_JP	0x00030010	
EUC_KR	0x0004000A	
EUC_TW	0x00050010	
Cp964	0x100203C4	
Cp970	0x100203CA	

Java name	OSF registry ID	Comments
Cp1383	0x10020567	
Cp33722C	0x100283BA	
Cp33722	0x100283BA	
Cp930	0x100203A2	
Cp1047	0x10020417	
UCS2	0x00010100	Valid only for ORBWCharDefault
UTF8	0x05010001	
UTF16	0x00010109	Valid only for ORBWCharDefault

For more information, read the CORBA/IIOP specification, cited in "Resources for learning."

Chapter 5. Welcome to Clusters

Clusters are groups of servers that are managed together and participate in workload management. A cluster can contain nodes or individual application servers. A node is usually a physical computer system with a distinct host IP address that is running one or more application servers. Clusters can be grouped under the configuration of a cell, which logically associates many servers and clusters with different configurations and applications with one another depending on the discretion of the administrator and what makes sense in their organizational environments.

Clusters are responsible for balancing workload among servers. Servers that are a part of a cluster are called cluster *members*. When you install an application on a cluster, the application is automatically installed on each cluster member.

Because each cluster member contains the same applications, you can distribute client tasks according to the capacities of the different machines by assigning weights to each server.

Assigning weights to the servers in a cluster improves performance and failover. Tasks are assigned to servers that have the capacity to perform the task operations. If one server is unavailable to perform the task, it is assigned to another cluster member. This has obvious advantage over running a single application server that can become overloaded if too many requests are made.

To learn more about clusters, see “Clusters” on page 73 and Chapter 6, “Balancing workloads with clusters,” on page 71 for more information.

Chapter 6. Balancing workloads with clusters

To monitor application servers and manage the workloads of servers, use server clusters and cluster members provided by the Network Deployment product.

To assist you in understanding how to configure and use clusters for workload management, below is a scenario. In this scenario, client requests are distributed among the cluster members on a single machine. (A client refers to any servlet, Java application, or other program or component that connects the end user and the application server that is being accessed.) In more complex workload management scenarios, you can distribute cluster members to remote machines.

1. Decide which application server you want to cluster.
2. Decide whether you want to configure replication domains and entries. Replication enables the sharing of data among processes and the backing up of failed processes.
3. Deploy the application onto the application server.
4. After configuring the application server and the application components exactly as you want them to be, create a cluster. The original server instance becomes a cluster member that is administered through the cluster.
5. You can create one or more cluster members of the cluster.
6. Regenerate the plug-in configuration. After changing configurations to plug-ins, transports or virtual hosts, you must regenerate your Web server plug-in for the changes to take effect.
7. Configure a backup cluster that handles requests if the primary cluster fails.
8. Start all of the application servers by starting the cluster. Workload management automatically begins when you start the cluster members of the application server.
9. Once you have the cluster running, you can perform the following tasks:
   - Stop the cluster.
   - Upgrade applications on clusters.
   - Detect and handle problems with server clusters and their workloads.
   - Tune the behavior of the workload management run time. If your application is experiencing problems with time-outs or your network experiences extreme latency, change the timeout interval for the com.ibm.CORBA.RequestTimeout property. Or, if the workload management state of the client is refreshing too soon or too late, change the interval for the com.ibm.websphere.wlm.unusable.interval property.

You need to define a bootstrap host for stand-alone Java clients. Stand-alone Java clients are clients that are located on a different machine from the application server and have no administrative server. Add the following line to the Java Virtual Machine (JVM) arguments for the client:

```
-Dcom.ibm.CORBA.BootstrapHost=machine_name
```

where *machine_name* is the name of the machine on which the administrative server is running.

Workload management (WLM)

Workload management optimizes the distribution of client processing tasks. Incoming work requests are distributed to the application servers, enterprise beans, servlets, and other objects that can most effectively process the requests. Workload management also provides failover when servers are not available, improving application availability.

Workload management provides the following benefits to WebSphere Application Server applications:

- It balances client workloads, allowing processing tasks to be distributed according to the capacities of the different machines in the system.
- It provides failover capability by redirecting client requests if one or more servers is unable to process them. This improves the availability of applications and administrative services.
- It enables systems to be scaled up to serve a higher client load than provided by the basic configuration. With clustering, additional instances of servers, servlets, and other objects can easily be added to the configuration.
- It enables servers to be transparently maintained and upgraded while applications remain available for users.
- It centralizes the administration of servers and other objects.

In the WebSphere Application Server environment, you implement workload management by using clusters, transports, and replication domains.

Techniques for managing state

Multimachine scaling techniques rely on using multiple copies of an application server; multiple consecutive requests from various clients can be serviced by different servers. If each client request is completely independent of every other client request, it does not matter whether consecutive requests are processed on the same server. However, in practice, client requests are not independent. A client often makes a request, waits for the result, then makes one or more subsequent requests that depend on the results received from the earlier requests. This sequence of operations on behalf of a client falls into two categories:

Stateless
: A server processes requests based solely on information provided with each request and does not reply on information from earlier requests. In other words, the server does not need to maintain state information between requests.

Stateful
: A server processes requests based on both the information provided with each request and information stored from earlier requests. In other words, the server needs to access and maintain state information generated during the processing of an earlier request.

For stateless interactions, it does not matter whether different requests are processed by different servers. However, for stateful interactions, the server that processes a request needs access to the state information necessary to service that request. Either the same server can process all requests that are associated with the same state information, or the state information can be shared by all servers that require it. In the latter case, accessing the shared state information from the same server minimizes the processing overhead associated with accessing the shared state information from multiple servers.

The load distribution facilities in WebSphere Application Server use several different techniques for maintaining state information between client requests:

- Session affinity, where the load distribution facility recognizes the existence of a client session and attempts to direct all requests within that session to the same server.
- Transaction affinity, where the load distribution facility recognizes the existence of a transaction and attempts to direct all requests within the scope of that transaction to the same server.
- Server affinity, where the load distribution facility recognizes that although multiple servers might be acceptable for a given client request, a particular server is best suited for processing that request.
- The WebSphere Application Server Session Manager, which is part of each application server, stores client session information and takes session affinity and server affinity into account when directing client requests to the cluster members of an application server. The workload management service takes server affinity and transaction affinity into account when directing client requests among the cluster members of an application server.

Clusters

Clusters are sets of servers that are managed together and participate in workload management. The servers that are members of a cluster can be on different host machines, as opposed to the servers that are part of the same node and must be located on the same host machine. A cell can have no clusters, one cluster, or multiple clusters.

Servers that belong to a cluster are *members* of that cluster set and must all have identical application components deployed on them. Other than the applications configured to run on them, cluster members do not have to share any other configuration data. One cluster member might be running on a huge multi-processor enterprise server system, while another member of that same cluster might be running on a smaller system. The server configuration settings for each of these two cluster members are very different, except in the area of application components assigned to them. In that area of configuration, they are identical. This allows client work to be distributed across all the members of a cluster instead of all workload being handled by a single application server.

When you create a cluster, you make copies of an existing application server template. The template is most likely an application server that you have previously configured. You are offered the option of making that server a member of the cluster. However, it is recommended that you keep the server available only as a template, because the only way to remove a cluster member is to delete it. Keeping the original intact allows you to reuse the template if you need to rebuild the configuration.

A *vertical cluster* has cluster members on the same node, or physical machine. A *horizontal cluster* has cluster members on multiple nodes across many machines in a cell. You can configure either type of cluster, or have a combination of vertical and horizontal clusters.

See Chapter 6, “Balancing workloads with clusters,” on page 71 to specify the amount of work targeted to each cluster member. You can distribute client tasks according to the capacities of different machines in the enterprise. A WebSphere Web server plug-in routes application access among cluster members by server-weighting, to provide better distribution control.

WebSphere Application Server can respond to increased use of an enterprise application by automatically replicating the application to additional cluster members as needed. This lets you deploy an application on a cluster instead of on a single node, without considering workload.

That multiple servers can service the same client request is also the basis for failover support. If a server fails while processing a client request, the failed request can be rerouted to any of the remaining cluster members. In fact, several servers could fail, and as long as at least one cluster member is running, client requests can continue to be serviced. For further information backing up failed processes, see "Replicating data" on page 84.

