WebSphere Application Server for z/OS V5.0.1:

Servers

GA22-7958-00

Note
Before using this information, be sure to read the general information under "Notices" on page 121.

Compilation date: September 22, 2003

Contents

How to send your comments

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

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

  Be sure to include the document name and number, the WebSphere Application Server version you are using, and, if applicable, the specific page, table, or figure number on which you are commenting.

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Chapter 1. Welcome to Servers

The product provides application servers and more.

Application servers

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

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

See ″Configuring application servers″.

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

Clusters

Clusters are groupings of servers. Each server in a cluster is a *member* of the cluster. Using clusters simplifies administration in that applications installed to a cluster are automatically installed to each member in the cluster. Likewise, applications removed from a cluster are automatically removed from each member in the cluster. When you create a cluster with multiple members, each member contains the same applications and, thus, can service the same client requests.

Using clusters can optimize the distribution of client processing tasks (load balancing) and improve application availability (failover).

Clusters can help balance loads in that clustering enables multiple servers to service the same client request; a request from a given client can be routed to any of the cluster members. Thus, rather than having all client requests handled by a single application server, client work can now be distributed across all members of a cluster. This enables systems to be scaled up to serve a higher client load than could be provided by a single server. Further, you can configure multiple cluster members on the same physical machine (vertical scaling), configure multiple cluster members on different physical machines (horizontal scaling), or do both. Finally, you can specify the amount of work targeted to each member of a cluster. For example, you can distribute client tasks according to the capacities of the different machines in the enterprise.

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.

Java Messaging (JMS) servers

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.

Chapter 2. Configuring application servers

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

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

A WebSphere Application Server administrator can configure one or more application servers and perform tasks such as the following:

1. Create application servers.
2. Manage application servers.
3. Configure transports.
4. Set up peer restart and recovery
5. Develop custom services.
6. Define processes for the application server. As part of defining processes, you can define monitoring policies to track the performance of a process and name-value pairs for properties.
7. Use the Java virtual machine.

After preparing a server, deploy an application or component on the server. See ″“Preparing to host applications” on page 68″ for a sample procedure that you might follow in configuring the application server run-time and resources.

Related tasks

Tuning performance

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.

Related tasks

Deploying and managing applications

Understanding the need for application server naming conventions

There are a number of reasons why you need to establish a naming convention for application servers:

1. **Because WebSphere Application Server servers are like IMS or CICS regions.**
   - They contain tailored procedures for the control and server regions.
   - They contain tailored environmental variables for each instance of a server.
   - They contain environmental variables for each instance of a server.
   - Servers may be self-contained or dependent on other servers.
2. **For security.**
   - Regions have user IDs associated with them.
   - Users are allowed access to servers and objects within.
3. **For Workload Manager (WLM).**
   - Classification of regions and work within the regions
   - Application environments.

A WebSphere for z/OS application server consists of a number of address spaces which require the installation to manage configuration files, security profiles, workload classification constructs, and so forth. To create, manage, and recognize application servers, a template is needed for stamping out servers and server instances. The template needs to apply to the following:

- **Server names:**
  - Control region PROC names
  - Server region PROC names
  - Application Environment names
  - Instance names
- **Security:**
  - User/group/uid/gid
  - Control regions
  - Server regions
  - Instance names
- **Procedures:**
  - Environmental files
  - Library names
- **Other:**
  - DB2 collection and package names
  - Log stream names

Here is a system for creating servers based on a 4-character application naming scheme, which we refer to as *XXXX*. Since multiple instances of a server may exist on one or more systems in the WebSphere for z/OS environment, there is also a requirement to distinguish between servers. You can use a system that looks like the following:

Note: Everything is determined by 4 characters: *XXXX* (and *Y*).

Table 1.

CBserver name	= CBXXXX
- APPLENV name	= CBXXXX

Table 1. (continued)

CBserver instance name	= CBXXXXAY
- ORBsrvname default value	= CBXXXXAY
Userid for control region	= CBXXXXC
- PROC for control region	= CBXXXXC
Group id for control region	= CBXXXXG
Userid for server region	= CBXXXXS
- PROC for server region	= CBXXXXS
Group id for server region	= CBXXXX
Default remote userid	= CBXXXXI
Default local userid	= CBXXXXD
Group id for default ids	= CBXXXXP

Here are the user IDs. Change as desired.

Here are the groups/GIDS. Change as desired.

The naming convention is also applied to:

Table 2.

Server-specific log streams	= CBXXXX.ERROR.LOG
LRMs	= CBXXXX_LRM_DB2
LRMIs	= CBXXXXAY_LRMI_DB2
DB2 collections	= CBXXXX_PK
HFS File system names	= /WSCapps/CBXXXX/bin and /WSapps/CBXXXX/lib
OS File names	= hlq.CBXXXX.LOADLIB, hlq.CBXXXX.HFS, and hlq.CBXXXXAY.PARMS
and so forth.	

The part of the naming scheme which breaks down is the management of the UID/GID associated with RACF identities. There appears to be no easy mechanism to automate the assignment or association of these entities with userids.

For example, below you will see one way of defining the procs for the control and server regions associated with application server APP1. Notice that each server instance has its own unique data set containing environmental settings. You could easily change this scheme so that there is one PDS for the entire sysplex specifying different members. The important limitation to remember is that there is minimal capability to pass symbolic parameter overrides to the server regions.

Also notice that data set names indicate whether the data set is unique to the server or common across the sysplex. In our naming scheme, the second level qualifier indicates whether the data set is to be used:

- sysplex -wide
- only for servers running on a specific system
- server-wide
- only for a given server instance.

Control Region Proc:

```
//BBOASR1 PROC SRVNAME='BBOASR1A',
//       PARMS='',
//       CBCONFIG='/WebSphere390/CB390'
//* See instructions at the bottom of this file
//  SET BBOLIB='BBO'
//  SET LELIB='CEE'
//  SET DB2='DB2'
//  SET RELPATH='controlinfo/envfile'
//BBOASR1 EXEC PGM=BBOCTL,REGION=0M,
// PARM='/ -ORBsrvname &SRVNAME &PARMS'
//*STEPLIB  DD DSN=&BBOLIB..SBBOLD2,DISP=SHR
//*         DD DSN=&BBOLIB..SBBOLOAD,DISP=SHR
//*         DD DSN=&LELIB..SCEERUN,DISP=SHR
//*         DD DSN=&DB2..SDSNLOAD,DISP=SHR
//BBOENV    DD PATH='&CBCONFIG/&RELPATH/&SYSPLEX/&SRVNAME/current.env'
//CEEDUMP   DD SYSOUT=*,SPIN=UNALLOC,FREE=CLOSE
//SYSOUT    DD SYSOUT=*,SPIN=UNALLOC,FREE=CLOSE
//SYSPRINT  DD SYSOUT=*,SPIN=UNALLOC,FREE=CLOSE
```

Server Region Proc:

```
//BBOASR1S PROC IWMSSNM='BBOASR1A',PARMS='-ORBsrvname ',
//      CBCONFIG='/WebSphere390/CB390'
//* See instructions at the bottom of this file
//  SET BBOLIB='BBO'
//  SET LELIB='CEE'
//  SET DB2='DB2'
//  SET RELPATH='controlinfo/envfile'
//BBOASR1S EXEC PGM=BBOSR,REGION=0M,TIME=NOLIMIT,
// PARM='/ &PARMS &IWMSSNM'
//STEPLIB  DD DSN=&BBOLIB..SBBOULIB,DISP=SHR
//*         DD DSN=&BBOLIB..SBBOLD2,DISP=SHR
//*         DD DSN=&BBOLIB..SBBOLOAD,DISP=SHR
//*         DD DSN=&LELIB..SCEERUN,DISP=SHR
//*         DD DSN=&DB2..SDSNLOAD,DISP=SHR
//BBOENV   DD PATH='&CBCONFIG/&RELPATH/&SYSPLEX/&IWMSSNM/current.env'
//CEEDUMP  DD SYSOUT=*
//SYSOUT   DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
```

WebSphere for z/OS run-time HFS

Overview

The following two sections show the HFS structures for a base Application Server node and a Deployment Manager node after you configure each in the Customization Dialog. The third and last section depicts the HFS structure of a location service daemon.

HFS structure of a base Application Server node

```
/<mount point>
  /<cell_shortname>.<node_shortname>.<server_shortname>
  /Daemon
  /AppServer
    /config
      /cells
        /<cell_name>
          cell.xml
          /applications
          /clusters
          /nodes
            /<node_name>
              node.xml
```

```
            serverindex.xml
            /servers
              /<server_name>
                was.env
      server.xml
```

HFS structure of a Deployment Manager node

```
/<DM mount point>
  /<DMcell_shortname>.<DMnode_shortname>.<DMserver_shortname>
  /Daemon
  /DeploymentManager
    /config
      /cells
        /<DMcell_name>
          cell.xml
          /applications
          /clusters
          /nodes
            /<DMnode_name>
              node.xml
              serverindex.xml
              /servers
                /<DMserver_name>
                  was.env
      server.xml
```

Note: The HFS for the Deployment Manager node is very similar to that of the base Application Server node because the nodes themselves are very much alike.

<DM mount point>
Directory that serves as the mount point for the HFS file system and holds the Deployment Manager's information and applications. Set the Deployment Manager mount point name to anything you wish.

<DMcell_shortname>.<DMnode_shortname>.<DMserver_shortname>
Symbolic link to `DMserver_name` directory, which houses the `was.env` file.

HFS structure of a location service daemon

```
/<mount point>
  /<cell_shortname>.<node_shortname>.<server_shortname>
  /<cell_shortname>.<cell_shortname>.<systemname>
  /Daemon
    /config
      /cells
        /<cell_name>
          /<cell_shortname>
            /SYSD
              was.env
```

<cell_shortname>.<cell_shortname>.<systemname>
Symbolic link to `/SYSD` directory, which houses the `was.env` file.

Note: The was.env file that the location service daemon uses is a different file than the one the controllers and servants use.

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. 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.

Related concepts

"Application servers" on page 3

Related tasks

Deploying and managing using scripting

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.

Related tasks

"Creating application servers" on page 8

Default server values for WebSphere Application Server for z/OS V5

Purpose

The following table lists the default server values for WebSphere Application Server for z/OS V5.

Parameters

Table 3. Default server values for WebSphere Application Server for z/OS V5

Server	Server name (long name)	Server short name	Cluster transition name / Application environment name	Run-time controller start procedure name	Run-time servant start procedure name	Subsystem type	Start parameter	Limit on starting server address space for a subsystem instance
Application Server	server1	BBOS001	BBOC001	BBO5ACR	BBO5ASR	CB	JOBNAME=&IWMSSNM.S, ENV=*cellname.nodename.*&IWMSSNM	No limit
Deployment Manager (ND only)	BBODMGR	BBODMGR		BBO5DCR	BBO5DSR	CB	JOBNAME=&IWMSSNM.S, ENV=*cellname.nodename.*&IWMSSNM	No limit

Table 3. Default server values for WebSphere Application Server for z/OS V5 (continued)

Server	Server name (long)	Server short name	Cluster transition name / Application environment name	Run-time controller start procedure name	Run-time servant start procedure name	Subsystem type	Start parameter	Limit on starting server address space for a subsystem instance
Location service daemon	BBODMNB (base) or BBODMNC (ND)			-	-	-	-	-
JMS server	BBOJ001			-	-	-	-	-
Node agent (ND only)	BBON001			-	-	-	-	-

Managing Application Servers

To view information about an Application Server, use the Application Servers page. For the Network Deployment product, you can also use the Application Servers page to manage Application Servers. For the base WebSphere Application Server product, you cannot manage Application Servers from the administrative console; you must manage Application Servers from a console hosted by a Network Deployment deployment manager (dmgr) process. From the base product or the Network Deployment product, use the wasadmin tool or command line tools such as startServer and stopServer to manage the Application Server.

1. Access the Application Servers page. Click **Servers > Application Servers** in the console navigation tree.
2. View information about Application Servers. The Application Servers page lists Application Servers in the cell and the nodes holding the Application Servers. The Network Deployment product also shows the status of the Application Servers. The status indicates whether a server is started, stopped, or unavailable.

   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.

Related tasks

Deploying and managing using scripting

Hot deployment and dynamic reloading

Server collection

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

Application Servers

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

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

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

JMS Servers

Each JMS server provides the functions of the JMS provider for a node in your administrative domain. There can be at most one JMS server on each node in the administration domain, and any application server within the domain can access JMS resources served by any JMS server on any node in the domain.

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

Related concepts

"Application servers" on page 3

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

Node
A node is a logical grouping of managed servers.

Related tasks

Chapter 2, "Configuring application servers," on page 3

Managing WebSphere internal JMS servers
Use this task to manage WebSphere internal JMS servers on WebSphere Application Server Network Deployment and WebSphere Application Server Enterprise.

Related reference

Administrative console buttons
The following button choices are available on various pages of the administrative console, depending on which product features you have enabled.

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

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

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

Object names

Node agent collection
Use this page to view information about node agents. Node agents are administrative agents that monitor application servers on a host system and route administrative requests to servers. A node agent is the running server that represents a node in a Network Deployment environment.

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

Related information

An overview of WebSphere asynchronous messaging using JMS

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

Node
Specifies the name of the node for the application server.

Status
Indicates whether the application server is started or stopped. (Network Deployment only)
Note that if the status is *Unavailable*, the node agent is not running in that node and you must restart the node agent before you can start the server.

Application server settings
Use this page to view or change the settings of an application server instance. To view this administrative console page, click **Servers > Application Servers >***server_name*.

The **Configuration** tab provides editable fields and the **Runtime** tab provides read-only information. The **Runtime** tab is available only when the server is running.

Related concepts

"Application servers" on page 3

Class loaders
Class loaders are part of the Java virtual machine (JVM) code and are responsible for finding and loading class files. Class loaders affect the packaging of applications and the run-time behavior of packaged applications deployed on application servers.

Related tasks

Chapter 2, "Configuring application servers," on page 3

Related reference

Administrative console buttons
The following button choices are available on various pages of the administrative console, depending on which product features you have enabled.

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

Object names

Administration service settings
Use this page to view and change the configuration for an administration service.

Class loader collection
Use this page to manage class-loader instances on an application server. A class loader determines whether an application class loader or a parent class loader finds and loads Java class files for an application.

Enterprise application collection
Use this page to view and manage enterprise applications.

"Custom service collection" on page 53

Debugging Service details
Use this page to view and modify the settings used by the Debugging Service.

Dynamic cache service settings
Use this page to configure and manage the dynamic cache service settings.

EJB container settings
Use this page to configure and manage a specific EJB container.

Web container advanced settings
Use this page to support Web container advanced settings. This support includes Network QoS and transaction class mapping

Message listener port collection
The message listener ports configured in the administrative domain

Object Request Broker service settings in administrative console
Use this page to configure the Java Object Request Broker (ORB) service.

"Process definition settings" on page 56

Server security settings
Use this page to configure server security and override the global security settings. If you need to revert to the global security defaults, deselect the appropriate check box in the administrative console.

Diagnostic trace service settings
Use this page to review and modify the properties of the diagnostic trace service.

Transaction service settings
Use this page to modify transaction service settings.

Web container settings
Use this page to configure the container settings.