See "Backup clusters" on page 79 for information on backup cluster support. This feature allows the client to continue functioning when all cluster members of the primary cluster are not available.

Creating clusters

You can manage application servers collectively using a cluster. Use the Server Cluster page to view and manage the cluster.

1. Go to the Server Cluster page. Click **Servers > Clusters**. The Server Cluster page lists clusters of application servers in the cell and states whether a cluster is stopped, started or unavailable.
2. Click **New** to access the Create New Cluster page.
3. Type a cluster name.
4. To enable or disable node scoped routing optimization, select **Prefer local enabled**. The default is enabled, which indicates that, if possible, EJB requests are routed to the client's node. If you enable this feature, performance is improved because client requests are sent to local EJBs.
5. To enable memory-to-memory replication of HttpSession (for failover) or replication of cached data and cache invalidations with a Web Container's dynamic caching, select options supporting data replication.
6. Choose whether to create an empty cluster or to create a cluster based on an existing server.

   To create an empty cluster, do not include an existing server in this cluster.

   To create a cluster based on an existing server, choose **Select an existing server to add to this cluster** and select the server you want to add. The server you add becomes a template for any additional cluster members that you add to the cluster. Be sure that the template is configured correctly before adding more cluster members that are based on this template.
7. Click **Next**.
8. Add application servers (cluster members) to the cluster. For each new cluster member, do the following:
   a. Type the name of a new application server (cluster member) to add to the cluster.
   b. Select the node on which the server will reside.
   c. Specify the server weight. The weight value controls the amount of work directed to the application server. If the weight value for the server is greater than the weight values assigned to other servers in the cluster, then the server receives a larger share of the servers' workload. The value can range from 0 to 20.
   d. Make sure **Generate Unique HTTP Port** enabled.

e. Specify whether to create a replication entry for the server. A replication entry enables memory-to-memory replication of HttpSession (for failover) or replication of cached data and cache invalidations with a Web Container's dynamic caching.

f. Specify the server template.

g. Click **Apply** to finish the cluster member. Repeat the above steps to define another cluster member.

9. Click **Next** and review the summary of changes.
10. Click **Finish** to complete the configuration.
11. Define a virtual host with a unique port number.
    a. Go to the Virtual Hosts page of the administrative console.
    b. Click **Environment** > **Virtual Hosts** from the navigation tree of the administrative console.
    c. Click **New** and, on the settings page for a virtual host that displays, specify an administrative name for the virtual host.
    d. Under **Additional Properties**, click **Host Aliases** and define an unique port number for the virtual host.
12. Associate this virtual host and port number with each cluster member. This enables each cluster member to know which port to listen on for requests from the virtual host.
    a. Go to the Application Servers page of the administrative console.
    b. Select one of the application servers that is a member of your cluster.
    c. Under **Additional Properties**, click **Web Container**.
    d. On the Web Container page, under **Additional Properties**, click **HTTP Transport**.
    e. Click **New** and, on the settings page for an HTTP Transport, specify the virtual host name and port number defined in the previous step.
    f. Specify the virtual host name and port number for each cluster member by repeating the previous steps.
13. Click **Save** on the administrative console taskbar and save your administrative configuration. As part of saving the change to the configuration, you can select **Synchronize changes with Nodes** before clicking **Save** on the Save page.
14. Before you can start the cluster, the configuration needs to be synchronized to the nodes. If you selected **Synchronize changes with Nodes** when saving your configuration in the previous step, you can ignore this step. If you are running automatic synchronization, wait until synchronization runs. Or, run manual synchronization to get the configuration files moved to the nodes. Click **System Administration > Nodes** and, on the Nodes page, select the node and click **Synchronize** or **Full Resynchronize**. The Nodes page displays status indicating whether the node is synchronized.
15. To further configure a cluster, click on the cluster's name under **Name**. This displays the settings for the server cluster instance. Note that, unless you have clicked **Save** and saved your administrative configuration, you only see the **Configuration** and **Local Topology** tabs; to see the **Runtime** tab as well you must save your administrative configuration. Also, ensure that changes are synchronized to the nodes (step 12).

Server cluster collection

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

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

Click **New** to access the Create New Cluster page, which you use to define a new cluster.

Name
Specifies a logical name for the cluster. The name must be unique among clusters within the containing cell.

Status
Specifies whether cluster members are stopped, starting, or running.

If all cluster members are stopped, the cluster status and state is *Stopped*. After you request to start a cluster by clicking **Start** or **Ripplestart**, the cluster state briefly changes to *Starting* and each server that is a member of that cluster launches, if it is not already running. When the first member launches, the state changes to *PartialStart*. The state remains *PartialStart* until all cluster members are running, then the state changes to *Running* and the status is *Started*. Similarly, when stopping a cluster by clicking **Stop** or **ImmediateStop**, the state changes to *PartialStop* as the first member stops and changes to *Stopped* when all members are not running.

Server cluster settings
Use this page to view or change the configuration and local topology of a server cluster instance. Provided you saved your administrative configuration after creating the server cluster instance, you can also view run-time information such as the status of the server cluster instance.

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

Cluster name
Specifies a logical name for the cluster. The name must be unique among clusters within the containing cell.

Data type String

Cluster short name
Specifies the cluster short name for this cluster. For clustered servers, the WLM application environment is the cluster short name. The cluster short name must be unique in the cell and cannot equal the value specified on the `ClusterTransitionName` custom property of any non-clustered server. Clustered servers should not have a `ClusterTransitionName` set.

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

Data type String

Unique Id
Specifies the unique ID of this cluster.

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

Prefer local

Specifies whether enterprise bean requests are routed to the node on which the client resides, if it is possible to do so.

Select the **Prefer Local** check box to specify routing of requests to the node on which the client resides. By default, the **Prefer Local** check box is selected, specifying routing of requests to the node.

Data type	Boolean
Default	true

wlcID

Specifies the currently registered workload controller (WLC) identifier for the cluster. This setting might not display for all configurations.

Data type	String

State

Specifies whether cluster members are stopped, starting, or running.

If all cluster members are stopped, the cluster state is *websphere.cluster.stopped*. After you request to start a cluster, the cluster state briefly changes to *websphere.cluster.starting* and each server that is a member of that cluster launches, if it is not already running. When the first member launches, the state changes to *websphere.cluster.partial.start*. The state remains *websphere.cluster.partial.start* until all cluster members are running, then the state changes to *websphere.cluster.running*. Similarly, when stopping a cluster, the state changes to *websphere.cluster.partial.stop* as the first member stops and changes to *websphere.cluster.stopped* when all members are not running.

Data type	String
Range	Valid values are websphere.cluster.starting, websphere.cluster.partial.start, websphere.cluster.running, websphere.cluster.partial.stop, or websphere.cluster.stopped.

Creating cluster members

You create a cluster member to represent an application server in a cluster. To create a cluster member, view information about cluster members, or manage members of a cluster, use the Cluster Members page.

1. Go to the Cluster Members page. Click **Servers > Clusters** in the console navigation tree. Then, click a cluster in the collection of clusters and click **Cluster Members**. The Cluster Members page lists members of a cluster, states the nodes on which members reside, and states whether members are started, stopped or encountering problems.
2. Click **New** and follow the steps on the Create New Cluster Members page.
   a. Type a name for the cluster member (application server).
   b. Select the node on which the server will reside.
   c. Specify the server weight. The weight value controls the amount of work directed to the application server. If the weight value for the server is

greater than the weight values assigned to other servers in the cluster, then the server receives a larger share of the servers' workload. The value can range from 0 to 100.

d. Specify whether to generate a unique HTTP port.

e. Specify whether to create a replication entry for the server.

f. Specify the server template.

g. Click **Apply** to finish the cluster member. Repeat steps 1 through 7 to define another cluster member.

h. Click **Next**.

i. Review the summary of information on new cluster members and click **Finish**.

3. Click **Save** on the administrative console taskbar and save your administrative configuration.
4. To examine a cluster member's settings, click on the member's name under **Member Name** on the Cluster Members page. This displays the settings page for the cluster member instance.

Cluster member collection

Use this page to view information about and manage members of an application server cluster.

To view this administrative console page, click **Servers > Clusters >** *cluster_name* **> Cluster Members**.

Member name

Specifies the name of the server in the cluster. On most platforms, the name of the server is the process name. The name must match the (object) name of the application server.

Node

Specifies the name of the node for the cluster member.

Status

Specifies whether a cluster member is running, stopped, or unavailable.

If a cluster member is stopped, its status is *Stopped*. After you request to start a cluster member by clicking **Start**, the status becomes *Started*. After you click **Stop**, its status changes to *Stopped* when it stops running.

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 cluster member.

Cluster member settings

Use this page to configure a member instance of an application server cluster.

To view this administrative console page, click **Servers > Clusters >***cluster_name* **> Cluster Members >***cluster_member_name*.

Member Name

Specifies the name of the server in the cluster. On most platforms, the name of the server is the process name. The name must match the (object) name of the application server.

Data type String

Weight

Controls the amount of work directed to the application server. If the weight value for the server is greater than the weight values assigned to other servers in the cluster, then the server receives a larger share of the server workload.

Data type Integer
Range 0 to 20

Unique ID

Specifies a numerical identifier for the application server that is unique within the cluster. The ID is used for affinity.

Data type Integer

Backup clusters

Version 5.0.2 of WebSphere Application Server introduces ″mirrored cluster support,″ which provides backup clusters that mirror the primary server clusters. Mirrored cluster support is for EJB requests only.

Overview and prerequisites

When all members of a cluster are no longer available to service EJB requests, clients that must interact with one of the EJB application servers in the cluster do not function. Mirrored clusters enable an EJB cluster (primary cluster) to failover to another EJB cluster (backup cluster) when none of the EJB application servers in the primary cluster are able to service a request. The backup cluster allows the client to continue functioning when all cluster members of the primary cluster are not available.

If cluster members in the primary cluster subsequently become available, the mirrored cluster support attempts to fail back the requests to the primary cluster. This fail back is automatic, so you do not have to instruct the mirrored cluster support explicitly to allow fail back.

For the backup cluster to take over servicing requests successfully, the objects and resources available in the primary cluster must also be available in the backup cluster. You must install the same applications, use the same application names, and define the same resources in the backup cluster as in the primary cluster.

For successful fail back, the primary cluster must be defined as a backup for the backup cluster. In other words, both primary and backup clusters must have a backup configured, and each cluster's backup must point to the opposite cluster.

The primary and backup clusters must have the same cluster name and the same application names. Thus, the primary cluster and the backup cluster must reside in separate cells because a cluster must have a unique name within a cell.

The mirrored cluster failover support is not cell (domain) level failover or node agent failover. For the most part, the mirrored cluster support depends on a running deployment manager and node agents, and is limited to cluster failover only. For example, if the primary cluster's entire cell stops processing, a new

client's requests to the primary cluster will fail and are not sent to the backup cluster. Information regarding the backup cluster cannot be retrieved from the primary cluster's cell because the primary cluster's cell is not processing. On the other hand, if the primary cluster's cell manager or node agents and application servers stop processing after the client has already been sending requests to the primary cluster, the client already knows about the backup cluster and is able to send the requests to the backup cluster.

For cluster failover and fail back to function as expected, all of the servers (deployment manager, node agents and application servers) for both the primary cluster and the backup cluster must be at a release and level that provides mirrored cluster support; that is, Version 5.0.2 or later.

Configuration

The mirrored cluster support defines a mirrored cluster as a backup cluster which is optional and which a user might or might not configure. Each cluster can have only one backup cluster, which must be configured before it is specified as a backup cluster.

To configure a backup cluster in a cluster, you must specify a name and an endpoint. The endpoint is called the ″domain bootstrap address″ and consists of a bootstrap host and port. The bootstrap host is the host that contains the deployment manager in which the backup cluster is configured. The bootstrap port is this same deployment manager's bootstrap port.

The primary cluster and the backup cluster must reside in separate cells. To place mirrored clusters in separate cells, configure the appropriate backup cluster endpoint. The backup cluster bootstrap host and port determine which cell contains the backup cluster.

You can configure a backup cluster using the administrative console or the ExtendedCluster MBean. To determine the value for the domain bootstrap address and configure a backup cluster using the console, see ″Creating backup clusters.″

As to configuring a backup cluster using the console, you use the **Configuration** tab on the Domain Bootstrap Address page to statically define the backup cluster; the static value is consumed each time the deployment manager starts. You use the **Runtime** tab on the Domain Bootstrap Address page to define the backup cluster during run time only; when the deployment manager stops, the run-time backup cluster information is discarded.

As to configuring a backup cluster using the ExtendedCluster MBean, you can change the run-time configuration only. The MBean change does not affect the static configuration. You can use the setBackup operation on the ExtendedCluster MBean to change the run-time configuration. For example, you can use the following Java code to set the primary cluster's backup cluster:

```
ac.invoke(extendedCluster, "setBackup", new Object[] {
  backupClusterName, backupBootstrapHost, backupBootstrapPort},
  new String[] {
    "java.lang.String", "java.lang.String", "java.lang.Integer"});
```

Where *ac* is the AdminClient, and *extendedCluster* is the ExtendedClusterObjectName for the primary cluster.

Fail back support

There are two scenarios that affect the cluster fail back support.

In the first scenario, requests are made by the client to the primary cluster, which eventually stops accepting requests. The requests are then routed to the backup cluster. The client initially sent requests to the primary cluster and therefore has information or "knows about" the primary cluster. As a result, when the primary cluster is available again, the requests fail back to the primary cluster.

In the second scenario, the client does not start sending requests until after the primary cluster is down, and the requests go directly to the backup cluster. In this case, the client has information or "knows about" only the backup cluster. Since the client knows only about the backup cluster, when the primary cluster becomes available, the requests from this client continue to route to the backup cluster and do not fail back to the primary cluster when it becomes available. This scenario occurs when an object is created on the backup cluster. In this case, the backup cluster becomes the primary cluster for this object.

Both of these scenarios can occur within the same client at the same time, if the client is sending requests to existing objects and creating new objects after the primary cluster stops processing.

Creating backup clusters

You configure a backup cluster to handle requests if the primary cluster fails.

To configure a backup cluster, you must specify a name and an endpoint. The endpoint is called a "domain bootstrap address" and consists of a bootstrap host and port. The bootstrap host is the host that contains the deployment manager in which the backup cluster is configured. The bootstrap port is this same deployment manager's bootstrap port.

The primary cluster and the backup cluster must reside in separate cells. The backup cluster bootstrap host and port determine which cell contains the backup cluster.

1. Determine the bootstrap host and port of the backup cluster.
   a. Connect the administrative console to the deployment manager that contains the backup cluster.
   b. Click **System Administration > Deployment Manager > End Points > BOOTSTRAP_ADDRESS** to go to the BOOTSTRAP_ADDRESS endpoint page. The host and port for the BOOTSTRAP_ADDRESS instance is the host and port that the backup cluster will use.
   c. Connect the administrative console to the deployment manager that contains the primary cluster.
2. Click **Servers > Clusters > Backup Cluster** to go to the Backup Cluster page.
3. Ensure that the name of the backup cluster is the same as the containing server cluster.
4. Click **Domain Bootstrap Address** and, on the Domain Bootstrap Address page, specify the backup cluster's deployment manager bootstrap host and port in the **Host** and **Port** fields. Then, click **OK**. The bootstrap host and port combined define a bootstrap address for the deployment manager. Note that, on the Domain Bootstrap Address page, you use the **Configuration** tab to statically define the backup cluster; the static value is consumed each time the deployment manager starts. You use the **Runtime** tab to define the backup

cluster during run time only; when the deployment manager stops, the run-time backup cluster information is discarded.

5. Click **OK**.
6. Click **Save** on the administrative console taskbar and save your administrative configuration.

Backup cluster settings

Use this page to configure a backup server cluster. The backup server cluster is used if the primary server cluster fails.

Configuration of a backup cluster is only useful if the cluster contains an EJB module and a client outside of the cluster uses the EJB module.