Name:

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

Data type	String
Default	server1

Initial State:

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

Data type	String
Default	Started

Application Class loader Policy:

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

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

Data type	String
Default	MULTIPLE

Application Classloading Mode:

Specifies whether the class loader should search in the parent class loader or in the application class loader first to load a class. The standard for JDK 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 overriden classes and non-overriden classes.

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

Data type	String
Default	PARENT_FIRST

Short name:

Specifies the short name of the server.

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

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

Unique Id:

Specifies the unique ID of this server.

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

Process ID:

Specifies a string identifying the process.

Data type	String

Cell Name:

Specifies the name of the cell for the application server.

Data type	String
Default	*host_name*Network

Node Name:

Specifies the name of the node for the application server.

Data type	String

State:

Indicates whether the application server is started or stopped.

Data type	String
Default	Started

End point collection:

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

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

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

Related reference

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

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

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

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* **> End Points >***end_point_name*

Related reference

Administrative console buttons
The following button choices are available on various pages of the administrative console, depending on which product features you have enabled.

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

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

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

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.

Related reference

Administrative console buttons
The following button choices are available on various pages of the administrative console, depending on which product features you have enabled.

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

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

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

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*

Related reference

Administrative console buttons
The following button choices are available on various pages of the administrative console, depending on which product features you have enabled.

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

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

Native processes:

Use this page to view and modify properties of the JMS Integral Provider native processes.

To view this administrative console page, click **Servers** > **Application Servers** > *server name* > **Server Components** > **JMS Servers**.

Related tasks

Configuring session tracking

Related reference

Administrative console buttons
The following button choices are available on various pages of the administrative console, depending on which product features you have enabled.

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

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

Short name:

Specifies the short name of the JMS queue manager.

The name is 1-4 characters, alpha-numeric or national language. It cannot start with a numeric.

The system assigns a unique default short name for the JMS queue manager.

Data type String

Command Prefix:

Specifies the subsystem command prefix for the JMS queue manager.

This field is read only because it is only configured through the WebSphere ISPF Customization Dialog.

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**.

Related reference

Administrative console buttons
The following button choices are available on various pages of the administrative console, depending on which product features you have enabled.

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

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

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

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*.

Related reference

Administrative console buttons
The following button choices are available on various pages of the administrative console, depending on which product features you have enabled.

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

"End point settings" on page 15

"Custom property collection" on page 17

"Custom service collection" on page 53
"Server component collection" on page 19

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

Servant Instance Settings:

Use this page to support Servant instance settings. This support controls the number of servant processes

To view this administrative console page, click **Servers** > **Application Servers** > *server name* > **Servant Instances**.

Related tasks

Configuring session tracking

Related reference

Administrative console buttons
The following button choices are available on various pages of the administrative console, depending on which product features you have enabled.

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

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

Multiple Instances Enabled:

Specifies whether multiple servant process instances are enabled for this server.

When the Multiple Instances Enabled setting is disabled, the server has exactly one servant process instance. When the Multiple Instances Enabled setting is enabled, the server has the following:
- A minimum number of servant process instances as specified on the Minimum Number of Instances setting
- A maximum number of servant process instances as specified on the Maximum Number of Instances setting

The z/OS Workload Manager dynamically determines the actual number of servant process instances.

Minimum Number of Instances:

Specifies the minimum number of servant process instances to activate

Data type	Integer
Range	1 to 255, inclusive

Maximum Number of Instances:

Specifies the maximum number of servant process instances to activate

Data type	Integer
Range	1 to 255, inclusive

WebSphere internal JMS server settings

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

Before you set any of the values on this panel, configure the JMS Server for each node by using the WebSphere Application Server ISPF Customization Dialog.

Related tasks

Managing WebSphere internal JMS servers
Use this task to manage WebSphere internal JMS servers on WebSphere Application Server Network Deployment and WebSphere Application Server Enterprise.

Related information

An overview of WebSphere asynchronous messaging using JMS

Name:

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

This name should not be changed.

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

Description:

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

This string should not be changed.

Data type	String
Default	WebSphere Internal JMS Server

Number of threads:

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

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

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

Queue Names:

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

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

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

Initial state:

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

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

Starting servers

Starting a server starts a new server process based on the process definition settings of the current server configuration. The node agent for the node on which the Application Server resides must be running before you can start the 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.

There are several options for starting an Application Server:

1. 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 command.
2. Start an Application Server using the administrative console.
   a. Click **Servers > Application Servers** from the administrative console navigation tree to go to the Application Server page.
   b. 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.
   c. Place a checkmark in the check box beside the Application Server to start and click **Start**.
   d. View the **Status** value and any messages or logs to see whether the Application Server starts.
3. Start an Application Server for tracing and debugging.

   To start the Application Server with standard Java debugging enabled:
   a. 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**.
   b. 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**.
   c. Save the changes to a configuration file.
   d. Stop the Application Server.
   e. Start the Application Server again as described previously.

Related reference

addNode command

removeNode command

serverStatus command

START nodeagent_proc_name command

START appserver_proc_name command
STOP nodeagent_proc_name command
STOP appserver_proc_name command
START dmgr_proc_name command
STOP dmgr_proc_name command

Running an Application Server with a non-root user ID and the nodeagent as root

Use this task to configure an Application Server to run as non-root.

By default, WebSphere Application Server on UNIX platforms uses the root user ID to run Application Servers. You can use a non-root user ID to run Application Servers.

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 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 not running `jmsserver`, you can use a group other than `mqm` in the following steps:

1. Log on to the Network Deployment system as root.
2. Create the `was1` user ID to be used to run the Application Server.
3. Add users `root` and `was1` to the `mqm` group.
4. Reboot the machine on a Windows platform. Log off and back on a Linux or UNIX-based platform.
5. Start the deployment manager process with the START *dmgr_proc_name* command .
6. Configure Application Server properties for the `root` and `was1` users.

   Use the administrative console to complete the following steps:

   a. Define the nodeagent to run as a root process.

      Click **System Management > 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 **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

7. Save and synchronize.
8. Stop all servers, including the server1 and jmsserver processes.

   Use the STOP *appserver_proc_name* command.
9. Stop the node agent.

   Use the STOP *nodeagent_proc_name* command.
10. As root, use operating system tools to change file permissions on Linux and UNIX-based platforms.

    The following examples assume that the WebSphere Application Server installation root directory is `/opt/WebSphere/AppServer`:

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

11. Start the node, the jmsserver, and all Application Servers.

    Start the `nodeagent` and the `jmsserver` from root. Start each Application Server from the `was1` user.
12. If running the JMS provider that WebSphere Application Server provides, verify that the MQ queue is running:

    Run the `dspmq.sh` (or `dspmq.bat`) 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 as root.

Related tasks

Running the deployment manager with a non-root user ID

"Running an Application Server and nodeagent with a non-root user ID"

"Starting servers" on page 23

Using the administrative console

Managing using command line tools

Running an Application Server and nodeagent with a non-root user ID

By default, each base Application Server node on Linux and UNIX platforms uses the root user ID to run the nodeagent process, the jmsserver process, and all Application Server processes. You can run the nodeagent, the jmsserver, and all Application Server processes under the same non-root user and user group.

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.

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
- `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 deployment manager 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. Reboot the machine on a Windows platform. Log off and back on a Linux or UNIX-based platform.
4. Start the deployment manager process with the START *dmgr_proc_name* command .
5. Define the nodeagent to run as a `wasadmin` process using the administrative console of the deployment manager.

   Click **System Management > 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

6. 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

7. If 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

8. Save and synchronize.
9. Stop all servers, including the server1 and jmsserver processes.

   Use the STOP *appserver_proc_name* command.
10. Stop the node agent.

    Use the STOP *nodeagent_proc_name* command.
11. If running the JMS provider that WebSphere Application Server provides, delete the default queue manager for the Application Server. Run the `deletemq.sh` (or `deletemq.bat`) script as root from the `/bin` directory of the installation root:

    ```
    deletemq.sh wascell wasnode jmsserver
    ```
12. If running the JMS provider that WebSphere Application Server provides, create the JMS provider queue manager and broker for the Application Server. Run the `createmq.sh` (or `createmq.bat`) script as `wasadmin` from the `/bin` directory of the installation root:

    ```
    createmq.sh /opt/WebSphere/AppServer wascell wasnode jmsserver
    ```
13. As root, use operating system tools to change file permissions on Linux and UNIX platforms:

    ```
    chgrp mqm /opt/WebSphere
    chgrp mqm /opt/WebSphere/AppServer
    chgrp -R mqm /opt/WebSphere/AppServer/config
    chgrp -R mqm /opt/WebSphere/AppServer/logs
    chgrp -R mqm /opt/WebSphere/AppServer/wstemp
    chgrp -R mqm /opt/WebSphere/AppServer/installedApps
    chgrp -R mqm /opt/WebSphere/AppServer/temp
    chgrp -R mqm /opt/WebSphere/AppServer/tranlog
    chgrp -R mqm /opt/WebSphere/AppServer/cloudscape
    ```

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

14. Log in as `wasadmin` on the Network Deployment node.
15. From wasadmin, run the START *appserver_proc_name* command to start the JMS server and all Application Servers:
16. If running the JMS provider that WebSphere Application Server provides, verify that the MQ queue is running:

    Run the `dspmq.sh` (or `dspmq.bat`) 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.

Related tasks

"Running an Application Server with a non-root user ID and the nodeagent as root" on page 24

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.
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.

Related tasks

"Managing Application Servers" on page 10

Stopping servers

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

1. Use the **stopServer** command to stop an Application Server from the command line.

2. Use the administrative console to stop an Application Server:
   a. Click **Servers > Application Servers** from the administrative console navigation tree to go to the Application Server page.
   b. Place a checkmark in the check box beside the Application Server that you want stopped and click **Stop**.
   c. Confirm that you want to stop the Application Server.
   d. View the **Status** value and any messages or logs to see whether the Application Server stops.

Related reference

STOP appserver_proc_name command

START appserver_proc_name command

Displaying the status of ARM-registered address spaces including WebSphere for z/OS servers and server instances

This section describes how to use ARM to display the status of all ARM-registered address spaces (including the address spaces of server instances) in the WebSphere for z/OS environment.

WebSphere for z/OS ships with all control regions issuing automatic restart management (ARM) registration commands. If your installation enables ARM, you should read this section.

This section describes how to use ARM to display the status of all ARM-registered address spaces (including the address spaces of server instances) in the WebSphere for z/OS environment. ARM is used to restart all address spaces that go down, if they are registered with ARM. This does not apply if the address spaces are canceled.

Each WebSphere for z/OS control region registers with ARM. If a control region terminates abnormally or the system fails, ARM will try to restart the failing address spaces. In doing this, ARM will ensure that dependent address spaces are grouped together and will start in the appropriate order. In general, the default ARM policy will restart WebSphere for z/OS in place. If using a sysplex, see the *WebSphere Application Server for z/OS V5.0: Installation and Customization*, for setup guidelines to ensure that no cross-system restarts are performed.

Steps for displaying the status of ARM-registered address spaces

Perform the following steps to use ARM to display the status of ARM registered address spaces (including the address spaces of server instances) in the WebSphere for z/OS environment:

1. Initialize all servers. The bare minimum required for a step is the cmd element. The info element is optional.
2. To display all registered address spaces (including the address spaces of server instances), issue the command: d xcf,armstatus,detail

Setting up peer restart and recovery

To allow WebSphere Application Server for z/OS to restart on an alternate system, the following prerequisites must be installed on every system (your original system as well as any systems intended for recovery) before reconfiguring the ARM policies to enable peer restart and recovery. You must also make sure all of the systems, where you might need to perform restart, are part of the same RRS log group.

- z/OS Version 1.2 or higher
- BCP APAR OA01584
- RRS APARs OA02556 and OA2556
- WebSphere Application Server for z/OS Version 5 or higher

Installing the prerequisite service updates on all of these systems will not hinder your current running environment if you want to continue to only restart in place. However, if this service is not installed, there is a possibility that the controller will not be able to move back. OTS will attempt to restart on the alternate system and fail. If there are any URs that are unresolved with RRS once this happens, the controller will not be allowed to restart on the home system until RRS is cancelled on the alternate system. For more information on OTS and RRS, see *z/OS MVS Programming: Resource Recovery*.

If you do not plan to use peer restart and recovery, you do not need to abide by these functional prerequisites. Your system will instead use the restart-in-place function.

The following products all support RRS. Individually, they also support peer restart and recovery, providing the above prerequisites are all properly installed:

- DB2 Version 7 or higher
- IMS Version 8 or higher
- CICS Version 1.3 or higher
- MQSeries Version 5.2 or higher

In addition to the preceding products, many JTA XAResource Managers can be used to assist in a WebSphere Application Server for z/OS peer restart and recovery. Consult your JTA XAResource Manager's documentation to determine if it supports restarting on an alternate system.

Prior to using peer restart and recovery:

- You must ensure that the location service Daemon and node agent are already running on all systems that might be used for recovery. Otherwise, the recovering system might attempt to recover on a system that is not running the location service Daemon and node agent. If this happens, the server will fail to start, and recovery will fail.
- You must be running in a cell with a WebSphere Application Server for z/OS datasharing configuration to utilize peer restart and recovery. You can not restart on a system that is not in datasharing with you, and you cannot restart out of place at all if you are not in datasharing with any other system. If you do not run in datasharing, your configuration is not known to the recovery system, and it can not properly execute the restart and recovery. Your only option in that case, and in the case where you are not in a cell at all, would be to restart-in-place. See *z/OS MVS Setting Up a Sysplex* for instructions on setting up a cell.

Note: Clients will see a performance impact if the systems are running at capacity. In an attempt to minimize the memory and CPU impact on the alternate system, the EJB and web containers are not restarted for servers running in peer-restart mode. This means that application servers that are in the state of being recovered will not be able to accept any inbound work.

Once you have the prerequisites installed, starting a server on a system to which it was not configured will implicitly place it into peer restart and recovery mode. If you are not using a shared HFS or if you are using a JTA XA Resource Manger, you need to perform the following steps before starting a server:

1. (Required only if you are using a non-shared HFS.) Enable non-shared HFS support. When using a non-shared HFS, the configuration settings must be replicated across the different systems in the sysplex. This is done automatically by the deployment manager and node agent. To enable this support, each node agent in your configuration must be set as a recovery node. This change is made in the administrative console:
   a. In the administrative console navigation, select **System Administration** > **Node Agents**.
   b. Select a node agent from the list. 4.
   c. Under **Additional Properties**, select **File Synchronization Service**.
   d. Under **Additional Properties**, select **Custom Properties**.
   e. Select **New**.
   f. Enter `recoveryNode` for **Name**, and `true` for **Value**. The **Description** field can be left blank.
   g. Repeat steps 3-7 for each node agent in your configuration.
   h. Save your configuration.
2. (Required only if you are using JTA XAResource Managers.) Make appropriate logs and classes are available on the alternate system If you plan to use WebSphere Application Server peer restart and recovery, and your applications access JTA XAResource Managers, you must ensure that the appropriate logs and classes are available on the alternate system.
   a. Point the WebSphere variable TRANLOG_ROOT to a shared HFS. The WebSphere variable TRANLOG_ROOT must point to a shared HFS, to which all systems in the WebSphere Application Server cell can write. The XA partner log is stored here, and the alternate system must be able to read and update this log.

      Use the administrative console to set the WebSphere variable, TRANLOG_ROOT, to the directory of a shared HFS, to which all systems in the WebSphere Application Server cell can write.