You can view run-time information such as the status of the backup cluster, provided you saved your administrative configuration after configuring the backup cluster, restarted the server and the workload management configuration reads the backup cluster value when the server starts.

To view this administrative console page, click **Servers > Clusters >** *cluster_name* **> Backup Cluster**.

Backup Cluster Name

Specifies the name of the backup cluster. The backup cluster must have the same name as the server cluster containing the backup cluster. The backup cluster and its containing server cluster can have identical names because they must reside in different cells.

Data type String

Domain bootstrap address settings

Use this page to specify the bootstrap address end point of the deployment manager that contains the backup cluster. End points provide host and port specifications.

When no host or port values are shown for the bootstrap address endpoint, the backup cluster is not set. No host or port values are shown if a user has never set values for the backup cluster or if the user defined a backup cluster with an actual host and port and then removed the backup cluster by removing all text from the host and port fields.

After you set host and port values on the **Configuration** tab, the bootstrap address endpoint values become active when you restart the server. Use the **Runtime** tab to update the bootstrap address endpoint values dynamically. The system uses the run-time values until you restart the server or change the values. The values on the two tabs are independent. That is, you can specify a run-time backup cluster that is different from the configuration backup cluster.

To view this administrative console page, click **Servers > Clusters >** *cluster_name* **> Backup Cluster > Domain Bootstrap Address**.

Host

Specifies the IP address, domain name server (DNS) host name with domain name suffix, or just the DNS host name, of the bootstrap host for the deployment manager of the backup cluster.

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

Port

Specifies the bootstrap port number for the deployment manager of the backup cluster. The port value is used in conjunction with the host name.

Data type Integer

Replication

WebSphere Application Server provides a service that transfers data or events among WebSphere Application Server servers. The service is called *WebSphere Internal Replication*, or *replication* for short.

The replication service transfers both J2EE application data and any internal data used to maintain the application data among WebSphere run-time processes in a cluster of application servers.

Currently, the Web container in WebSphere Application Server leverages replication.

The replication service can replicate HttpSession data among processes and retrieve the HttpSession if the process that currently maintains the HttpSession fails. Using replication for HttpSession failover provides a potentially lower cost and more easily administrable alternative to storing HttpSession in a relational database. Further, the service can distribute across a WebSphere cluster information on invalid data and actual cached data maintained by a Web container's dynamic caching.

Replication entry

A replication entry (or *replicator*) is a run-time component that handles the transfer of internal WebSphere Application Server data.

WebSphere Application Server processes can connect to any replicator within a domain to receive data from other processes connected to any other replicator in the same domain. If the replicator a process is connected to goes down, the WebSphere Application Server process automatically attempts to reconnect to another replicator in the domain and recover data missed while unconnected.

You can define replicators to operate within a running application server process. Replicators are not enabled by default. You must define replicators as needed as part of application server and cluster management.

You can take the default settings for replicators or specify settings values for replicators that better suit your server configuration. The default configuration options are suitable for many scenarios.

Replication domain

A replication domain is a collection of replicator entry (or *replicator*) instances used by clusters or individual servers within a cell.

All replicators within a replication domain connect with each other, forming a network of replicators.

The default is to define a replication domain for a cluster when creating the cluster. However, replication domains can span across clusters.

Global default settings apply to all replication use for a given replication domain across a cell. Most default settings tune and control the behavior of replicator entries in managed servers across the cell. Such default settings control the use of encryption or the serialization and transferring of objects. Some default settings tune and control how specific WebSphere Application Server functions (for example, session manager and dynamic caching) leverage replication, such as session use of partitions.

For situations that require settings values other than the default, change the values for a given replication domain on the Internal Replication Domains page. Settings include various resource allocation, replication strategies (such as grouping or partitioning) and methods, as well as some security related items.

If you are using replication for HttpSession failover, you might also need to filter where the session replicates. For example, only replicate to two places out of many. The global default settings define the partition size or number of groups and the session manager settings define the groups to which a particular instance belongs.

Filtering is less important if you are using replication to distribute information on invalid data and actual cached data maintained by a Web container's dynamic caching. Replication does not occur for failover as much as for data synchronization across a cluster or cell when you likely want to avoid expensive costs for generating data potentially needed across those various servers.

Note that you can filter or segment by using multiple replication domains.

Replicating data

To enable the sharing of data among processes and the backing up of failed processes, you can use the replication service provided by WebSphere Application Server. To use the service, you define replication domains, which list interconnected replicator entries (residing in managed servers in the cell) that can exchange data.

There are two ways to define replication domains and replicator entries:

- You can use the Internal Replication Domains page and Replicator Entry page to define replication domains and replicator entries. To access the Internal Replication Domains page, click **Environment > Internal Replication Domains** in the console navigation tree. To access the Replicator Entry page, click a replication domain on the Internal Replication Domains page and then click **Replicator Entries**. When you create the entries on the Replicator Entry page, you can select any server for the replicator to reside in. The page lists all servers in the cell that do not already have replicators defined.
- You can define replication domains and replicator entries when you create a cluster on the Create New Cluster page. Using the page allows you to create a replication domain that has the same name as the cluster and, as you add or create new application servers in the cluster, define replicator entries in those servers. To access the Create New Cluster page, click **Servers > Clusters** in the console navigation tree to go to the Server Clusters page and click **New**.

A replicator does not need to run in the same process as the application server that uses it. However, it might be easier to manage replicators and replication domains if a one-to-one correspondence exists between replicators and application servers. During configuration, an application server connects by default to its local replicator, so you do not need to explicitly specify the replicator to use. All processes share equally in the replication cost.

1. Create an application server. Later, enable a replication domain and its replicators (step 2).

   Or, create a cluster and add an application server to it. When you define the cluster, you can specify that you want a replication domain associated with the cluster. Also, when you define a cluster, you can specify that you want a replicator associated with an application server. For example, you might specify that a replicator launch in the same Java virtual machine as a Web container. Or, you can enable a replicator later (step 2).
2. Create a replication domain if one is not already created for the processes you want supported by data replication. Go to the Replication Domains page and click **New**. On the settings for a replication domain instance, specify values for the instance. The default values generally will be sufficient, especially as to pooling and timeout values.
   a. Name the replication domain.
   b. Specify the timeout interval.
   c. Specify the encryption type. The DES and TRIPLE_DES options encrypt data sent between WebSphere Application Server processes and better secure the network joining the processes.
   d. Partition the replication domain to filter the number of processes to which data is sent. Partitioning the replication domain is most often done if you are replicating data to support retrieval of an HttpSession if the process maintaining the HttpSession fails. Partitioning is not supported for sharing of cached data maintained by Web container dynamic caching.
   e. Specify whether you want a single replication of data to be made. Enable the option if you are replicating data to support retrieval of an HttpSession if the process maintaining the HttpSession fails.
   f. Specify whether processes should receive data in objects or bytes. Processes receiving data in objects receive the data and class definitions. Processes receiving data in bytes receive the data only.
   g. Configure a pool of replication resources. Pooling replication resources can enhance the performance of the internal data replication service.
3. Create replicators for the processes you want supported by data replication, if replicators have not already been created for the processes. The default convention is to define a replicator in each application server that uses replication. However, you can define a pool of replicators, separate from the servers hosting applications.
   a. Click on the replication domain instance on the Replication Domains page and then **Replicator Entries** to access the Replicator Entry page.
   b. Click **New** and, on the replicator entry settings page, define a replicator. Specify a replicator name and, from the drop-down list of the available servers within the cell to which you can assign a replicator, select a server. Also specify a host name and ports. Note that a replicator has two end points (replicator and client end points) that use the same host name but have different ports.

4. If you use the DES or TRIPLE_DES encryption type for a replicator, click **RegenerateKey** on the settings for a replication domain instance at regular intervals, such as monthly.

   Periodically changing the key enhances security.

Internal replication domain collection

Use this page to view and manage replicator instances used within a cell. Replicators can transfer both application data and any internal data used to maintain the application data among WebSphere Application Server run-time processes in a cluster of application servers.

To view this administrative console page, click **Environment > Internal Replication Domains**.

Using replicators, you can replicate HttpSession data among processes and retrieve the HttpSession if the process that currently maintains the HttpSession fails. Further, you can distribute across a cell the creation, modification, and invalidation of cached data maintained by a Web container's dynamic caching.

If you are using replication for HttpSession failover, you will likely need to filter where the session replicates to. For example, only replicate to two places out of many. The global default settings define the partition size or number of groups and the session manager settings define the groups to which a particular instance belongs. Filtering is less important if you are using replication to distribute information on cached data maintained by a Web container's dynamic caching.

The default is to define a replication domain for a cluster when creating the cluster. However, you can create a new domain from this page. Click **New** and follow the instructions on the page displayed.

Clicking **Delete** deletes a domain and all replicators defined under the domain.

Name

Specifies a name for the replication domain.

Internal replication domain settings

Use this page to configure a replicator instance.

To view this administrative console page, click **Environment > Internal Replication Domains >***replication_domain_name*.

An application server connected to replicator within a domain can access the same set of data sent out by any application server connected to any other replicator (including the same replicator). Data is not shared across replicator domains.

Name

Specifies a name for the replication domain.

Data type String

Request Timeout

Specifies the number of seconds that a replicator waits when requesting information from another replicator before giving up and assuming the information does not exist. The default is 5 seconds.

Data type	Integer
Units	Seconds
Default	5

Encryption Type

Specifies the type of encryption used before transfer. The options include NONE, DES, TRIPLE_DES. The default is NONE. The DES and TRIPLE_DES options encrypt data sent between WebSphere processes and better secure the network joining the processes.

If you specify DES or TRIPLE_DES, a key for global data replication is generated after you click **Apply** or **OK**. When you use the DES or TRIPLE_DES encryption type, click **RegenerateKey** at regular intervals such as monthly because periodically changing the key enhances security.

Data type	String
Default	NONE

DRS Partition Size

Specifies the number of groups into which a replication domain is partitioned. By default, data sent by a WebSphere Application Server process to a replication domain is transferred to all other WebSphere Application Server processes connected to that replication domain. To filter or reduce the number of destinations for the data being sent, partition the replication domain. The default partition size is 10, and the partition size should be 10 or more to enhance performance.

Partitioning the replication domain is most often done if you are replicating data to support retrieval of an HttpSession if the process maintaining the HttpSession fails. Partitioning is not supported for sharing of cached data maintained by Web container dynamic caching. As to dynamic caching, all partitions or groups are always active and used for data replication.

When you partition a replication domain, you define the total number of groups or partitions. Use this setting to define the number of groups. Then, when you configure a specific session manager under a Web container or as part of an enterprise application or Web module, select the partition to which that session manager instance listens and from which it accepts data. To specify the groups to which an application server listens, change the settings for affected servers on a Session Manager page. In addition, you can set a *replicator role* for a server. This replicator role affects whether a WebSphere process sends data to the replication domain, receives data, or does both. The default is both to receive and send data.

Data type	Integer
Default	10

Single Replica

Specifies that a single replication of data be made. Enable this option if you are replicating data to support retrieval of an HttpSession if the process maintaining the HttpSession fails. This option restricts the recipient of the data to a single instance.

This setting provides filtering beyond grouping or partitioning. Using this setting, you can choose to have data only sent to one other listening instance in the replication domain.

Data type	Boolean
Default	false

Serialization Method

Specifies the object serialization method to use when replicating data. An administrative concern with replicating Java objects is locating the class definition, especially in a J2EE environment where class definitions might reside only in certain web modules or enterprise applications. Object serialization methods define whether the processes receiving data also need the class definition.

The options for this setting are OBJECT and BYTES. The default is BYTES.

OBJECT instructs a replicator to write the object directly to the stream. With OBJECT, a replicator must reinstantiate the object on the receiving side so must have the class definition.

BYTES instructs a replicator to break down the object into bytes and then send only the bytes across the stream. With BYTES, a replicator does not need to instantiate the object on the receiving side. The BYTES option is useful for failover, where the data is not used at the receiving side and thus the class definitions do not need to be stored there. Or, the option requires that you move class definitions from the Web application class path to the system class path.

Data type	String
Default	BYTES
Range	Valid values are OBJECT or BYTES.

DRS Pool Size

Specifies the maximum number of items allowed in a pool of replication resources. The default is 10.

Pooling replication resources can enhance the performance of the WebSphere internal data replication service.

Data type	Integer
Default	10
Range	1 to 50

DRS Pool Connections

Specifies whether the data replication service includes replicator connections in a pool of replication resources. Whether this option is enabled or not, the pool includes replicator sessions, publishers and subscribers.

The default is not to include replicator connections in the pool.

Data type	Boolean
Default	false

Replicator entry collection

Use this page to view and manage replicator entries.

To view this administrative console page, click **Environment > Internal Replication Domains >***replication_domain_name* **> Replicator Entries**.

To configure a new replicator entry, click **New** and follow the instructions on the page. You add a replicator to an existing server in the cell.

Replicator Name
Specifies a name for the replicator entry.

Replicator entry settings
Use this page to view and configure a replicator entry (or *replicator*).

To view this administrative console page, click **Environment > Internal Replication Domains >***replication_domain_name* **> Replicator Entries >***replicator_entry_name*.

Replicators communicate using TCP/IP. Thus, you must allocate an IP address and ports for replicators. Use this page to name a replicator and then to allocate an IP address and two ports (replicator and client ports) for the replicator.

Replicator Name
Specifies a name for the replicator entry.

Data type String

Server
Specifies the server for which you are defining a replicator. The drop-down list provides the names of servers that do not already have replicators.

Data type String
Default None

Replicator and Client 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 file, or HTML page).

A replicator port and client port share the same host name.

Data type String
Default None

Replicator Port
Specifies the port for which the Web server is configured to accept client requests. The port value is used in conjunction with the host name.

The replicator port enables communication among replicators. It provides replicator port to replicator communication. The usual value specified is 7874.

Data type Integer
Default None

Client Port
Specifies the port for which the Web server is configured to accept client requests. The port value is used in conjunction with the host name.

The client port enables communication between an application server process and a replicator. It provides client port to application server communication. The usual value specified is 7873.

Data type	Integer
Default	None

Starting clusters

Make sure that the members of your cluster have the debug port properly set. If multiple servers on the same node have the same debug port set, the cluster could fail to start. See "Java virtual machine settings" on page 37 for more information on how to change the debug port.

You can start all members of a cluster at the same time by requesting that the state of a cluster change to *running*. That is, you can start all application servers in a server cluster at the same time.

When you request that all members of a cluster start, the cluster state changes to *websphere.cluster.partial.start* and each server that is a member of that cluster launches, if it is not already running. After all members of the cluster are running, the cluster state becomes *websphere.cluster.running*.

1. Click **Servers > Clusters** in the console navigation tree to access the Server Cluster page.
2. Put a checkmark in the check boxes beside those clusters whose members you want started.
3. Click **Start** or **RippleStart**.
   - **Start** launches the server process of each member of the cluster by calling the node agent for each server to start the servers. After all servers are running, the state of the cluster changes to *websphere.cluster.running*. If the call to a node agent for a server fails, the server will not start.
   - **RippleStart** combines stopping and starting operations. It first stops and then restarts each member of the cluster.

Note to Windows users: If you start and stop application servers that are part of a cluster using the Windows Services facility, the cluster state does not always update correctly. For example, if a cluster is running and you stop a cluster member through the Services GUI, the cluster state remains as *Started* even though the server is no longer running.

Stopping clusters

You can stop all members of a cluster at the same time by requesting that the state of a cluster change to *stopped*. That is, you can stop all application servers in a server cluster at the same time.

Note to Windows users: If you start and stop application servers that are part of a cluster using the Windows Services facility, the cluster state does not always update correctly. For example, if a cluster is running and you stop a cluster member through the Services GUI, the cluster state remains as *Started* even though the server is no longer running.

1. Click **Servers > Clusters** in the console navigation tree to access the Server Cluster page.

2. Select the clusters whose members you want stopped.
3. Click **Stop** or **Immediate Stop**.
   - **Stop** halts each server in a manner that allows the server to finish existing requests and allows failover to another member of the cluster. When the stop operation begins the cluster state changes to *websphere.cluster.partial.stop*. After all servers stop, the cluster state becomes *websphere.cluster.stopped*.
   - **Immediate Stop** brings down the server quickly without regard to existing requests. When the stop operation begins, the cluster state changes to *websphere.cluster.partial.stop*. After all servers stop, the cluster state becomes *websphere.cluster.stopped*.