      In the administrative console, click **Environment** > **Manage WebSphere Variables**. Then click on the **TRANLOG_ROOT** variable to bring up an new window in which you can specify the directory of the shared HFS.
   b. Store the driver (i.e., JDBC Driver, JMS Provider, or JCA Resource Adapter, etc.) for each JTA XAResource Manager in an HFS that is readable by all systems in the WebSphere Application Server cell. For example, if your connector is a JDBC driver for a database, the driver would likely be stored in a read-only HFS that is accessible by all systems in the sysplex. This allows the alternate system to read the saved classpath for the resource, and reconstruct it during a restart.

      If the connector used to access a JTA XAResource Manager is not stored in an HFS that is readable by all systems that might be used for recovery, when an application server restarts on an alternate system, it will either

appear that there is no XA recovery work to do, or it will be impossible to load the classes necessary to communicate with the JTA XAResource Manager

3. Resolve InDoubt units.

   During a recovery, there will be instances when manual intervention is required to resolve InDoubt units. You will need to use RRS panels for this manual intervention.

Peer restart and recovery

The goal of every system is to have as little downtime as possible. Sometimes, however, system failures are inevitable (i.e., the power unexpectedly goes out in your main system). When this happens, a course of restart action you can take is to restart on a peer system in the sysplex, a function called peer restart and recovery. Starting a server on a system to which it was not configured will implicitly place it into peer restart and recovery mode.

When you experience a main system failure that results in InDoubt transactions with unknown outcomes, you need to obtain those intended transactional outcomes (ideally correctly) before the data can be utilized again. Peer restart and recovery provides an automated means of accomplishing this by restarting the control region on a peer system so that the ″locks″ that block the data can be dropped and the outcomes determined. This is in contrast to how a system usually handles a failure by automatically rolling back.

If a failure occurs, automatic restart management:

- Can restart WebSphere Application Server for z/OS and related servers on the same system, or
- Can use the WebSphere Application Server for z/OS peer restart and recovery function to restart related servers on an alternate system in the cell. (The application server itself is not a recoverable *resource* manager. It is a recoverable *communication* manager. It has no recoverable locks of its own and it doesn't need to manage locks nor manage lock states in a log. It just needs to make sure that both callers and callees are connected in each of the communications sessions of a distributed transaction.)

Peer restart and recovery restarts the controller on another system and goes through the transaction restart and recovery process so that we can assign outcomes to transactions that were in progress at the time of failure. During this transaction restart and recovery process, data might be temporarily inaccessible until the recovery process is complete. The restart and recovery process does not result in lost data.

Resource managers (such as DB2) that were being accessed at the time of failure may hold locks that are scoped to a transaction UR (unit of recovery). Once an outcome has been assigned to a UR, the resource managers will, generally, drop those locks.

InFlight work and presumed abort mode

If you have a distributed transaction that spans several servers, transactional locks may be held by resource managers involved in that work. When a failure occurs before that distributed transaction has started to commit, WebSphere Application Server for z/OS and the resource managers go into presumed abort mode. In this mode, the resource managers abort (rollback) the transaction.

- The effect of a server failure or communications failure will vary depending on which server is executing the work at the time of failure.

- An OTS timeout may be required to rollback the subordinate branches of the distributed transaction tree.

Example: A common case of this is when you have a server B Web client that is driving a session bean in the same server. That session bean has executed work against entity beans in servers C and D. All of the servers are involved in the same distributed, global transaction. Suddenly, server B fails while the session bean is InFlight (meaning it hadn't started to commit yet). Servers C and D are waiting for more work or the start of the two-phase commit protocol, but, while in this state, the transactional locks may still be held by the resource managers. So, the server roles are as follows:
- Server A: Servlet/JSP executed
- Server B: Session bean accessed
- Server C: Entity bean accessed
- Server D: Entity bean accessed

Once the timeout occurs, since we were InFlight at the time of the failure, we will rollback the transaction branch.

When local resource managers are involved, RRS will ensure that they are called to perform presumed abort processing. When doing recovery, RRS will work with the resource managers to ensure that the recovery is done properly. When a failure occurs while work is InFlight, RRS will direct the resource managers involved in the local UR to rollback.

The WebSphere Application Server for z/OS runtime always assumes that there is recovery to do. Every time a server comes up, it does something different depending on what mode it is in:
- If the server is running in restart/recovery mode, it checks to see whether there is any recovery required. If so, it attempts to complete the recovery and either succeeds or terminates.
- If the server is running normally, the restart/recovery transaction does not have to complete before it takes on new work. Once it knows what the restart work is, it can begin to take in new work.

Handling new work during recovery

The procedures for the recovery of InFlight and InDoubt work have been described in some detail, but how is new work handled on a recovered server? Once the InDoubt and InFlight work has been completed, the WebSphere Application Server for z/OS server shuts down. A new application server configured for that system may now be started up to accept new work.

Special considerations must be taken to begin new work on a WebSphere Application Server for z/OS using IMS Connect after recovering to an alternate system. Once the recovery has been completed, IMS Connect starts, but is not usable without some manual intervention. On the current IMS Connect WTOR perform the following commands `nn,viewhws` followed by `nn,opends` *XXX* where *XXX* is the IMS subsystem name displayed in the result of the `nn,viewhws` query. The IMS datastore needs to reflect 'active' status, which can be seen in the example below:

```
*17 HWSC0000I *IMS CONNECT READY*  IMSCONN
 R 17,VIEWHWS
 IEE600I REPLY TO 17 IS;VIEWHWS
 HWSC0001I   HWS ID=IMSCONN     Racf=N
 HWSC0001I      Maxsoc=100  Timeout=12000
 HWSC0001I    Datastore=IMS      Status=ACTIVE
 HWSC0001I      Group=IMSGROUP Member=IMSCONN
```

```
HWSC0001I      Target Member=IMSA
HWSC0001I    Port=9999     Status=ACTIVE
HWSC0001I      No active Clients
HWSC0001I    Port=LOCAL    Status=ACTIVE
HWSC0001I      No active Clients
```

Once this has been completed IMS Connect is ready for new work to be completed on the server.

When might PRR fail to recover servers

The major reason for recovery failure is if you experience a network outage while in the process of recovering. If the system cannot reach the superior or subordinate because the network is dead, communications cannot reestablish and the transaction cannot completely resolve.

When WebSphere Application Server for z/OS cannot automatically resolve all of the URs returned from RRS at restart, RRS will not allow the application server to move back to the home (original) system. If the application serve tries to go back while URs are still incomplete, you will receive an error code (C9C2186A) and a message describing an F02 return code from ATRIBRS. In order to get around this, manual resolution is required to mark the server for ″restart anywhere.″ RRS will do that once all of the URs in which WebSphere is involved are ″forgotten.″ If RRS fails to mark the server ″restart anywhere,″ the server, upon failure, is required to start on the recovery system. This is not good because it doesn't allow you to move the server back to its true home system.

The ultimate goal of this is to resolve all transactions that the application serve (the server instance- owned interests that could not complete recovery) is involved in, and then, if necessary, remove all of the the application serve interests that remain in those URs. Once that is complete, browsing the RM data log will show if the resource manager is marked ″restart anywhere.″

You **want** to see:

```
RESOURCE MANAGER=BSS00.SY1.BBOASR4A.IBM
RESOURCE MANAGER MAY RESTART ON ANY SYSTEM
```

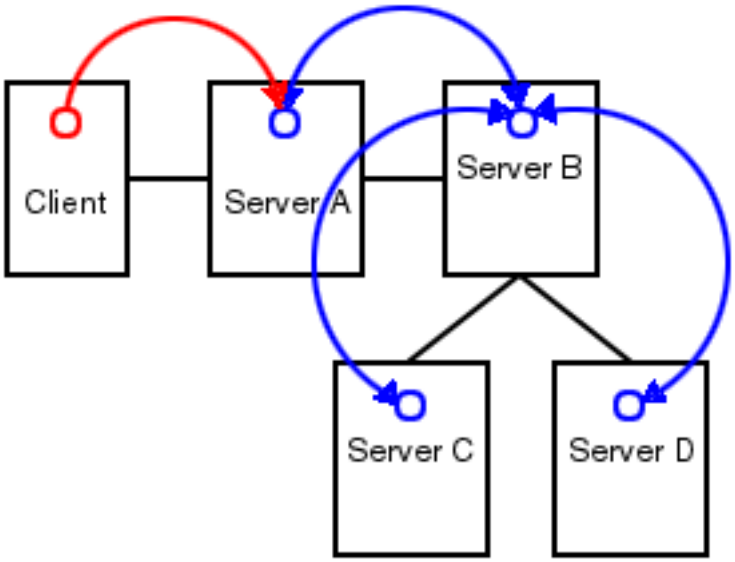
You do **not** want to see:

```
RESOURCE MANAGER=BSS00.SY2.BBOASR4A.IBM
RESOURCE MANAGER MUST RESTART ON SYSTEM SY2
```

Recoverable communication manager

WebSphere Application Server for z/OS itself is not a recoverable *resource* manager. It is a recoverable *communication* manager. This means that the application server has no recoverable locks of its own and it doesn't need to manage locks nor manage lock states in a log. Instead, the application server needs to make sure that both callers and callees are connected in each of the communications sessions of a distributed transaction.

This example outlines the organization of servers in a recoverable communication manager setup. Use the following diagram to understand the sysplex layout:



Suppose there is a client that talks to server A. Server A then talks to server B, which in turn talks to servers C and D. In this example, server A is the superior, server B is the superior subordinate, and servers C and D are subordinates. With respect to server B, there is a communication session **from** server A and one each **to** servers C and D. OTS logs each of these communication sessions as *recoverable resource references* and prepares them for recovery so that, in the event of a failure, server B can reestablish the communication sessions to these servers.

When server B fails, server A and servers C and D cannot communicate to each other since server B is the intermediary between them. So, when server B recovers, it reads the log and reestablishes the communication sessions to the other servers. Further, once connectivity is reestablished, servers B, C, and D can determine the outcome of the transaction.

Using RRS panels to resolve indoubt units of recovery

There are RRS version requirements that you must heed when using peer restart and recovery. For more information on these requirements, please see *WebSphere Application Server for z/OS V5.0: Installation and Customization*, and *z/OS MVS Programming: Resource Recovery*

If you receive the console message:

```
BBOT0015D OTS UNABLE TO RESOLVE ALL INCOMPLETE TRANSACTIONS FOR SERVER
string.  REPLY CONTINUE OR TERMINATE.
```

1. Note the server named in *string.*
2. Go to SYSPRINT (the status queue for that server) and search for messages BBOT0019 - BBOT0022 that refer to that server.
3. Follow the resulting messages.
4. RRS will not allow an operator to resolve an indoubt UR if the DSRM for that UR is active at the time, so you need to stop the server. To do this, reply "TERMINATE" to the CONTINUE/TERMINATE WTOR.

The following non-console messages, which can be used to trigger automation, provide details about daemon activities when attempting restart and recovery:

- **BBOO003E** WEBSPHERE FOR z/OS CONTROL REGION string ENDED ABNORMALLY, REASON= *hstring*.
- **BBOO009E** WEBSPHERE FOR z/OS DAEMON string ENDED ABNORMALLY, REASON= *hstring*.
- **BBOO0171I** WEBSPHERE FOR z/OS CONTROL REGION string NOT STARTING ON CONFIGURED SYSTEM *string*
- The following messages, which are written only in recovery and restart mode, provide details about transaction(s) that could not be resolved during restart and recovery:
  - **BBOT0008I** TRANSACTION SERVICE RESTART INITIATED ON SERVER *string*
  - **BBOT0009I** TRANSACTION SERVICE RESTART UR STATUS COUNTS FOR SERVER string: IN-BACKOUT= *dstring*, IN-DOUBT= *dstring*, IN-COMMIT= *dstring*
  - **BBOT0010I** TRANSACTION SERVICE RESTART AND RECOVERY ON SERVER string IS COMPLETE
  - **BBOT0011I** SERVER *string* IS COLD STARTING WITH RRS
  - **BBOT0012I** SERVER *string* IS WARM STARTING WITH RRS
  - **BBOT0013I** TRANSACTION SERVICE RESTART AND RECOVERY ON SERVER *string* IS COMPLETE. THE SERVER IS STOPPING.
  - **BBOT0014I** TRANSACTION SERVICE RECOVERY PROCESSING FOR RRS URID ' *string* ' IN SERVER *string* IS COMPLETE.
  - **BBOT0016I** TRANSACTION SERVICE RESTART AND RECOVERY FOR SERVER *string* IS NOT COMPLETE. THE SERVER IS STOPPING DUE TO OPERATOR REPLY.
  - **BBOT0017I** TRANSACTION SERVICE RESTART AND RECOVERY FOR SERVER *string* IS CONTINUING DUE TO OPERATOR REPLY.
  - **BBOT0018I** TRANSACTION SERVICE RESTART AND RECOVERY FOR SERVER *string* IS STILL PROCESSING *dstring* INCOMPLETE UNIT(S) OF RECOVERY.
  - **BBOT0019W** UNABLE TO RESOLVE THE OUTCOME OF THE TRANSACTION BRANCH DESCRIBED BY URID: ' *string* ' XID FORMATID: ' *string* ' XID GTRID: ' *string* ' XID BQUAL: ' *string* ' BECAUSE THE OTS RECOVERY COORDINATOR FOR SERVER *string* ON HOST *string*: *dstring* COULD NOT BE REACHED.
  - **BBOT0020W** UNABLE TO PROVIDE THE SUBORDINATE OTS RESOURCE IN SERVER string ON HOST *string*: *dstring* WITH THE OUTCOME OF THE TRANSACTION DESCRIBED ON THIS SERVER BY URID: ' *string* ' XID FORMATID: ' *string*' XID GTRID: ' *string* ' XID BQUAL: ' *string* ' BECAUSE THIS SERVER HAS BEEN UNABLE TO RESOLVE THE OUTCOME WITH A SUPERIOR NODE.
  - **BBOT0021W** UNABLE TO *string* THE SUBORDINATE OTS RESOURCE IN SERVER *string* ON HOST string: dstring FOR THE TRANSACTION DESCRIBED ON THIS SERVER BY URID: ' *string* ' XID FORMATID: ' *string* ' XID GTRID: ' *string* ' XID BQUAL: ' *string* ' OR ANOTHER RESOURCE INVOLVED IN THIS UNIT OF RECOVERY BECAUSE ONE OR MORE RESOURCES COULD NOT BE REACHED OR HAVE NOT YET REPLIED.
  - **BBOT0022W** UNABLE TO FORGET THE TRANSACTION WITH HEURISTIC OUTCOME DESCRIBED ON THIS SERVER BY URID: ' *string* ' XID FORMATID: '*string* ' XID GTRID: ' *string* ' XID BQUAL: ' *string* ' BECAUSE THE SUPERIOR COORDINATOR FOR SERVER *string* ON HOST *string*: *dstring* HAS NOT INVOKED FORGET ON THE REGISTERED RESOURCE.

Pay particular attention to the URID, XID FormatId, XID Gtrid and XIDBqual attributes. These pieces of information will be used to manually resolve the relevant units of work via the RRS panels.

Related concepts

Resource Recovery Services (RRS)
WebSphere Application Server for z/OS supports resource adapters that use Resource Recovery Services (RRS)to support global transaction processing. RRS is an z/OS extension to the JCA resource adapter specifications.

Resolving InDoubt units if you receive message BBOT0019W or BBOT0020W

These two messages indicate that this server could not determine the outcome from its superior. **BBOT0020W**, which describes the resource to which we could not provide an outcome, will always be accompanied by **BBOT0019W** (if you look in the server that is indicated by **BBOT0020W**, you should see a **BBOT0019W** message as well. See *z/OS MVS Programming: Resource Recovery* for more information on how to use the RRS panels and what administrative access (RACF access to the facility class, for example) is needed to resolve URs and remove interests.

Use RRS panels to perform the following steps to view the outcome of other branches in the transaction and set the outcomes of InDoubt branches to match.

1. Select option 3, ″Display/Update RRS Unit of Recovery information″ on the main RRS panel.
2. Attempt to resolve InDoubt URs with an outcome identical to the outcome of other branches of the same transaction (unless those other branches show an outcome of ″InPrepare,″ in which case you should resolve to ″InBackout.″. Complete these steps to view the outcome of other known transaction branches:
   a. Specify the XID FormatId (in decimal) in the ″Format ID″ field. The XID FormatId must be converted from hex to decimal.
   b. Specify the XID Gtrid (in hex) in the ″GTRID Pattern″ fields on the query panel. The XID Gtrid must be entered in 16 bytes to a line.
   c. Press ″enter″ to execute the query.
3. Match up the ″InDoubt″ branches. The query results obtained in step 2 above will show all URIDs involved with the specified transaction. In the query results, take note of the outcomes in the ″state″ column. If any other branches of the transaction are ″InCommit″ or ″InBackout,″ match the corresponding ″InDoubt″ branches to read the same:

   In the left-hand column labeled ″S″, enter ″c″ to commit or ″b″ to backout the UR that was identified in message **BBOT0019W**. Manually resolving the transaction can lead to mixed transaction outcomes across resource managers and servers.
4. Copy to the clipboard the URID on this panel.
5. Remove the interest for this URID

You know you are done when RRS marks the subordinate server as ″restart anywhere.″ Determine this by choosing option 1 under ″Browse and RRS log stream″ and then choosing suboption 4 under ″RRS Resource Manager Data log.″

Resolving InDoubt units if you receive message BBOT0021W

This message indicates that a server has determined the transaction outcome but has not been able to communicate it to its subordinates. When this message is

displayed, there is a possibility of an heuristic outcome. See *z/OS MVS Programming: Resource Recovery* for more information on how to use the RRS panels and what administrative access (RACF access to the facility class, for example) is needed to resolve URs and remove interests.

Perform the following steps to remove an expression of interest in this UR.

1. Select option 3, "Display/Update RRS Unit of Recovery information" on the main RRS panel.
2. To view the details of this URID, enter it in the "URID Pattern" field on the query panel. Press "enter" to execute the query. The query results should display the UR. Take note of whether the state of this UR is "InCommit," "InBackout" or "InForget." In the column labeled "S", enter "v" to display the details for this UR.
3. Remove the OTS interest. The "RRS Unit of Recovery Details" panel will open. Near the bottom of the panel will be a heading titled "Expressions of Interest" followed by one or more rows below it. These rows represent each individual expression of interest in this UR. Complete these steps to remove the OTS interest:
   a. Find the row that represents the "OTS" interest. This row will have an RM name in the form of "BSS00.xxx.yyy.IBM", where 'xxx' is the system to which the server was configured and 'yyy' is the specific server name.
   b. Type "r" in the column labeled "S" to indicate that you want to remove this interest.
   c. Press "enter" to execute the query.

The "RRS Remove Interest Confirmation" panel will open. The RM name and UR identifier fields are pre-filled. Press "enter" to confirm the removal of this interest.

From this point onward, any subordinate nodes that restart and ask this server about this UR will be unable to obtain this information. If you restart the server containing these nodes, they may be assigned an outcome different from the outcome of the transaction. You must manually resolve these nodes before you bring up the servers and start the server for which you just released the UR.

You know you are done when RRS marks the subordinate server as "restart anywhere." Determine this by choosing option 1 under "Browse and RRS log stream" and then choosing suboption 4 under "RRS Resource Manager Data log."

Resolving InDoubt units if you receive message BBOT0022W

This message indicates that the transaction outcome has been determined and communicated to the subordinate, but the subordinate resource has not yet been "forgotten." When this message is displayed, there has already been a heuristic outcome.

The subordinate will not let the UR move to the "in-forget" state until it is told to forget it. In other words, the subordinate will still be involved in a UR, so RRS will not mark the subordinate server as "restart anywhere."

See *z/OS MVS Programming: Resource Recovery* for more information on how to use the RRS panels and what administrative access (RACF access to the facility class, for example) is needed to resolve URs and remove interests.

Use RRS panels to perform the following steps to remove an expression of interest in this UR.

1. Select option 3, ″Display/Update RRS Unit of Recovery information″ on the main RRS panel.
2. To view the details of this URID, enter the URID in the ″URID Pattern″ field on the query panel. Press ″enter″ to execute the query. The query results should display the UR. In the column labeled ″s″, enter ″v″ to display the details for this UR.
3. Remove the OTS interest The ″RRS Unit of Recovery Details″ panel will open. Near the bottom of the panel will be a heading titled ″Expressions of Interest″ followed by one or more rows below it. These rows represent each individual expression of interest in this UR. Complete these steps to remove the OTS interest:
   a. Find the row that represents the ″OTS″ interest. his row will have an RM name in the form of ″BSS00.xxx.yyy.IBM″, where 'xxx' is the system to which the server was configured and 'yyy' is the specific server name.
   b. Type ″r″ in the column labeled ″S″ to indicate that you want to remove this interest.
   c. Press ″enter″ to execute the query.
4. Press ″enter″ to confirm the removal of this interest. The ″RRS Remove Interest Confirmation″ panel will open. The RM name and UR identifier fields are pre-filled. Press ″enter″ to confirm the removal of this interest.

You know you are done when RRS marks the subordinate server as ″restart anywhere.″ Determine this by choosing option 1 under ″Browse and RRS log stream″ and then choosing suboption 4 under ″RRS Resource Manager Data log.″

Resource Recovery Services Operations

Purpose

This topic provides tips for using the z/OS Resource Recovery Services with WebSphere Application Server for z/OS

Tips for RRS Operations

See .dita and *z/OS MVS Programming: Resource Recovery*, for RRS operations guidelines.

Tips for RRS operations:

- If you have configured your logstreams to the coupling facility, then monitor your log streams to ensure offload is not occurring. RRS will perform better if its recovery logs do not offload.

  Note: Proper sizing of the RRS logs is important. Too small and you get reduced throughput since logger is off-loading the logs too frequently. Too large and you could overflow your coupling facility.
- Keep the main and delayed (only contains active or live data) logs in your coupling facility. Make sure the CF definitions don't overflow.

  Note: A commit cannot occur until the log record is written.
- Until you stabilize your workloads, it is a good idea to use the archive log. If you have an archive log configured, RRS will unconditionally use it. However, there is a performance penalty for using it.

Related concepts

Resource Recovery Services (RRS)
WebSphere Application Server for z/OS supports resource adapters that use Resource Recovery Services (RRS)to support global transaction processing. RRS is an z/OS extension to the JCA resource adapter specifications.

Setting up WebSphere Application Server for z/OS on multiple systems in a sysplex

A typical WebSphere Application Server for z/OS base run-time includes a cell with a location service daemon (BBODMNB) and one node which includes an Application Server (server1) with a controller and any number of servants. Once you have installed the Application Server run-time and associated business application servers on a monoplex, you can migrate the run-time and associated application servers to a sysplex configuration.

The benefits of migrating to a sysplex include:
- You can balance the workload across multiple systems, thus providing better performance management for your applications.
- As your workload grows, you can add new systems to meet demand, thus providing a scalable solution to your processing needs.
- By replicating the run-time and associated business application servers, you provide the necessary system redundancy to assure availability for your users. Thus, in the event of a failure on one system, you have other systems available for work.
- You can upgrade the Application Server from one release or service level to another without interrupting service to your users.

The following table shows the subtasks and associated procedures for enabling WebSphere Application Server for z/OS in a sysplex.

Table 4.

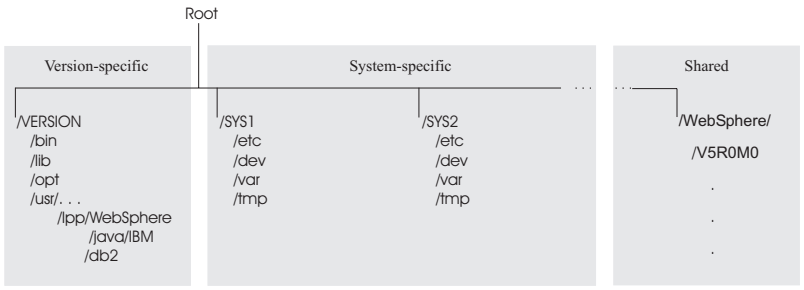
Subtask	Associated procedure (See . . .)
Setting up a sysplex	*z/OS MVS Setting Up a Sysplex*
Making decisions about the WebSphere Application Server for z/OS configuration and sysplexes	"Steps for planning WebSphere Application Server for z/OS and cells" on page 41
Preparing your security system	
Setting up data sharing	*DB2 Data Sharing: Planning and Administration*
Customizing base z/OS functions on the other systems in the sysplex	"Steps for customizing base z/OS functions on the other systems in the sysplex" on page 42
Making changes to TCP/IP	"Steps for making changes to TCP/IP" on page 44
Defining new Application Server clustered host instances in the sysplex	"Defining multiple WebSphere Application Server for z/OS systems in a sysplex" on page 44

Overview of a WebSphere Application Server for z/OS sysplex

A typical WebSphere Application Server for z/OS base run-time includes a cell with a location service daemon (BBODMNB) and one node which includes an Application Server (server1) with a controller and any number of servants.Once

you have installed the Application Server run-time and associated business application servers on a monoplex, you can migrate the run-time and associated application servers to a sysplex configuration. The benefits of migrating to a sysplex include:

- You can balance the workload across multiple systems, thus providing better performance management for your applications.
- As your workload grows, you can add new systems to meet demand, thus providing a scalable solution to your processing needs.
- By replicating the run-time and associated business application servers, you provide the necessary system redundancy to assure availability for your users. Thus, in the event of a failure on one system, you have other systems available for work.
- You can upgrade the Application Server from one release or service level to another without interrupting service to your users.



Steps for planning WebSphere Application Server for z/OS and cells

You should have completed the WebSphere Application Server for z/OS installation and customization on a monoplex or on a single system in a sysplex. Also, you must have enabled a z/OS sysplex. For more information on sysplexes, see *z/OS MVS Setting Up a Sysplex*.

Follow these steps to plan for using the Application Server and cells:

1. Decide whether you want a single-system view of the error log. If you want a single-system view of the error log, and initially you set up the error log in the system logger and used DASD for logging, you must now configure the error log in the coupling facility.
2. Decide how you will share application executables in the cell.
3. Set up ARM. This release does not support cross-system restart, so you must set up your ARM policy accordingly. Make sure you specify TARGET_SYSTEM for the system on which each element runs (if you take the default TARGET_SYSTEM=*, you get cross-system restart).
4. Decide whether you will run all the WebSphere for z/OS run-time servers on every system in the cell.

   Recommendations: The following table provides recommendations and requirements for running servers in a cell.

Table 5. Running servers in a cell

Server	Recommendations and requirements for running servers in a cell
location service daemon and node agent	• You must run both these servers on each system in the cell in which you want Application Server work to run. Thus, you may have some systems in your cell that do not run the Application Server or Application Server applications at all. But, for those systems on which you want Application Server applications to run, you must have the location service daemon and node agent. • If a server indicates that PassTickets are desirable for interaction with a client, you must run the location service daemon and node agent on the system where the z/OS client resides.
deployment manager	Make sure you follow the correct steps to configure a deployment manager cell.

5. Optional: Follow these steps to build WebSphere Application Server for z/OS Deployment Manager cells:
   a. Install the base server on each node in your sysplex.
   b. Install a deployment manager cell on one node in your sysplex.
   c. Add base server nodes to the deployment manager cell.

Steps for customizing base z/OS functions on the other systems in the sysplex

You must have the WebSphere Application Server for z/OS product code installed through SMP/E and have created copies of the product sample files.

Repeat the same customizations to base z/OS functions that you did for your initial installation and customization of the Application Server. The steps are repeated here for convenience.

Note: The following steps assume that your default Application Server data set high-level qualifier (hlq) is "BBO". If it is not, modify the examples to use your specified hlq.

Perform the following steps to change the base system:

1. Change SCHEDxx to include the statements from the BBOSCHED sample file you ran in the customization dialog.
2. APF-authorize the BBO.SBBOLOAD, BBO.SBBOLD2, and BBO.SBBOLPA data sets.

   Example: Your PROGxx PARMLIB member could include:

```
APF FORMAT(DYNAMIC)
/*********************************************************************/
/* BOSS LOCAL DATASETS                                               */
/*********************************************************************/
APF ADD
    DSNAME(BBO.SBBOLOAD)
    VOLUME(vvvvvv)
APF ADD
    DSNAME(BBO.SBBOLD2)
```

```
    VOLUME(vvvvvv)
APF ADD
    DSNAME(BBO.SBBOLPA)
    VOLUME(vvvvvv)
```

where vvvvvv is your volume identifier.

3. Ensure that the Language Environment data set, SCEERUN, and the DB2 data set, SDSNLOAD, are authorized.
4. Do **not** APF-authorize BBO.SBBOULIB or BBO. SBBOMIG, because they should run under the authority of the client user.
5. Place Application Server modules. Use the following table to place Application Server modules:

Table 6. Placing modules in LPA or link list

Modules	Notes
BBO.SBBOLPA	Load all members into the LPA.
BBO.SBBOLOAD	We recommend you dynamically load all members into the LPA. If your virtual storage is constrained, place the members in the link list.
BBO.SBBOMIG	You can put members into the link list or LPA.
BBO.SBBOLD2	Do **not** put members from SBBOLD2 in the LPA. Place these members in the link list.
BBO.SBBOULIB	Do **not** place these members in **either** the LPA or link list.

Rule: These data sets are PDSEs and cannot be added to members in LPALSTxx or IEALPAxx.

Recommendation: For automation, if you want to ensure the Application Server modules are loaded into dynamic LPA and available after an IPL, create a new PROGxx member with the SETPROG LPA commands and invoke the PROGxx member from PARMLIB COMMNDxx.

Example:

```
SETPROG LPA,ADD,MASK=*,DSNAME=
BBO.
SBBOLOAD
SETPROG LPA,ADD,MASK=*,DSNAME=
BBO.
SBBOLPA
```

- Change "BBO" if it is not the high-level qualifier for your Application Server data sets.
- If using SETPROG on a running system, be sure to purge modules with the same name as those from BBO.SBBOLPA, BBO.SBBOLOAD, or BBO.SBBOMIG that are already in the LPA.

6. Optional: If you used a PROGxx file for APF authorizations or the LPA, be sure to issue:

   ```
   SET PROG=xx
   ```

7. Optional: Make sure all the BBO.* data sets are cataloged. While not required, this is highly recommended.
8. Update your SYS1.PARMLIB(BLSCUSER) member with the IPCS models supplied by member BBOIPCSP. For details in BLSCUSER, see *z/OS MVS IPCS User's Guide*.
9. Optional: Start SMF recording. If you want to start SMF recording to collect system and job-related information on the WebSphere Application Server system:
   a. Edit the SMFPRMxx parmlib member.