You can also stop and start server clusters from the settings page for a server cluster instance. To access such a page, click on the server cluster that you want to start or stop in the collection under **Name** on a Server Cluster page. You can view the status of a server cluster (that is, whether the cluster is started or stopped) on the **Runtime** tab of the settings page for a server cluster instance. Note that the **Runtime** tab is only shown if you have clicked **Save** on the administrative console taskbar since creating the server cluster instance.

Deleting clusters

Use this task to remove a cluster that has cluster members.

Before removing a cluster, you must uninstall all Enterprise applications added to the cluster. See Uninstalling applications for more information.

Removing a cluster will delete the cluster and all associated cluster members. When you delete a cluster, there is no option to keep certain cluster members or applications that you have installed on any part of the cluster.

1. Click **Servers > Clusters** in the console navigation tree to access the Server Cluster page.
2. Make sure the cluster you want to remove is **stopped**. If the cluster is **started**, see “Stopping clusters” on page 90.
3. Remove the cluster. Select the cluster and click **Delete**.
4. Save your configuration. Click **Save** on the administrative console task bar. As part of saving the change to the configuration, select **Synchronize changes with Nodes** before clicking **Save** on the Save page.

The cluster is deleted.

Deleting cluster members

Use this task to remove a cluster member from an existing cluster. Removing a cluster member deletes the associated application server. You cannot remove an application server from a cluster without deleting it.

1. Choose the cluster that your cluster member belongs to. Click **Servers > Clusters** in the console navigation tree to access the Server Cluster page. Select the cluster member’s cluster and click **Cluster Members**.
2. Make sure the cluster member you want to remove is **stopped**. If the cluster is **started**, see “Stopping servers” on page 21.
3. Remove the cluster member from the cluster. Select the cluster member you want and click **Delete**.

4. Save your configuration. Click **Save** on the administrative console task bar. As part of saving the change to the configuration, select **Synchronize changes with Nodes** before clicking **Save** on the Save page.

The cluster member is deleted.

Tuning a workload management configuration

You can set values for several workload management client properties to tune the behavior of the workload management run time. You set the properties as command-line arguments for the Java virtual machine (JVM) process in which the workload management client is running.

Caution: Set the values of these properties only in response to problems that you encounter. In most cases, you do not need to change the values. If workload management is functioning correctly, changing the values can produce undesirable results.

To change the property values, you can use the Java Virtual Machine page of the administrative console or use the wsadmin tool. In cases such as where a servlet is a client to an enterprise bean, use the administrative console page for the application server where the servlet is running to configure the properties. The steps below describe how to change the values using the console.

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 where the client is running.
   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 one or more of the following command-line arguments in the **Generic JVM arguments** field:**-Dcom.ibm.CORBA.RequestTimeout=*timeout_interval***If your application is experiencing problems with timeouts, this argument changes the value for the com.ibm.CORBA.RequestTimeout property, which specifies the timeout period for responding to requests sent from the client. This argument uses the -D option. *timeout_interval* is the timeout period in seconds. If your network experiences extreme latency, specify a large value to prevent timeouts. If you specify a value that is too small, an application server that participates in workload management can time out before it receives a response. Note: Be careful specifying this property; it has no recommended value. Set it only if your application is experiencing problems with timeouts.**-Dcom.ibm.websphere.wlm.unusable.interval=*interval***If the workload management state of the client is refreshing too soon or too late, this argument changes the value for the com.ibm.websphere.wlm.unusable.interval property, which specifies the time interval that the workload management client run time waits after it marks a server as unavailable before it attempts to contact the server again. This argument uses the -D option. *interval* is the time in seconds between attempts. The default value is 300 seconds. If the property is set to a large value, the server is marked as unavailable for a long period of time. This prevents the workload management refresh protocol from refreshing the workload management state of the client until after the time period has ended.
3. Click **OK**.

4. Stop the application server and then restart the application server.

Workload management run-time exceptions

The workload management service can throw the following exceptions if it encounters problems:

org.omg.CORBA.TRANSIENT with a minor code 1229066306 (0x40421042)
This exception is thrown if the workload management routing service cannot retry a request and the failure resulted from a connection error. This exception indicates that the application should invoke some compensation logic and resubmit the request.

org.omg.CORBA.NO_IMPLEMENT with a minor code 1229066304 (0x49421040)
This exception is thrown if the workload management service cannot contact any of the EJB application servers that participate in workload management.

The WebSphere Application Server client can catch these exceptions and then implement its own strategies to handle the situation. For example, it can display an error message if no servers are available.

The workload management routing service can reroute a failed request to a different target transparently to the application if the application will not be adversely affected by a second attempt. Currently, the only way is to check if the request did not execute in whole or part on the previous attempt. When a request executes in whole or in part, an *org.omg.CORBA.TRANSIENT with the minor code 1229066306 (0x49421042)* exception is thrown to signal that a request can be made again. This informs the application that another target might be available to satisfy the request, but the request could not be failed over transparently to the application. Thus, the application can resubmit the request. The routing service throws an *org.omg.CORBA.NO_IMPLEMENT with the minor code 1229066304 (0x49421040)* exception if it cannot locate a suitable target for the request. The exception is thrown, for example, if the cluster is stopped or if the application does not have a path to any of the cluster members.

Clustering and workload management: Resources for learning

Use the following links to find relevant supplemental information about clustering and workload management. 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
- WebSphere V5.0 Performance, Scalability, and High Availability: WebSphere Handbook Series, http://publib-b.boulder.ibm.com/Redbooks.nsf/RedbookAbstracts/SG246198.html

- IBM WebSphere Application Server V5.0 System Management and Configuration: WebSphere Handbook Series, http://publib-b.boulder.ibm.com/redbooks.nsf/RedbookAbstracts/sg246195.html?Open

Programming instructions and examples

- WebSphere Application Server education, http://www.software.ibm.com/wsdd/education/enablement/curriculum/cur_webappsrvadm.html
- Listing of all IBM WebSphere Application Server Redbooks, http://publib-b.boulder.ibm.com/Redbooks.nsf/Portals/WebSphere

Chapter 7. 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.

Cell-wide settings

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.

For more information, refer to Chapter 9, "Welcome to Cell-wide settings," on page 121.

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.

For more information, refer to "Configuring WebSphere variables" on page 128.

Chapter 8. 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.

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.

Property Name="esiEnable" Value="*true/false***"**
Used to enable or disable the Edge Side Include (ESI) processor. If the ESI processor is disabled, the other ESI elements in this file are ignored.

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

Property Name="esiMaxCacheSize" Value="*interger***"**
An integer specifying, in 1K byte units, the maximum size of the cache. The default maximum size of the cache is 1024K bytes (1 megabyte). If the cache is full, the first entry to be evicted from the cache is the entry that is closest its expiration time.

Property Name="ESIInvalidationMonitor" Value="*true/false***"**
Used to indicate whether or not the ESI processor should receive invalidations from the Application Server.

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

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.

In this example, the application server could potentially get up to 500 connections. (You take the number of nodes, 5, multiply it by the number of processes, 2, and then multiply that number by the number specified for the MaxConnections attribute, 50, for a total of 500 connections.)