1) Insert an 'ACTIVE' statement to indicate SMF recording.
2) Insert a SYS statement to indicate the types of SMF records you want the system to create.

   Example: Use SYS(TYPE(120:120)) to select type 120 records only. Keep the number of selected record types small, to minimize the performance impact.

b. To start writing records to DASD, issue the following command:

```
t smf=xx
```

Where xx is the suffix of the SMF parmlib member (SMFPRMxx). For more information about the SMF parmlib member, see *z/OS MVS System Management Facilities (SMF)*.

When you activate writing to DASD, the data is recorded in a data set (specified in SMFPRMxx).

Steps for making changes to TCP/IP

You must have TCP/IP installed and configured.

1. Change DNS entries. Assuming you use an implementation of the DNS that allows use of generic IP names that dynamically resolve to like-configured servers, you must adjust the IP names in your DNS. Keep the generic IP name of the location service daemon, but add a new IP name for the second and subsequent location service daemon servers. This is important not only for workload balancing, but in the event of a server failure: the DNS can direct work to other servers.
2. In the TCP/IP profile for each additional system in the cell, add a port for the location service daemon and associate it with a new location service daemon server name.

   By default, WebSphere for z/OS uses port 5655 for the location service daemon. Also, WebSphere for z/OS names the first location service daemon server DAEMON01 and increments the suffix on that name for each new location service daemon server (DAEMON02, DAEMON03, and so forth). Thus, on your second system in the cell, add a port and associate it with DAEMON02.

   Example:

```
5655
   TCP     DAEMON02
```

   Follow the same pattern for your third and subsequent systems in the cell.

Defining multiple WebSphere Application Server for z/OS systems in a sysplex

You must have your initial WebSphere Application Server for z/OS system installed and running. If not, start RRS.

Follow these steps to define the second WebSphere Application Server for z/OS system:

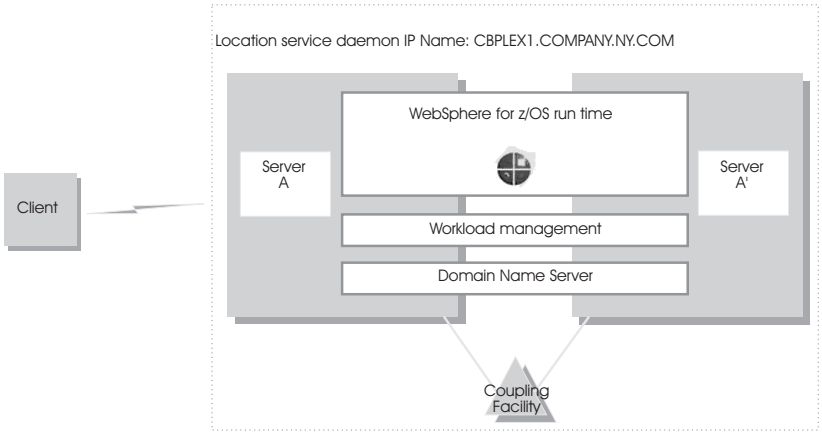
Use the Administrative Console to define additional systems in the sysplex with their servers. We assume you have already created the first WebSphere Application Server for z/OS system with an application server called BBOASR2, BBOC001, or both (the application servers used for the Installation Verification Test).

We provide instructions for defining the second system. Follow the same pattern of steps for the third and subsequent systems.

This procedure explains how to use the Administrative Console to create a second WebSphere Application Server for z/OS run-time system.

1. Log onto the Administrative Console.
2. Define a second system in the sysplex. The run-time servers are defined automatically for you.
3. Check the WebSphere variables for each run-time sever instance on the second system. The WebSphere variables are defined hierarchically in the following order: sysplex, server, node, then cluster. A WebSphere variable lower in the hierarchy overrides a matching one higher in the hierarchy. Check the WebSphere variables for the following servers. Some WebSphere variables are common for all systems in the sysplex, while others are unique for each system.
4. Specify start procedures to be used by the location service daemon to start the node agent and deployment manager servers (controllers) on the second system. After you start the location service daemon, it starts these server controllers automatically.

Connection optimization



Characteristics of a configuration in which the Domain Name Server cooperates with workload management (WLM) to route client requests throughout a cell are:

- The domain name server (DNS) is replicated by setting up a secondary DNS on more than one system in the cell.
- The client needs to know the location service daemon IP Name in order to connect to WebSphere for z/OS.
- Each system in the cell has the same location service daemon IP Name and Resolve IP Name. Workload management and the domain name server determine the actual system to which client requests go. The client sees the cell as a single system, though its requests may be balanced across systems in the cell.
- As part of workload balancing and maximizing performance goals, workload management also routes work requests to systems in the cell. This function is possible because WebSphere for z/OS cooperates with workload management. Because the system references that a client sees are indirect, even requests from that same client may be answered by differing systems in the cell.
- The implication for clients is that they should not cache IP addresses unless they can recover from failed connections. That is, if a connection fails, a client

should be able to reissue a request, but, because the IP address is an indirect address, a reissue of the request can be answered by another system in the cell.

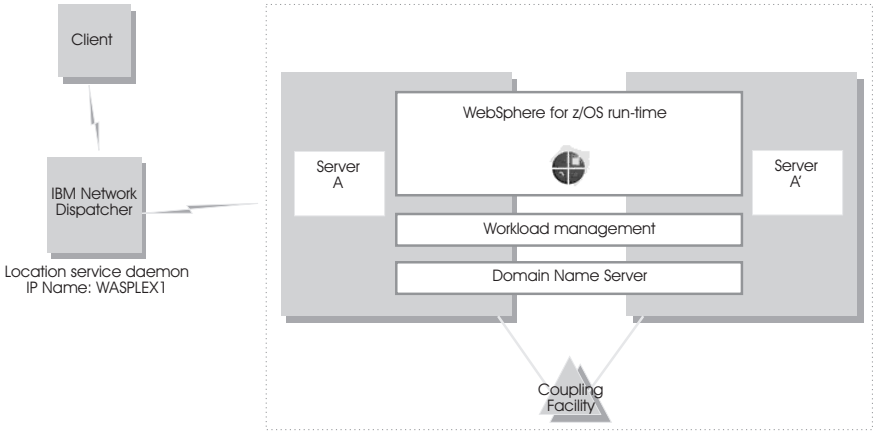
For additional details on setting up servers for connection optimization, see *z/OS Communications Server: IP Configuration Reference*.

IBM Network Dispatcher

The IBM Network Dispatcher is a router that handles network requests for the cell.

Characteristics of such a configuration are:

- The location service daemon IP Name is associated with the IP address of the router.
- The IBM Network Dispatcher cooperates with workload management to route requests through the cell. The client never sees a change in IP addresses.
- The implication for clients is that they can cache the IP addresses, because this configuration does not change them dynamically.



Bind-specific support in WebSphere Application Server for z/OS

Bind-specific support in WebSphere Application Server for z/OS allows you to control the use of TCP/IP resources in WebSphere for z/OS. This support allows you to have the Application Server ORB and other products and applications on the same z/OS system without requiring the client code to configure unique ports. In other words, this support allows use of port 2809 by the Application Server and other products and applications on the same system. This support allows the utilization of multiple TCP/IP stacks (Common INET) by the WebSphere for z/OS ORB and the use of multiple IP addresses on the same TCP/IP stack.

To use bind-specific support, use the protocol_iiop_listenIPAddress WebSphere variable, which specifies the IP address in dotted decimal format. WebSphere Application Server for z/OS servers listen for client connection requests on this IP address.

Because a given IP address is associated with a given TCP/IP stack, you could specify the protocol_iiop_listenIPAddress variable in the environment file so that a WebSphere Application Server for z/OS server uses a specific TCP/IP stack.

In addition, because you can define multiple IP addresses for a given TCP/IP stack, the Application Server port 2809 servers could share the same TCP/IP stack with other products and applications requiring port 2809, because you made their IP addresses unique with protocol_iiop_listenIPAddress.

Alternatively, you can, without the use of bind-specific support, define alternate ports for port 2809 and the location service daemon, which are the only values defined by the CORBA standard. However it is not clear that all client ORBs will easily support configuring the Application Server port to something other than 2809. Configure the ports for the location service daemon and node by specifying port numbers on the protocol_iiop_daemon_port WebSphere variable.

For details on WebSphere variables, see the Administrative Console or the InfoCenter.

For more information about multiple TCP/IP stacks (Common INET), see *z/OS UNIX System Services Planning*. For more information about multiple IP addresses on the same TCP/IP stack, see *z/OS Communications Server: IP Configuration Reference*.

Transports

A transport is the request queue between a WebSphere Application Server plug-in for Web servers and a Web container in which the Web modules of an application reside. When a user at a Web browser requests an application, the request is passed to the Web server, then along the transport to the Web container.

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

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

The internal transport

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

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

At times, you might be able to configure the internal transport to use ports other than 9080 and 9443. The transport configuration is a part of the Web container configuration. To change the port number, you must adjust your virtual host alias and what you type into the Web browser.

Configuring transports

You configure transports to specify:

- How to manage a set of connections. For example, to specify the number of concurrent requests to allow.
- Whether to secure the connections with SSL
- Host and IP information for the transport participants

Note: Only one HTTP transport and one HTTPS transport can be defined per application server.

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**.
3. Regenerate the WebSphere plug-in for the Web server.

If the Web server is located on a machine remote from the application server, you might also need to perform special configuration tasks to redirect application requests from the Web server machine to the application server machine.

Related concepts

"Transports" on page 47

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**.

Related concepts

"Transports" on page 47

Related tasks

"Configuring transports" on page 47

Related reference

Administrative console buttons
The following button choices are available on various pages of the administrative console, depending on which product features you have enabled.

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

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

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

"HTTP transport settings"

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*.

Related reference

Administrative console buttons
The following button choices are available on various pages of the administrative console, depending on which product features you have enabled.

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

"Custom property collection" on page 17

Secure Sockets Layer configuration repertoire settings
Use this page to define a new Secure Sockets Layer (SSL) alias. Through the SSL configuration repertoire, administrators can define any number of SSL settings to use in configuring the Hypertext Transfer Protocol with SSL (HTTPS), Internet InterORB Protocol with SSL (IIOPS) or Lightweight Directory Access Protocol with SSL (LDAPS) connections. You can pick one of the SSL settings defined here from any location within the administrative console that supports SSL connections. This simplifies the SSL configuration process because you can reuse many of these SSL configurations by specifying the alias in multiple places.

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

Example: Setting custom properties for an HTTP transport

WebSphere Application Server has several transport properties that are not shown in the settings page for an HTTP transport. They are as follows:

ConnectionIOTimeout
: Specifies the maximum number of seconds to wait when trying to read or process data during a request. Data type: Integer. Default: 120 seconds.

  If you specify a value for this property, the same value will be used for the wait time for both reading and processing data during a request.

ConnectionKeepAliveTimeout
: Specifies the maximum number of seconds to wait for the next request on a keep alive connection. Data type: Integer. Default: 30 seconds.

ConnectionResponseTimeout
: Specifies the maximum number of seconds to wait when trying to read or write data during a response. Data type: Integer. Default: 300 seconds.

MaxKeepAliveRequests
: Specifies the maximum number of requests which can be processed on a single keep alive connection.This parameter can help prevent denial of service attacks when a client tries to hold on to a keep-alive connection. The Web server plug-in keeps connections open to the application server as long as it can, providing optimum performance. Data type: Integer. Default value: 50 requests.

MutualAuthCBindCheck
: Specifies whether or not a client certificate should be resolved to a SAF principal. Default is `false`. If this property is set to `true`, all SSL connections from a browser must have a client certificate, and the user ID associated with that client certificate must have RACF CONTROL authority for CB.BIND.servername. If these conditions are not met, the connection will be closed. Issue the following RACF command to give the user ID associated with that client certificate RACF CONTROL authority:

  ```
  PERMIT CB.BIND.servername CLASS(CBIND) ID(clientCertUserid) ACCESS(CONTROL)
  ```

protocol_http_large_data_inbound_buffer
: Specifies, in bytes, the length of a serially reusable inbound buffer that is used for HTTP requests that are larger than 10 megabytes. Inbound HTTP

requests that are larger than 10 megabytes are rejected. The default is `0 (zero)` which indicates that no buffer is needed and requests that are larger than 10 megabytes are rejected.

TrustedProxy
Specifies that Private Headers received from a WebSphere plug-in for Web servers are to be trusted. Default is `false`.

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:

Steps for this task

1. Access the settings page for transport properties:
   a. In the console navigation tree, click **Servers > Application Servers >***server_name* **> Web Container > HTTP Transport**
   b. Click on the **HOST** whose properties you want to set.
   c. Under **Additional Properties** select **Custom Properties**.
   d. On the Custom Properties page, click **New**.
2. Click **Save** on the console taskbar and save the changes to the configuration.
3. Restart the server.
4. Regenerate the Web server plug-in.
5. On the settings page for a new property, enter the name of the transport property and the value to which you want it set. For example, if you want the transport to wait a maximum of 60 seconds when trying to read or write data during a request, enter `ConnectionIOTimeout` for name and `60` for value. Then click **OK**.

Related tasks

"Configuring transports" on page 47

Modifying the default Web container configuration

Related reference

Tuning parameter index

z/OS port assignments

Purpose

The following table lists the default server values for WebSphere Application Server for z/OS V5.

z/OS port assignments

The following table lists the z/OS port assignments.

Port	Base location service daemon	Base Application Server	ND location service daemon	ND Application Server	Node Agent	JMS Server	Deploy. Manager
HTTP		9080		9080			9090
HTTP/S		9443		9443			9043

Port	Base location service daemon	Base Application Server	ND location service daemon	ND Application Server	Node Agent	JMS Server	Deploy. Manager
Bootstrap		2809		9810	2809	2810	9809
ORB	5655	2809	5755	9810	2809	2810	9809
ORB SSL	5656	0	5756	0	0	0	0
SOAP/JMX		8880		8880	8878	8876	8879
DRS		7873		7873	7888		7989
JMS Queued		5558				5558	
JMS Direct		5559				5559	
Node Discovery					7272		
Node Multi-cast Discovery					5000		
Cell Discovery							7277

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.

Related tasks

"Developing custom services"

Developing custom services

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

1. Develop a custom service class that implements the com.ibm.websphere.runtime.CustomService interface. The properties passed by the application server run-time to the initialize method can include one for an external file containing configuration information for the service (retrieved with externalConfigURLKey). In addition, the properties can contain any name-value pairs that are stored for the service, along with the other system administration configuration data for the service. The properties are passed to the initialize method of the service as a Properties object.

   There is a shutdown method for the interface as well. Both methods of the interface declare that they may throw an exception, although no specific exception subclass is defined. If an exception is thrown, the run-time logs it, disables the custom service, and proceeds with starting the server.
2. On the Custom Service page of the administrative console, click **New**. Then, on the settings page for a custom service instance, create a custom service configuration for an existing application server, supplying the name of the class

implemented. If your custom services class requires a configuration file, specify the fully-qualified path name to that configuration file in the **externalConfigURL** field. This file name is passed into your custom service class.

3. Stop the application server and then restart the server.
4. Check the application server to ensure that the initialize method of the custom service ran as intended. Also ensure that the shutdown method performs as intended when the server stops.

As mentioned above, your custom services class must implement the CustomService interface. In addition, your class must implement the initialize and shutdown methods. Suppose the name of the class that implements your custom service is *ServerInit*, your code would declare this class as shown below. The code below assumes that your custom services class needs a configuration file. It shows how to process the input parameter in order to get the configuration file. If your class does not require a configuration file, the code that processes configProperties is not needed.

```
public class ServerInit implements CustomService
{
/**
* The initialize method is called by the application server run-time when the
* server starts. The Properties object passed to this method must contain all
* configuration information necessary for this service to initialize properly.
*
* @param configProperties java.util.Properties
*/
    static final java.lang.String externalConfigURLKey =
       "com.ibm.websphere.runtime.CustomService.externalConfigURLKey";

    static String ConfigFileName="";

    public void initialize(java.util.Properties configProperties) throws Exception
    {
        if (configProperties.getProperty(externalConfigURLKey) != null)
        {
           ConfigFileName = configProperties.getProperty(externalConfigURLKey);
        }

       // Implement rest of initialize method
    }
/**
* The shutdown method is called by the application server run-time when the
* server begins its shutdown processing.
*
* @param configProperties java.util.Properties
*/
    public void shutdown() throws Exception
    {
        // Implement shutdown method
    }
```

Related concepts

"Custom services" on page 52

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**.

Related concepts

"Custom services" on page 52

Related tasks

"Developing custom services" on page 52

Related reference

Administrative console buttons
The following button choices are available on various pages of the administrative console, depending on which product features you have enabled.

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

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

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

Related information

../javadoc/ae/com/ibm/websphere/management/configservice/ConfigService.html

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*.

Related concepts

"Custom services" on page 52

Related tasks

"Developing custom services" on page 52

Related reference

Administrative console buttons
The following button choices are available on various pages of the administrative console, depending on which product features you have enabled.

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

Related information

../javadoc/ae/com/ibm/websphere/management/configservice/ConfigService.html

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

Related tasks

"Defining application server processes"

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 defintion 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 defintion, 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 monitoring policies to track the performance of a process.
4. Specify name-value pairs for properties needed by the process definition.
5. Stop the application server and then restart the server.
6. Check the application server to ensure that the process definition runs and operates as intended.

Related concepts

"Process definition"

Process definition settings

Use this page to view or change settings for a process definition, which provides command-line information for starting, initializing, or stopping a process.

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

Each of the commands that follows can be used for the control process. The servant process uses only the Start command and Start Command Args. Specify the commands for the control process on one process definition panel and the commands for the servant process on another process definition panel. Do not put the commands for the two different processes on the same panel.

Related concepts

"Process definition"

Related tasks

"Defining application server processes"

Related reference

Administrative console buttons
The following button choices are available on various pages of the administrative console, depending on which product features you have enabled.

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

"Java virtual machine settings" on page 63

"Custom property collection" on page 17

Start Command

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

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

Servant process

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

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

Start Command Args

Specifies any additional arguments required by the start command.
Control process

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

Servant process

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

Stop Command

Specifies the platform-specific command to stop the server process
Specify two commands in the field, one for the Stop command and one for the Immediate Stop (CANCEL) command.

Data type	String
Format	STOP *server short name*;CANCEL *server short name*
Example	STOP BBOS001;CANCEL BBOS001

Stop Command Args

Specifies any additional arguments required by the stop command.
Specify arguments for the Stop command and the Immediate Stop (CANCEL) command.

Data type	String
Format	*stop command arg string;immediate stop command arg string*
Example	;ARMRESTART **Note:** In this example, Stop has no arguments. Immediate Stop has the argument ARMRESTART. A semicolon precedes ARMRESTART.

Terminate Command

Specifies the platform-specific command to terminate the server process

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

Terminate Command Args

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

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

Executable Name

Specifies the executable name of the process.

Data type	String

Executable Arguments

Specifies executable commands that run when the process starts.
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 in which the process will run.
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

Monitoring policy settings

Use this page to view or change settings that control how the node agent monitors and restarts a process.

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

Related reference

Administrative console buttons
The following button choices are available on various pages of the administrative console, depending on which product features you have enabled.

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

Maximum Startup Attempts:

Specifies the maximum number of times to attempt to start the application server before giving up.

Data type	Integer

Ping Interval:

Specifies the frequency of communication attempts between the parent process, such as the node agent, and the process it has spawned, such as an application server. Adjust this value based on your requirements for restarting failed servers. Decreasing the value detects failures sooner; increasing the value reduces the frequency of pings, reducing system overhead.

Data type	Integer

Ping Timeout:

When a parent process is spawning a child process, such as when a process manager spawns a server, the parent process pings the child process to see whether the child was spawned successfully. This value specifies the number of seconds that the parent process should wait (after pinging the child process) before assuming that the child process failed.

Data type	Integer
Units	Seconds

Automatic Restart:

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

Data type	Boolean
Default	true

Node Restart State:

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

STOPPED.

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

Process definition type settings

Use this page to view or change settings for a process definition type
To view this administrative console page, click **Servers > Application Servers >***server_name* **> Process Definition**.

Related reference

Administrative console buttons
The following button choices are available on various pages of the administrative console, depending on which product features you have enabled.

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

Control:

Specifies the process definitions for the control process

Servant:

Specifies the process definitions for the servant process.

Sysplex Distributor

The IBM-recommended implementation if you are running in a sysplex is to set up your TCP/IP network with Sysplex Distributor. This makes use of dynamic virtual IP addresses (DVIPAs), which increase availability and aid in workload balancing.

The following are recommended environment considerations for Sysplex Distributor:

- You need only basic sysplex functionality to utilize DVIPAs and Sysplex Distributor because these functions do not rely on data stored permanently in the coupling facility.
- Set up your system such that each HTTP request connection results in no saved state or the HTTP and Application Servers are configured to share a persistent state.

When going this route, HTTP server plug-ins send no-affinity connections to Sysplex Distributor (a secondary connection load balancer) with more information to make a better distribution decision.

Note: As long as the HTTP catcher itself is not bound to any particular IP address, the application-specific DVIPA can be used when affinities dictate a particular server. This allows use of the Sysplex Distributor server address for requests that are not tied to a server, covering the same set of servers in the sysplex.

Since the client connection terminates at the plug-in/proxy and the secondary connection is established by the plug-in itself, there is no need for network address translation.

Requests to the node agent do not require any affinity, and each request is independent of other requests. Sysplex Distributor can be used to balance work requests among node agents, with the added benefit that Sysplex Distributor knows which nodes are available. Therefore, it will never route a work request to a node that is not listening for new connection requests.

Note: If you are running z/OS 1.2 or earlier, Sysplex Distributor is limited to distribution on only four ports for a particular distributed DVIPA. You may configure multiple DVIPAs when more than four ports exist, but this is a configuration burden.

Multiple TCP/IP stacks

When configuring WebSphere Application Server for z/OS on a system with multiple stacks, you must first establish the Application Server's stack affinity to the desired stack so that all socket communications are bound to that stack, and then you establish the Application Server's allocation of the proper host name resolution configuration data sets so that host name lookups have the desired results.

You may want to run multiple TCP/IP stacks on the same system to provide network isolation for one or more of your applications. For instance, you may have multiple OSA Features, each one connecting your system to a different network. You may assign a TCP/IP stack to each one. To do so, use the common INET physical file system (C_INET PFS). This physical file system allow you to configure multiple physical file systems (network sockets) and make them active concurrently. First, specify common INET through the NETWORK DOMAINNAME parameter of SYS1.PARMLIB(BPXPRMxx). Second, if you plan to configure WebSphere Application Server for z/OS to use a non-default TCP/IP stack, consult *z/OS UNIX System Services Planning*, and *z/OS Communications Server: IP Configuration Reference*, for details.

Note: Should you opt not to run multiple IP stacks, be sure to set the jvm property `-Dcom.ibm.websphere.singlehost=1` in each controller and servant JVM. You can do this be accessing the modifying the jvm configuration section of the administrative console, which can be accessed by following this path through the console **servers >*servername*> process definitions > controller > jvm configuration and servers > *servername* > process definitions > servant > jvm configuration**.

1. Establish the Application Server's host name resolution configuration data set.
   a. Set the RESOLVER_CONFIG environment variable to the desired data set name.
   b. Place the RESOLVER_CONFIG environment variable in the current.env file for each server.
   c. Export the RESOLVER_CONFIG environment variable in client shell scripts.
   - You can also use JCL to specify the name resolution configuration data set. To use JCL, add //SYSTCPD DD DSN=some.tcpip.DATA,DISP=SHR to the server JCL. The RESOLVER_CONFIG environment variable overrides the SYSTCPD DD statement.

   See *z/OS Communications Server: IP Configuration Reference*, for more information on the RESOLVER_CONFIG environment variable.
2.
3. Establish the Application Server's stack affinity to the desired stack

4. Then do this.
   a. Set the BPXK_SETIBMOPT_TRANSPORT environment variable to the value of the desired transport.
   b. Place the BPXK_SETIBMOPT_TRANSPORT environment variable in the current.env file for each server.
   c. Export the BPXK_SETIBMOPT_TRANSPORT environment variable in client shell scripts.

   See *z/OS UNIX System Services Planning*, for more information on the BPXK_SETIBMOPT_TRANSPORT environment variable.

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.

Related tasks

"Using the JVM"

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 8" provides an example that involves specifying a value for the **Generic JVM Arguments** property on the Java Virtual Machine page to enable UTF-8 encoding on an application server. Enabling UTF-8 allows multiple language encoding support to be used in the administrative console.

"“Example: Configuring JVM sendRedirect calls to use context root” on page 66" provides an example that involves defining a property for the JVM.

Related concepts

"Java virtual machines (JVMs)"

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**.

Related concepts

"Java virtual machines (JVMs)" on page 62

Related tasks

"Using the JVM" on page 62

Tuning performance

Related reference

Administrative console buttons
The following button choices are available on various pages of the administrative console, depending on which product features you have enabled.

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

"Custom property collection" on page 17

Tuning parameter index

Classpath

Specifies the standard class path in which the Java virtual machine code looks for classes.

Enter each classpath entry into a table row. You do not need to add the colon or semicolon at the end of each entry.

Data type	String
Units	Class path

Boot Classpath

Specifies bootstrap classes and resources for JVM code. This option is only available for JVM instructions that support bootstrap classes and resources. You can separate multiple paths by a colon (:) or semi-colon (;), depending on operating system of the node.

Data type	String

Verbose Class Loading

Specifies whether to use verbose debug output for class loading. The default is not to enable verbose class loading.

Data type	Boolean
Default	false

Verbose Garbage Collection

Specifies whether to use verbose debug output for garbage collection. The default is not to enable verbose garbage collection.

Data type	Boolean
Default	false

Verbose JNI

Specifies whether to use verbose debug output for native method invocation. The default is not to enable verbose Java Native Interface (JNI) activity.

Data type	Boolean
Default	false

Initial Heap Size

Specifies the initial heap size available to the JVM code, in megabytes.
Increasing the minimum heap size can improve startup. The number of garbage collection occurrences are reduced and a 10% gain in performance is realized.

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

Data type	Integer
Default	64 for OS/400, 50 for all other platforms

Maximum Heap Size

Specifies the maximum heap size available to the JVM code, in megabytes.
Increasing the heap size can improve startup. 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. Therefore, set the maximum heap size low enough to contain the heap within physical memory.

Data type	Integer
Default	0 for OS/400, 256 for all other platforms. Keep the value low enough to avoid paging or swapping-out-memory-to-disk.

Run HProf

Specifies whether to use HProf profiler support. To use another profiler, specify the custom profiler settings using the HProf Arguments setting. The default is not to enable HProf profiler support.
If you set the Run HProf property to true, then you must specify command-line profiler arguments as values for the HProf Arguments property.

Data type	Boolean
Default	false

HProf Arguments

Specifies command-line profiler arguments to pass to the JVM code that starts the application server process. You can specify arguments when HProf profiler support is enabled.
HProf arguments are only required if the Run HProf property is set to true.

Data type	String

Debug Mode

Specifies whether to run the JVM in debug mode. The default is not to enable debug mode support.

If you set the Debug Mode property to true, then you must specify command-line debug arguments as values for the Debug Arguments property.

Data type	Boolean
Default	false

Debug Arguments

Specifies command-line debug arguments to pass to the JVM code that starts the application server process. You can specify arguments when Debug Mode is enabled.

Debug arguments are only required if the Debug Mode property is set to true.

Data type	String
Units	Java command-line arguments

Generic JVM Arguments

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

The following are optional command line arguments that you can use by entering them into the **General JVM Arguments** field:

- **-Xquickstart:** You can use this value for initial compilation at a lower optimization level than in default mode, and later, depending on sampling results, you can recompile to the level of the initial compile in default mode. Use quickstart for applications where early moderate speed is more important than longrun throughput. In some debug scenarios, test harnesses and short-running tools, it is possible to realize startup time gains between 15-20%. **-DCOPT_NQREACHDEF** can improve startup by an additional 15%.
- **-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%.
- **-Xnoclassgc:** You can use this value to disable class garbage collection, making class reuse more available, and slightly improving performance. Class garbage collection is enabled by default, but it is recommended that you enable it. You can monitor garbage collection using the verbose:gc configuration setting because its output includes class garbage collection statistics.
- **-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*. It is recommended that you use parallel garbage collection if your machine has more than one processor. This argument applies only to the IBM Developer Kit.
- **-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. When entering this value in the **Generic JVM Arguments** field, also enter the number of processors that your machine has, for example, `-Xnocompactgc=`*number_of_processors*.
- **-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 application helps you include them in the calculation.

- **-Xmc:** The thread local heap size is a portion of the heap that is allocated exclusively for a thread. Because of the thread local heap size, the thread does not need to lock the entire heap when allocating objects. However, when the thread local heap is full, object allocation is done from the heap that needs synchronization. A good local cache size is critical to performance and requires knowledge of the application and its objects.
- **-Xml:** You can use this value to set the limit of an object size to allocate from the local cache. Objects that exceed the limit size need allocating in the regular heap. Allocate objects from the local cache as much as possible or the local cache depletes because it does not grow dynamically. If you know some objects are going to be very large, allocate them from the regular heap.

Data type	String
Units	Java command line arguments

Executable JAR File Name

Specifies a full path name for an executable JAR file that the JVM code uses.

Data type	String
Units	Path name

Disable JIT

Specifies whether to disable the just in time (JIT) compiler option of the JVM code. If you disable the JIT compiler, throughput decreases noticeably. Therefore, for performance reasons, keep JIT enabled.

Data type	Boolean
Default	false (JIT enabled)
Recommended	JIT enabled

Operating System Name

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

Data type	String

Example: Configuring JVM sendRedirect calls to use context root

If the com.ibm.websphere.sendredirect.compatibility property is not set and your application servlet code has statements such as *sendRedirect("/home.html")*, your Web browser might display messages such as *Error 404: No target servlet configured for uri: /home.html*. To instruct the server not to use the Web server's document root and to use instead the Web application's context root for sendRedirect() calls, configure the JVM by setting the com.ibm.websphere.sendredirect.compatibility property to a `true` or `false` value.

1. Access the settings page for a property of the JVM.
   a. Click **Servers > Application Servers** in the console navigation tree.
   b. On the Application Server page, click on the name of the server whose JVM settings you want to configure.

c. On the settings page for the selected application server, click **Process Definition**.
d. On the Process Definition page, click **Java Virtual Machine**.
e. On the Java Virtual Machine page, click **Custom Properties**.
f. On the Properties page, click **New**.