This attribute is not necessary on the z/OS platform. The z/OS 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. for example, the portion of the plugin_cfg.xml file containing these elements might look like the following:

```
<Transport Hostname="192.168.1.2" Port="9443" Protocol="HTTPS">
<Property Name="keyring" value="c:/WebSphere/AppServer/keys/
       keyring.kdb"/>
<Property Name="stashfile" value="c:/WebSphere/AppServer/keys/
       keyring.sth"/>
<Property Name="password" value="WebAS"/>
```

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

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

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

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

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"  loggingEnabled="true"
   rmEnabled="false" traceLevel="PERF_DEBUG">
```

armEnabled (zero or one attribute for RequestMetrics)
This attribute indicates whether the ARM 4 agent is enabled in the plug-in. When it is set to `true`, the ARM 4 agent will be called.

Note: For the SunOne (iPlanet) Web server the following directive must be included in the obj.conf file to enable ARM 4 support:

```
AddLog fn="as_term"
```

If this directive is not included, the arm_stop procedure will never be called.

loggingEnabled (exactly one attribute for RequestMetrics)
This attribute indicates whether request metrics logging is enabled in the plug-in. When it is set to `true` and the traceLevel is not set to NONE, the request response time (and other request information) is logged. When it is set to `false`, there is no request logging. The value of loggingEnabled depends on the value specified for the system property com.ibm.websphere.pmi.reqmetrics.loggingEnabled. When this system property is not present, loggingEnable is set to `true`.

rmEnabled (exactly one attribute for RequestMetrics)
This attribute indicates whether or not the request metrics is enabled in the plug-in. When it is set to `true`, the plug-in request metrics will look at the filters and log the request trace record in the plug-in log file. This action is performed if a request passes the filters. When this attribute is set to `false`, the rest of the request metrics attributes will be ignored.

traceLevel (exactly one attribute for RequestMetrics)
When rmEnabled is `true`, this attribute indicates how much information is logged. When this attribute is set to `NONE`, no request logging is performed. When this attribute is not set to `NONE`, and loggingEnabled is set to `true`, the request response time (and other request information) is logged when the request is done.

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

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

type (exactly one attribute for each filter)
There are two types of filters: SOURCE_IP (for example, client IP address) and URI. For the SOURCE_IP filter type, requests are filtered based on a known IP address. You can specify a mask for an IP address using the asterisk (*). If the asterisk is used, the asterisk must always be the last character of the mask, for example 127.0.0.*, 127.0.*, 127*. For performance reasons, the pattern matches character by character, until either an asterisk is found in the filter, a mismatch occurs, or the filters are found as an exact match.

For the URI filter type, requests are filtered based on the URI of the incoming HTTP request. The rules for pattern matching are the same as matching SOURCE_IP address filters.

If both URI and client IP address filters are enabled, Request Metrics requires a match for both filter types. If neither is enabled, all requests are considered a match.

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

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

Supported distributed platform Web server plug-in configurations

Each of the supported distributed platform WebSphere Web server plug-ins run on a number of operating systems. The following table lists the supported distributed platform Web servers and plug-in executable files for each operating system.

Operating system	Web server	Plug-in executable file
Windows 2000 and Windows NT	IBM HTTP Server for distributed platforms	mod_ibm_app_server_ http.dll
	IBM HTTP Server (IHS) 2.0	mod_ was_ap20_http.dll
	Lotus Domino 5.0 and 6.0	libdomino5_http.dll
	Apache	mod_app_server_http.dll
	Apache 2.0	mod_ was_ap20_http.dll
	SunOne (iPlanet)	libns41_http.dll
	Microsoft Internet Information Server (IIS)	iisWASPlugin _http.dll
IBM AIX	IBM HTTP Server for distributed platforms (IHS)	mod_ibm_app_server_ http.so
	IBM HTTP Server for distributed platforms (IHS) 2.0	mod_ was_ap20_http.so
	Lotus Domino 5.0 and 6.0	libdomino5_http.so
	Apache	mod_app_server_http.so
	Apache 2.0	mod_ was_ap20_http.so
	SunOne (iPlanet)	libns41_http.so
HP-UX	IBM HTTP Server for distributed platforms (IHS)	mod_ibm_app_server_ http.sl
	IBM HTTP Server for distributed platforms (IHS) 2.0	mod_ was_ap20_http.sl
	Lotus Domino 5.0 and 6.0	libdomino5 _ http.sl
	Apache	mod_app_ server_http.sl
	Apache 2.0	mod_ was_ap20_http.sl
	SunONE (iPlanet)	libns41_http.sl
Solaris Operating Environment	IBM HTTP Server for distributed platforms (IHS)	mod_ibm_app_server_ http.so
	IBM HTTP Server for distributed platforms (IHS) 2.0	mod_ was_ap20_http.so
	Lotus Domino 5.0	libdomino5 _ http.so
	Lotus Domino 6.0	libdomino6 _ http.so
	Apache	mod_app_server_http.so
	Apache 2.0	mod_ was_ap20_http.so
	SunOne (iPlanet)	libns41_ http.so
LINUX	IBM HTTP Server for distributed platforms (IHS)	mod_ibm_app_server_ http.so

	IBM HTTP Server for distributed platforms (IHS) 2.0	mod_ was_ap20_http.so
	Apache	mod_app_server_http.so
	Apache 2.0	mod_ was_ap20_http.so

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

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

However, if you are using another brand of Web server or if you have installed two IBM HTTP Servers, on the same machine, you will have to specify a location for the plug-in for these other Web servers and for the second IBM HTTP Server.

For example, suppose the IBM HTTP Server ″installation 1″ is installed in the standard directory in which the IBM HTTP 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 if there is a second IBM HTTP Server ″installation 2″ installed at /opt/IHS2, when the Web server plug-in for this second Web server is installed, it too modifies the httpd.conf file in /opt/IBMHTTPD/conf, modifying the same information that the first plug-in had changed.

One way to avoid this conflict is to manually edit the httpd.conf file of the second IBM HTTP Server installation, instead of using the install process. (If 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, you can copy the modified configuration file from the first Web server and replace the configuration file of the second 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. You can then specify the location of one of the Web server files. Then repeat the plug-in installation, this time specifying the other location.

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. 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.
```

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.
3. If you edited the plugin-cfg.xml files on federated nodes, turn off automatic synchronization of nodes on the File Synchronization Service page to prevent the edited files from being overwritten. Because a plugin-cfg.xml file on a Network Deployment machine is stored in the configuration repository, the plugin-cfg.xml file is overwritten on all federated nodes in your network whenever the Network Deployment machine updates the configuration repository on these nodes. If you need to regenerate the plugin-cfg.xml file on the Network Deployment machine:
   a. Save your edited plugin-cfg.xml files on the federated nodes.
   b. Force node synchronization of each federated node.
   c. Replace the updated plugin-cfg.xml file with the saved copy or merge the customized pieces of the saved copy into the new plugin-cfg.xml file.

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.

Note: To get the current usage information for the plugin_cfg.xml file,run the GenPluginCfg -help script. You will receive the following information in response to this query:

```
Usage: GenPluginCfg [[-option.name optionValue] ...]

Valid Options:
==============

     -config.root configroot_dir
         (defaults to environment variable CONFIG_ROOT)
     -cell.name cell
         (defaults to environment variable WAS_CELL)
     -node.name node
         (defaults to environment variable WAS_NODE)
     -server.name server
         (required for single server plugin generation)
     -output.file.name file_name
         (defaults to configroot_dir/plugin-cfg.xml)
     -destination.root root
         (install root of machine configuration will be used on)
     -destination.operating.system windows/unix
         (operating system of machine configuration will be used on)
     -debug yes/no
         (defaults to no)

==============

Examples:

     To generate a plugin config for all of the clusters in a cell:
        GenPluginCfg -cell.name NetworkDeploymentCell

     To generate a plugin config for a single server:
        GenPluginCfg -cell.name BaseApplicationServerCell
        -node.name serverNode -server.name serverName
```

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 8, “Configuring Web server plug-ins,” on page 97.

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.

When the deployment manager is installed on a machine that is remote from the base WebSphere Application Server installation, one of the following solutions must be implemented in order for the `plugin-cfg.xml` file to retain the *install_root*`/AppServer` directory structures, and 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/cells` directory.

- **Command line**:

  At a command prompt, change to the `DeploymentManager/bin` directory and type `GenPluginCfg -destination.root <`*install_root*`>` 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 *install_root*`/AppServer` directories.

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

  ```
  GenPluginCfg -destination.root "E:\WebSphere\AppServer"
  ```

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

  Edit the `plugin-cfg.xml` file, located in the *install_root*`/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.

Installing a Global Security Kit for a Web server plug-in

If you intend to use the Secure Sockets Layer (SSL) transport (also known as HTTPS), in addition to the plug-in files, you must also install the Global Security Kit (GSKit) on the workstation hosting your Web server. This kit helps the Web server connect to your Application Server

There is one GSKit installation image per platform. (The GSKit image is the same for all Web servers running on that platform.) "Gskit install images files" on page 118 provides a list of the GSKit files by supported operating system.