2. On the settings page for a property, specify a name of `com.ibm.websphere.sendredirect.compatibility` and either `true` or `false` for the value, then click **OK**.
3. Click **Save** on the console taskbar.
4. Stop the application server and then restart the application server

Related tasks

"Using the JVM" on page 62

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, **Application Server** >*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

Specifies whether or not multihome support is enabled for the Discovery and SOAP ports. If the property is set to false, then WebSphere Application Server will only listen on the configured host name for Discovery and SOAP messages.

The default value is true. 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	**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. .

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 the **verbose:gc**JVM configuration setting. The **verbose:gc** format is not standardized between different JVMs or release levels.

For this type of investigation, set the minimum and maximum heap sizes to the same value. Choose a representative, repetitive workload that matches production usage as closely as possible, user errors included.

To ensure meaningful statistics, run the fixed workload until the application state is steady. It usually takes several minutes to reach a steady state.

Detecting over-utilization of objects

You can 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.

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

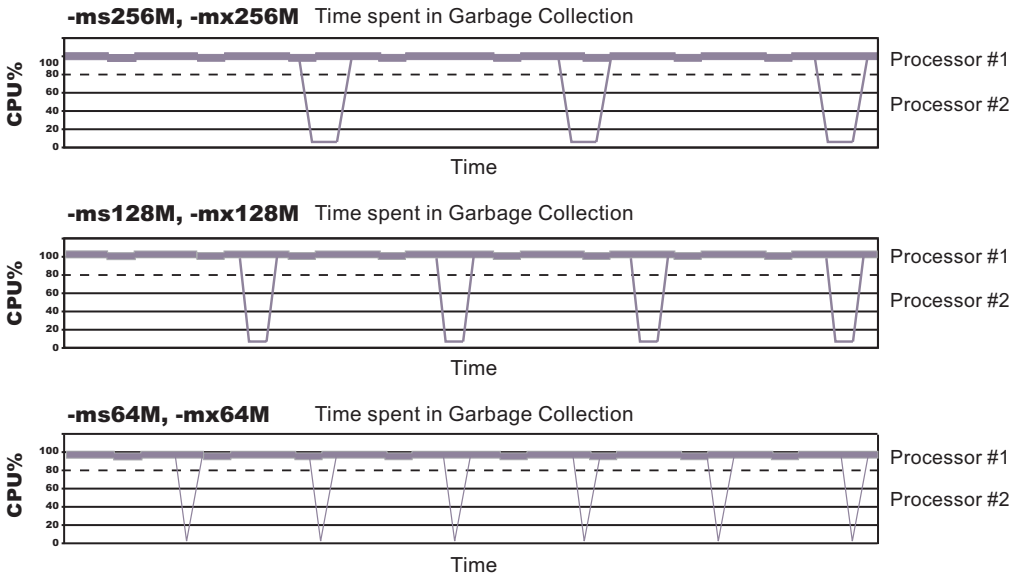
The Java heap parameters also influence the behavior of garbage collection. Increasing the heap size supports more object creation. Because a large heap takes longer to fill, the application runs longer before a garbage collection occurs. However, a larger heap also takes longer to compact and causes 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.

Varying Java Heap Settings



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.

Related tasks

Tuning performance

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:
- "Application servers: Resources for learning"
- "Application servers: Resources for learning"
- "Application servers: Resources for learning"

Programming instructions and examples
- WebSphere Application Server education

Programming specifications
- The Java™ Virtual Machine Specification, Second Edition
- Sun's technology forum for the Java™ Virtual Machine Specification

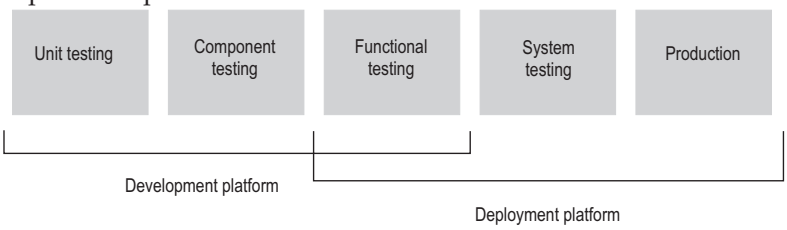
Administration
- Listing of all IBM WebSphere Application Server Redbooks

Related tasks

Chapter 2, "Configuring application servers," on page 3

Testing and production phases

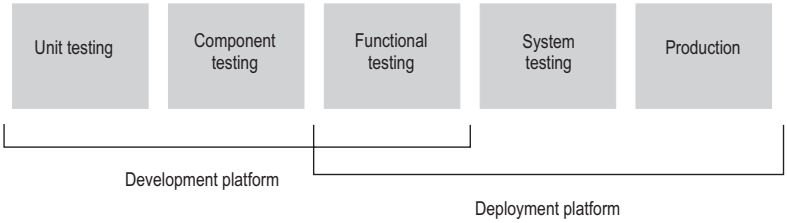
Before explaining the test and production configurations for WebSphere Application Server for z/OS, you must understand which test phase should be done on the z/OS platform and which should be done on other platforms. The graphic below shows the test and production phases. The sections that follow explain the phases.



Sharing resources between a production workload and a test workload potentially can expose the production workload to a set of error conditions to which it would not be exposed if the production and test workloads ran in different cells. For this reason, you should run production and test workloads in separate cells on your system.

Before explaining the test and production configurations for the Application Server, you must understand which test phase should be done on the z/OS platform and which should be done on other platforms. The sections that follow explain the following phases:

- Unit test phase
- Component test phase
- Function test phase
- System test phase
- Production phase



Unit test phase

The development platform for WebSphere Application Server for z/OS is a WebSphere distributed system (for example, Windows or Linux Intel). The development environment includes tools such as WSAD for Web content delivery. The IBM tooling solution assumes that you develop enterprise beans in WSAD and perform unit (basic) testing of the business logic in the WebSphere Test Environment.

Component test phase

Component testing involves the joining together of several beans into logical components, providing them with access to data, and testing them together. While this can be done on WebSphere Application Server for z/OS, most of our current installations perform this level of testing using a distributed platform such as Windows 2000 running WebSphere Application Server Advanced Edition. This allows a small team of developers to join the pieces of code that they have developed together and test the interactions. This testing does not test z/OS platform functions and features directly, but focuses on the individual beans and the relationships between them.

Function test phase

Function testing involves joining the various components together, connecting them to test data in the target database, and validating the function that the application provides. Where this test is performed is dependent on what the function is, and what its data requirements are. If the target deployment platform is z/OS, then it may make sense to do this level of testing there. This is possible by setting up one or more **test servers** into which the application is installed.

When the application is installed into the test server, the installer defines where in the JNDI directory the references to the application will be stored. The test clients will need to be configured with this information that tells the clients the location of the test application. The test clients will then drive requests against the test server to perform the functional testing. You can also use remote debugging tools to diagnose problems you encounter along the way.

System test phase

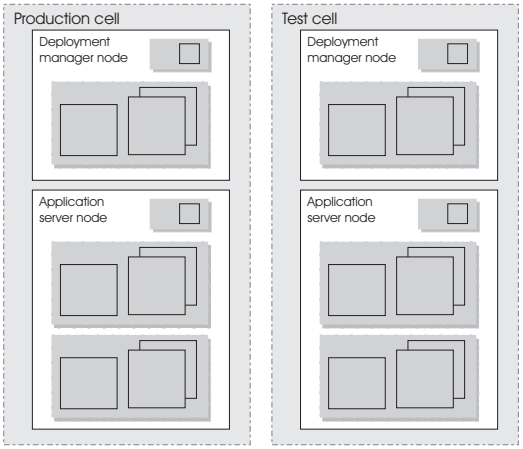
Before you put an application into production on z/OS, you should deploy the code into a WebSphere Application Server for z/OS server for testing. You may bring up the application and simulate a real load on the application. The important point here is that the code needs to run on z/OS before it goes into production. To do this, you need to define an additional **test server** (on a whole new cell dedicated to the test system) and install the application into it. When installed, beans that are part of the application should be registered in a different subtree of the JNDI directory (this occurs by default). The test clients need to be configured to the version of the application that is being tested and the tests run.

Production phase

You can install the application in a production WebSphere Application Server for z/OS cell after you are satisfied with the functional and system testing. The difference between a production cell and a test cell is whether the remote debugger is allowed to be attached. Normally, it is not acceptable for a production workload to stop because someone flowed a remote debugging request to it.

Test cell and production cell configuration

As the graphic below indicates, placing test and production servers into separate cells eliminates all local sharing between test and production and provides the highest risk reduction possible. If you require complete availability of your production system, this configuration eliminates the risk of including production and test in the same cell.



Chapter 3. 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. 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 time-out 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 time-out 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 time-out 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 time-out 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 time-out 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 time-out 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.numJNIReaders system property**

  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. The default value of this property is 2. Valid settings for this property range from 0 to 2147483647.
- **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.

Related reference

"Object Request Broker service settings in administrative console" on page 80

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.

Related tasks

Chapter 3, "Managing Object Request Brokers," on page 77

Related reference

"Object Request Broker communications trace" on page 86

"Object Request Brokers: Resources for learning" on page 93

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 count

Specifies the number of times that the ORB attempts to send a request if a server fails. Retrying sometimes enables recovery from transient network failures.

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

Data type	int
Default	1
Range	1 to 10

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 that can be added to the administrative console

Use the Properties page to set and monitor settings associated with the Java Object Request Broker (ORB) service that do not appear on the main settings page by default.

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**.

The 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.

For a command-line or programmatic alternative, see ″Programming tips for the Java Object Request Broker service.″

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.

For a command-line or programmatic alternative, see ″Programming tips for the Java Object Request Broker service.″

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 this when the amount of data that is sent over IIOP exceeds 1k. Definitly adjust this if thread dumps show most client side threads stuck in a wait coming from writeLock. Sometimes a low amount of fragmentation is ok, so the

parameter should be set such that most messages have few to no fragments.

Units	Bytes.
Default	1024
Range	From 64 to largest value of Java int 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

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]
```

Related tasks

Chapter 3, "Managing Object Request Brokers," on page 77

Related reference

"Character codeset conversion support for the Java Object Request Broker service" on page 92

"Object Request Broker service settings in administrative console" on page 80

"Object Request Brokers: Resources for learning" on page 93

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.

Related reference

"Object Request Brokers: Resources for learning" on page 93

Related information

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	

Java name	OSF registry ID	Comments
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	
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.″

Related reference

“Object Request Brokers: Resources for learning”

Related information

Chapter 3, “Managing Object Request Brokers,” on page 77

Object Request Brokers: Resources for learning

Use the following links to find relevant supplemental information about Object Request Brokers (ORBs). 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 this 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:
- Planning, business scenarios, and IT architecture
- Administration
- Programming specifications

Planning, business scenarios, and IT architecture
- CORBA FAQ

  Getting started with object request brokers and CORBA.

Administration
- IANA Character Set Registry

  This contains a list of all valid character encoding schemes.
- WebSphere Interoperability between Versions 3.5.x and 4.0.x

  This WebSphere Developer Domain article by Joel Sundman and Matt Kelm (February 2002, updated May 2002) is not directly related to the Java ORB service, but it touches upon ORB-related issues.

Programming specifications
- Catalog Of OMG CORBA/IIOP Specifications

Related concepts

"Object Request Brokers" on page 77

Related tasks

Chapter 3, "Managing Object Request Brokers," on page 77

ORB services advanced settings

Use this page to support ORB service advanced settings. This support includes ORB listener keep alive, ORB SSL listener keep alive, control threads, workload profile.

To view this administrative console page, click **Servers** > **Application Servers** > *server name* > **ORB service** > **Advanced Settings**.

ORB listener keep alive

Specifies the maximum time in seconds that an idle IIOP session remains connected.

This property defines the value in seconds provided to TCP/IP on the SOCK_TCP_KEEPALIVE option for the IIOP listener. The function of this option is to verify if idle sessions are still valid by polling the client TCP/IP stack. If the client goes away without notifying the server, it would unnecessarily leave the session active on the server side. Use this option to clean up these unnecessary sessions. If the client does not respond, the session closes. The default is zero. If the property is not set, the TCP/IP option is not set. Setting the SOCK_TCP_KEEPALIVE option generates network traffic on idle sessions, which can be undesirable.

ORB SSL listener keep alive

Specifies the maximum time in seconds that an idle IIOP SSL session remains connected.

This property defines the value in seconds provided to TCP/IP on the SOCK_TCP_KEEPALIVE option for the SSL IIOP listener. The function of this option is to verify if idle sessions are still valid by polling the client TCP/IP stack. If the client goes away without notifying the server, it would unnecessarily leave

the session active on the server side. Use this option to clean up these unnecessary sessions. If the client does not respond, the session closes. The default is zero. If the property is not set, the TCP/IP option is not set. Setting the SOCK_TCP_KEEPALIVE option generates network traffic on idle sessions, which can be undesirable.

Workload Manager Queue Timeout

Specifies the maximum time in seconds that a request is queued while awaiting dispatch to a servant process.

Data type	Integer
Range	1 - 2147040
Default	300

Workload profile

Specifies the server workload profile, which can be ISOLATE, IOBOUND, CPUBOUND, or LONGWAIT

The Workload profile controls workload-pertinent decisions made by the WebSphere for z/OS runtime, such as the number of threads used in the server region. The default value is IOBOUND, which is the appropriate value for most applications. Use one of the other values when your application requires more threads.

Workload profile	Number of Threads	Description
ISOLATE	1	Specifies that the servant region is restricted to a single application thread. Use ISOLATE to ensure that concurrently dispatched applications do not execute in the same servant. Two requests processed in the same servant can cause one request to corrupt another.
IOBOUND	MIN(30, MAX(5,(Number of CPUs*3)))	Specifies more threads in applications that perform I/O-intensive processing on the z/OS operating system. The calculation of the thread number is based on the number of CPUs. IOBOUND is used by most applications that have a balance of CPU intensive and remote operation calls. A gateway or protocol converter are two examples of applications that use the IOBOUND profile.

CPUBOUND	MAX((Number of CPUs-1),3)	Specifies that the application performs processor-intensive operations on the z/OS operating system, and therefore would not benefit from more threads than the number of CPUs. The calculation of the thread number is based on the number of CPUs. Use the CPUBOUND profile setting in CPU intensive applications, like XML parsing and XML document construction, where the vast majority of the application response time is spent using the CPU.
LONGWAIT	40	Specifies more threads than IOBOUND for application processing. LONGWAIT spends most of its time waiting for network/remote operations to complete. Use this setting when the application makes frequent calls to another application system, like CICS screen scraper applications, but does not do much of its own processing.

Chapter 4. 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. If you did not do so when creating a cluster, create one or more cluster members of the cluster.
6. Start all of the application servers by starting the cluster. Workload management automatically begins when you start the cluster members of the application server.
7. Stop the cluster.
8. Upgrade applications on clusters.
9. Detect and handle problems with server clusters and their workloads.

You need to define a bootstrap host for stand-alone Java clients, which are clients located on a different machine from the application server that 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 incoming work requests 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.

WebSphere for z/OS requires that z/OS run workload management in goal mode. If your system runs in compatibility mode, you must implement goal mode. For details on workload management, see *z/OS MVS Planning: Workload Management*, which is available on the z/OS Internet Library Web site. You may also find *z/OS MVS Programming: Workload Management Services* helpful.

In addition to setting up workload management in goal mode, you need to define workload management policies for WebSphere for z/OS servers and your business application servers. This section discusses specifics for the run-time servers. For details on workload management and business applications, see the assembling applications information in the z/OS view of the WebSphere Application Server InfoCenter, which you can access via the WebSphere for z/OS library Web site.

Note: To get started, you do not need to define special classification rules and work qualifiers, but you may want to do this for your production system.

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

- It balances server workloads, allowing processing tasks to be distributed according to the capacities of the different machines in the sysplex.
- 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.

Related tasks

Chapter 4, “Balancing workloads with clusters,” on page 97

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 requests, a particular server is best suited for processing that request.

Related concepts

"Workload management (WLM)" on page 97

Related tasks

Chapter 4, "Balancing workloads with clusters," on page 97

Sysplex routing of work requests

WebSphere Application Server for z/OS routes work requests throughout the cell by using the domain name server (DNS).

The DNS accepts a generic host name from the client and maps the name to a specific system. In order to select the best available system, the DNS asks workload management (WLM) for a recommendation. Workload management analyzes the current state of the cell and considers a number of factors, such as CPU, memory, and I/O utilization, to determine the best placement of new work. The DNS then routes the client request to the optimal system for execution. This use of workload management and the DNS is optional but highly recommended because it eliminates a single point of failure.

Each system in the cell has the WebSphere Application Server for z/OS run-time (the location service daemon, node agent and Deployment Manager), plus business application servers. The client uses the CORBA General Inter-ORB Protocol (GIOP) to make requests of WebSphere for z/OS. The location service daemon acts as a location service agent. It accepts locate requests with object keys in the requests. The location service daemon uses the object key to locate a server that supports the object represented by the object key, then hands the server name to workload management. Workload management chooses the optimal server in the cell to handle the request. The location service daemon merges specific IOR information related to the chosen server with object key information stored in the original IOR. The result of this merging is a direct IOR that gets returned to the client. The client ORB uses this returned reference to establish the IOR connection to the server holding the object of interest.

The transport mechanism that WebSphere Application Server for z/OS uses depends on whether the client is local or remote. If the client is remote (that is, not running on the same z/OS system), the transport is TCP/IP. If the client is local, the transport is through a program call. Local transport is fast because it avoids the physical trip over the network, eliminates data transforms, simplifies the marshalling of requests, and uses optimized RACF facilities for security rather than having to invoke Kerberos or SSL.

Address space management for work requests

WebSphere for z/OS propagates the performance context of work requests through the use of workload management (WLM) enclaves. Each transaction has its own enclave and is managed according to its service class.

The controller of a server, which workload management views as a queue manager, uses the enclave associated with a client request to manage the priority of the work. If the work has a high priority, workload management can direct the work to a high-priority servant in the server. If the work has a low priority, workload management can direct the work to a low-priority servant. The effect is to partition the work according to priority within the same server.

Enclaves can originate in several ways:

- WebSphere Application Server for z/OS uses its own set of rules to create an enclave for a client request from the network.
- Some subsystems (such as Web Server) create enclaves and pass them to the Application Server, which, in turn, passes the enclaves on.
- WebSphere for z/OS treats batch jobs as if they were remote clients.

To communicate the performance context to workload management, you must classify the workloads in your system according to the following work qualifiers.

Table 7. WLM work qualifiers and corresponding WebSphere for z/OS entities

Work qualifier abbreviation	Work qualifier	Corresponding WebSphere for z/OS entity
CN	Collection name	Cluster name
UI	User ID	User ID under which work is running

For more information about classification rules and workload qualifiers, see *z/OS MVS Planning: Workload Management*.

In addition to client workloads, you must consider the performance of the WebSphere Application Server for z/OS run-time servers and your business application servers. In general, server controllers act as work routers, so they must have high priority. Because workload management starts and stops servants dynamically, servants also need high priority in order to be initialized quickly. Once initialized, however, servants run work according to the priority of the client enclave, so the servant priority you assign has no significance after initialization.

In summary, use the following table to set the performance goals for each class:

Table 8. Workload management rules

If you are classifying...	... assign it to:	Reason
The location service daemon	SYSSTC	The system treats it as a started task, and it must route work requests quickly.
An WebSphere for z/OS run-time server controller	SYSSTC	A controller must route work quickly.
An WebSphere for z/OS run-time server servant	SYSSTC	A servant must initialize quickly, but, once initialized, it runs work according to the priority of the client enclave.

Table 8. Workload management rules (continued)

If you are classifying...	... assign it to:	Reason
Your business application controller	A class having at least as much importance as that of the work that flows through it.	A controller must route work quickly, but you must balance the priority of your business application server with other work in the system.
Your business application servant	SYSSTC	A servant must initialize quickly, but, once initialized, it runs work according to the priority of the client enclave.
A client workload	A class having importance relative to other work in your system	Application Server for z/OS and workload management run the work according to the goals you set.

Example of classification rules

Purpose

Let us assume you have three workload management service classes defined for WebSphere Application Server for z/OS (subsystem type CB):

1. CBFAST-designed for transactions requiring fast response times.
2. CBSLOW-designed for long-running applications that do not require fast response times.
3. CBCLASS-designed for remaining work requests.

You design a client workload called BBOC001 that requires fast response times. Also, you want to give work that runs under your manager's user ID (DBOOZ) slower response times. Finally, all remaining work requests should run under the default service class, CBCLASS.

Table 9. Classification rules example

Type column	Name column	Service column	Goal
CN	BBOC001	CBFAST	90% complete in 2 seconds
UI	DBOOZ	CBSLOW	Velocity 50, importance = 3
(default)	(blank)	CBCLASS	Discretionary

You could set the following performance goals through IWMARIN0:

1. Issue IWMARIN0 and choose option 4:

```
  File  Utilities  Notes  Options  Help
-------------------------------------------------------------------------------
  Functionality LEVEL003   Definition Menu  WLM Appl LEVEL004    Command ===>
_______________________________________________________________

Definition data set  . . : 'CB.MYCB.WLM'
Definition name  . . . . . CB390      (Required)
Description  . . . . . . . WLM Setup for WebSphere for z/OS
Select one of the following options. . . . . 4__
1.  Policies
```

```
2.  Workloads
3.  Resource Groups
4.  Service Classes
5.  Classification Groups
6.  Classification Rules
7.  Report Classes
8.  Service Coefficients/Options
9.  Application Environments
10.  Scheduling Environments
```

2. Create a service class called CBFAST and specify that it be 90% complete in 2 seconds.

   Note: The example assumes you have defined a workload called ONLINE.

```
      Service-Class  Notes  Options  Help
------------------------------------------------------------------------------
Create a Service Class
Row 1 to 2 of 2   Command ===> ____________________________________________________
Service Class Name . . . . . . CBFAST    (Required)
Description  . . . . . . . . . Quick CB transactions
Workload Name  . . . . . . . . ONLINE    (name or ?)
Base Resource Group  . . . . . ________  (name or ?)
Specify BASE GOAL information.  Action Codes: I=Insert new period,
E=Edit period, D=Delete period.
---Period---  --------------------Goal--------------------
Action  #  Duration   Imp.  Description

__                                                                          __
1             1    90% complete within 00:00:02.000
 ******************************* Bottom of data ********************************

.-------------------------------------------------------------------------.
| Press EXIT to save your changes or CANCEL to discard them. (IWMAM970) |
'-------------------------------------------------------------------------'
```

3. Save the service class. You see the following:

```
      Service-Class  View  Notes  Options  Help
------------------------------------------------------------------------------
Service Class Selection List
Row 1 to 14 of 21   Command ===> ____________________________________________________
Action Codes: 1=Create, 2=Copy, 3=Modify, 4=Browse, 5=Print, 6=Delete,
/=Menu Bar
Action  Class
Description
Workload
__     CBFAST
 Quick CB Transactions
ONLINE
******************************* Bottom of data ********************************
```

4. Repeat these steps for the CBSLOW service class.
5. Create classification rules using the new service class. Choose option 6 on the main panel:

```
      File  Utilities  Notes  Options  Help
------------------------------------------------------------------------------
Functionality LEVEL003         Definition Menu         WLM Appl LEVEL004
Command ===> ____________________________________________________________
Definition data set  . . : 'CB.MYCB.WLM'
Definition name  . . . . . CB390      (Required)
Description  . . . . . . . WLM Setup for WebSphere for z/OS
Select one of the following options. . . . . 6__
1.  Policies
2.  Workloads
3.  Resource Groups
4.  Service Classes
5.  Classification Groups
6.  Classification Rules
```

```
7.  Report Classes
8.  Service Coefficients/Options
9.  Application Environments
10.  Scheduling Environments
```

6. Create a set of rules for your service classes:

```
      Subsystem-Type  Xref  Notes  Options  Help
------------------------------------------------------------------------------
Create Rules for the Subsystem Type        Row 1 to 2 of 2
Command ===> _____________________________________________   SCROLL ===> PAGE
Subsystem Type . . . . . . . . CB    (Required)
Description  . . . . . . . . .  WebSphere  classification
Fold qualifier names?  . . . . Y  (Y or N)
Action codes:  A=After    C=Copy         M=Move     I=Insert rule
B=Before   D=Delete row   R=Repeat   IS=Insert Sub-rule
 -------Qualifier-------------
-------Class--------
Action    Type       Name     Start
Service     Report
DEFAULTS: CBCLAS      ________
____  1  CN
BBOC001    ___
CBFAST       ________
____  1  UI
DBOOZ      ___
CBSLOW       ________
****************************** BOTTOM OF DATA ******************************
```

In this example, all work for BBOC001, except for work running under the user ID DBOOZ, gets classified as CBFAST. Work for DBOOZ gets classified as CBSLOW. All other work, such as work coming from clients outside the cell and including the work for WebSphere for z/OS run-time servers, gets classified as CBCLASS.

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 small laptop. 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.

A *vertical cluster* has cluster members on the same node. A *horizontal cluster* has cluster members on multiple nodes.

A network dispatcher 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.

Related tasks

"Creating clusters"

Creating clusters

You can manage application servers collectively using a cluster. To create a cluster, view information about clusters, or manage server members on a cluster, use the Server Cluster page.

1. Go to the Server Cluster page. Click **Servers > Clusters** in the console navigation tree. 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, place a checkmark in the **Prefer local enabled** check box. 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 member, add server members to this cluster. To add a server member, choose **Select an existing server to add to this cluster** and then, from the drop-down list, select the server you want to add.
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. Specify whether to generate a unique HTTP port.
   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. 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.
12. 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.
13. 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).

Related tasks

Chapter 4, "Balancing workloads with clusters," on page 97

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.

Related concepts

"Clusters" on page 103

Related tasks

"Creating clusters" on page 104

"Replicating data" on page 111

Related reference

Administrative console buttons
The following button choices are available on various pages of the administrative console, depending on which product features you have enabled.

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

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

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

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*.

Related concepts

"Clusters" on page 103

Related tasks

"Creating clusters" on page 104

Related reference

Administrative console buttons
The following button choices are available on various pages of the administrative console, depending on which product features you have enabled.

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

"Cluster member collection" on page 108

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.

Related tasks

Chapter 4, "Balancing workloads with clusters," on page 97

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**.

Related concepts

"Clusters" on page 103

Related tasks

Chapter 4, "Balancing workloads with clusters," on page 97

"Creating cluster members" on page 107

Related reference

Administrative console buttons
The following button choices are available on various pages of the administrative console, depending on which product features you have enabled.

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

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

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

Node agent collection
Use this page to view information about node agents. Node agents are administrative agents that monitor application servers on a host system and route administrative requests to servers. A node agent is the running server that represents a node in a Network Deployment environment.

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*.

Related concepts

"Clusters" on page 103

Related tasks

"Creating cluster members" on page 107

Related reference

Administrative console buttons
The following button choices are available on various pages of the administrative console, depending on which product features you have enabled.

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

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

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.

Related tasks

"Replicating data" on page 111

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.

Related tasks

"Replicating data" on page 111

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 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 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.

Related tasks

"Replicating data"

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.

Related concepts

"Replication" on page 110

Related tasks

Configuring cache replication

Related reference

ae/uprf_rcachereplication.dita

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.

Related concepts

"Replication domain" on page 110

Related tasks

"Replicating data" on page 111

Related reference

Administrative console buttons
The following button choices are available on various pages of the administrative console, depending on which product features you have enabled.

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

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

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

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.

Related concepts

"Replication domain" on page 110

Related tasks

"Replicating data" on page 111

Related reference

Administrative console buttons
The following button choices are available on various pages of the administrative console, depending on which product features you have enabled.

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

"Internal replication domain collection" on page 113

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.

Related concepts

"Replication entry" on page 110

Related tasks

"Replicating data" on page 111

Related reference

Administrative console buttons
The following button choices are available on various pages of the administrative console, depending on which product features you have enabled.

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

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

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

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.

Related concepts

"Replication entry" on page 110

Related tasks

"Replicating data" on page 111

Related reference

Administrative console buttons
The following button choices are available on various pages of the administrative console, depending on which product features you have enabled.

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

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

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*.

Note: From the z/OS MVS console, you need to start each individual server that you wish to run. With the administrative console, you can start each server individually or start a cluster which will start all defined servers automatically. For more information about starting servers from the z/OS MVS console, see the CONSOLxx parmlib member section of the z/OS MVS Initialization and Tuning Reference, SA22-7592.

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.

Related tasks

Chapter 4, "Balancing workloads with clusters," on page 97

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.

Stop means that all currently-running transactions are carried out before the application controller is taken down, while or **immediate stop** (cancel) means that the application controller is immediately taken down without waiting for the active transactions to complete.

Example: When you request that a server **stop**, the current work is finished before the server is stopped, and when you request an **immediate stop** (cancel), the server stops immediately and ignores any current or pending tasks.

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.

Before you begin: When you stop the daemon on a base application server, it brings down the application servers on that system.When you stop the location service daemon on one system, it doesn't bring down the servers on the other system(s).

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 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*.

   All application servers in the sysplex associated with this cluster will stop. In addition, a **stop** can be issued against each individual server from the MVS console. To shut down the WebSphere for z/OS environment on a system, stop that system's Daemon (see below). It will bring down all other server instances on the system. To bring WebSphere for z/OS down on all systems, stop the Daemons on all systems. When you stop the Daemon, you may get an `A03` abend and dump in your Daemon address space, but this does not impair the stop.

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.

Related tasks

Chapter 4, “Balancing workloads with clusters,” on page 97

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:
- “Clustering and workload management: Resources for learning”
- “Clustering and workload management: Resources for learning”

Programming model and decisions
- Improving availability by clustering the WebSphere Application Server
- Redbook on WebSphere Scalability: WLM and Clustering Using WebSphere Application Server Advanced Edition
- Failover and Recovery in WebSphere Application Server Advanced Edition 4.0

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

Related tasks

Chapter 4, “Balancing workloads with clusters,” on page 97

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

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Mail Station P300
522 South Road
Poughkeepsie, NY 12601-5400
USA
Attention: Information Requests

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

Trademarks and service marks

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

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

The term CORBA used throughout this book refers to Common Object Request Broker Architecture standards promulgated by the Object Management Group, Inc.

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

The Duke logo is a trademark or registered trademark of Sun Microsystems, Inc. in the United States and other countries.

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.