To Install the GSKit installation image to the workstation on which the Web server is running:

1. On the workstation that is hosting your Web server, add the GSKit installation directory to the Web server's PATH statement.
2. Do a file search on the workstation hosting the Application Server to find the appropriate GSKit installation file for the workstation that is hosting your Web server, and move it to the GSKit installation directory you created in the previous step. If the workstation hosting the Application Server is different from the workstation hosting the Web Server, you will need to use FTP or another file transfer mechanism to download, in binary format, this file to the workstation hosting the Web server.
3. Using the native install process for the operating system you are running on your workstation, install this file onto workstation hosting the Web server. For example, DSMIT should be run on AIX, or the gsk5bas.exe, which invokes InstallShield, should be run on a Windows system.
4. If you intend to use Secure-socket layer SSL:
   - Configure the Web server for SSL support. (See your Web server documentation for a description of how to configure SSL for your specific Web server.)

     For an IBM HTTP Server for distributed platforms, you must also add the following lines to the bottom of the Web server `httpd.conf` file:

     ```
     LoadModule ibm_ssl_module modules/IBMModuleSSL128.dll
     Listen port_number
     Keyfile C:\ssl\http_session\plug-inKeys.kdb
     <VirtualHost virtual_host_name:port_number>
           ServerName virtual_host_name
           SSLEnable
           SSLClientAuth none
     </VirtualHost>
     ```

     These lines cause the Web server to listen on the specified port.

SSLClientAuth none indicates that you do not want to enable client authentication. If you want to use client authentication, change this line to **SSLClientAuth enable**.

This change causes the HTTP Server to send a request for a certificate to the browser. Your browser might prompt you to choose a certificate to send to the Web server for performing client authentication.

- Create an SSL key file for the Web server plug-in.

  The content of this file depends on whom you want to allow to communicate directly with the application server over the port number specified for the HTTPS internal transport. It defines the HTTPS server security policy. The following procedure describes how to create an SSL key file with a restrictive security policy in which only the WebSphere plug-ins for the Web server are allowed to connect to the application server HTTPS internal transport:

  a. Create an SSL key file without the default signer certificates.
     1) Start IKeyMan.
     2) Create a new key database file. Click **Key Database File** > **New**. Then specify settings:
        – Key database type: **JKS**
        – File Name: **appServerKeys.jks**
        – Location: *path _to_your_ keys_directory*

        Click **OK**.
     3) Enter a password (twice for confirmation) and click **OK**.
     4) Delete all of the signer certificates.
     5) Click **Signer Certificates** > **Personal Certificates**.
     6) Add a new self-signed certificate. Click **New Self-Signed** to add a self-signed certificate. Specify settings:
        – Key Label: **appServerTest**
        – Organization: **IBM**

        Click **OK**.
     7) Extract the certificate from this self-signed certificate so that you can import it into the plug-in SSL key file.
        – Click **Extract Certificate**. Specify settings:
          - Data Type: **Base64-encoded ASCII data**
          - Certificate file name: **appServer.arm**
          - Location: *path_to_your_keys_directory*

          Click **OK**.
     8) Import the plug-in certificate. Click **Personal Certificates** > **Signer Certificates** > **Add**. Specify settings:
        – Data Type: **Base64-encoded ASCII data**
        – Certificate file name: **appServer.arm**
        – Location: *path_to_your_keys_directory*

        Click **OK**.
     9) Enter **plug-in** for the label and click **OK**.
     10) Click **Key Database File** > **Exit**.

  b. Add the application server signer certificate to the plug-in SSL key file.
     1) Start the key management utility.
     2) Click **Key Database File** > **Open**.
     3) Select the file *fully_qualified_path*`plug-inKeys.kdb`.

        The `plug-inKeys.kdb` file is the key database file, created using the gskkyman utility, that contains the public keys, private keys, trusted CAs, and certificates for the Web server plug-ins.

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

4) Enter the associated password and click **OK**.
5) Click **Personal Certificates** > **Signer Certificates**.
6) Click **Add**. Then specify settings:
   – Data Type: **Base64-encoded ASCII data**
   – Certificate File Name: **appServer.arm**
   – Location: *path_to_your_keys_directory*
7) Click **OK**.
8) Click **Key Database File** > **Exit**.

c. Reference the key file in the administrative console.

   Reference the appropriate SSL key file in the default SSL settings configuration panel or in the HTTPS SSL settings configuration panel. Using the default SSL settings panel, you would:
   1) Start the administrative console.
   2) Open the Security Center.
   3) Specify settings in the default SSL configuration:
      – Key File Name: *fully_qualified_file_name*`appserver.jks`
      – Key File Password: enter your password
      – Key File Format: **JKS**
      – Trust File Name: (empty)
      – Trust File Password: (empty)
      – Client Authentication: **selected**

d. Modify the Web server plug-in configuration file to indicate that you are using an HTTPS internal transport, and to add the `keyring` and `stashfile` properties to the definition of this internal transport.

   Example: The ServerCluster definition for cluster `Cluster1` with servers `SY1_ClusterMember1`, and `SY1_ClusterMember2` defined, looks like the following:

```
<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>
```

where:
- `plug-inKeys.kdb` is the key database file created using the gskkyman utility, that contains the public keys, private keys, trusted CAs, and certificates for the Web server plug-ins.file containing the keys for the plug-ins.

  Note: The default password for viewing the plugin-key.kdb file using iKeyMan is WebAS.
- `plug-inpw.sth` is the stash file in which the gskkyman utility stored the encrypted database password for these certificates.

See your Web server documentation for more information about these files.

5. Stop the application server and the Web server and start them again.

   The configuration is complete. In order to activate the configuration, stop and restart both the application server and the Web server.

Gskit install images files

The following table lists the Global Security Kit (GSKit) installation image files for WebSphere plug-ins for Web servers that are running on a distributed platform. The appropriate file must be downloaded to the workstation on which your Web server is running.

These files are provided with your WebSphere Application Server product. If your Web server is running on a workstation other than the one on which the Application Server is running, you can do a file search and then download the appropriate GSKIT file to the workstation on which the Web server is running.

Operating system	GSKit Installation image file
Windows 2000 and Windows NT	gsk5bas.exe
AIX	gsk/gskkm.rte
HP-UX	gsk/gsk5bas.tar.Z
Solaris Operating Environment	gsk/gsk5bas.tar.Z
Linux	gsk/gsk5bas-5.0-4.79. i386.rpm

Plug-ins: Resources for learning

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

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

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

Programming instructions and examples
- IBM HTTP Server documentation, http://www-306.ibm.com/software/webservers/httpservers/library/index.html

- Listing of all IBM WebSphere Application Server Redbooks, http://publib-b.boulder.ibm.com/Redbooks.nsf/Portals/WebSphere
- WebSphere Application Server education, http://www.software.ibm.com/wsdd/education/enablement/curriculum/cur_webappsrvadm.html

Web server plug-in tuning tips

During normal operation, the backlog of connections pending to an application server is bound to grow. Therefore, balancing workloads among 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
```

Chapter 9. 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 10. 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 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`.

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, value and description for the variable. Then click **OK**.
4. Verify that the variable is shown correctly in the list of variables. The administrative console will not pick up errors in spelling. The variable is ignored if added incorrectly.
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 Class loading.

   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.

View links to additional information about “Environment: Resources for learning” on page 133

Programming instructions and examples

- WebSphere Application Server education, http://www.software.ibm.com/wsdd/education/enablement/curriculum/cur_webappsrvadm.html
- Listing of all IBM WebSphere Application Server Redbooks, http://publib-b.boulder.ibm.com/Redbooks.nsf/Portals/WebSphere

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
- AS/400
- CICS
- Cloudscape
- DB2
- DFSMS
- Domino
- Everyplace
- iSeries
- IBM
- IMS
- Informix
- iSeries
- Language Environment
- Lotus
- MQSeries
- MVS
- OS/390
- RACF
- Redbooks
- RMF
- SecureWay
- SupportPac
- Tivoli
- 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.