WebSphere Application Server for z/OS V5.0.2:

Servers and Environment

GA22-7958-01

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

Compilation date: December 1, 2003

Contents

How to send your comments v

Chapter 1. Welcome to Application Servers 1

Chapter 2. Configuring application servers 3

Application servers 3
- Understanding the need for application server naming conventions 4

WebSphere for z/OS run-time HFS 6
Creating application servers 8
- Configuring application servers for UTF-8 encoding 9
- Default server values for WebSphere Application Server for z/OS V5 9

Managing application servers 10
- Server collection 11
- Starting servers 24
- Detecting and handling problems with run-time components 27
- Stopping servers. 27
- Displaying the status of ARM-registered address spaces including WebSphere for z/OS servers and server instances 28

Setting up peer restart and recovery 28
- Peer restart and recovery 30
- Recoverable communication manager. 33
- Using RRS panels to resolve indoubt units of recovery 34

Setting up WebSphere Application Server for z/OS on multiple systems in a sysplex 38
- Overview of a WebSphere Application Server for z/OS sysplex 39
- Steps for planning WebSphere Application Server for z/OS and cells 40
- Steps for customizing base z/OS functions on the other systems in the sysplex 40
- Steps for making changes to TCP/IP 42
- Defining multiple WebSphere Application Server for z/OS systems in a sysplex 43

Connection optimization 44
IBM Network Dispatcher 44
Bind-specific support in WebSphere Application Server for z/OS 45
Transports 46
Configuring transports 46
- HTTP transport collection 47
- HTTP transport settings 48
- HTTP transport custom properties. 49

Custom services 52
Developing custom services 52
- Custom service collection 54

Process definition 56
Defining application server processes 56
- Process definition settings 57
- Sysplex Distributor 61
- Multiple TCP/IP stacks 62

Java virtual machines (JVMs) 63
Using the JVM 63
- Java virtual machine settings 63
- Example: Configuring JVM sendRedirect calls to use context root 68
- Example: Setting Custom JVM Properties . . . 68
- Tuning Java virtual machines 69

Preparing to host applications 70
Java memory tuning tips 70
Testing and production phases 75
Test cell and production cell configuration 77
Tuning application servers 77

Chapter 3. Welcome to Clusters 79

Chapter 4. Balancing workloads with clusters 81

Workload management (WLM) 82
- Techniques for managing state 82
- Sysplex routing of work requests 84
- Address space management for work requests. . 85
- Example of classification rules 87
- Managing the number of servant regions . . . 89

Clusters 90
Creating clusters. 91
- Server cluster collection 92

Creating cluster members. 95
- Cluster member collection 95

Replication 97
- Replication entry 97
- Replication domain 98

Replicating data 99
- Internal replication domain collection 100

Starting clusters 106
Stopping clusters 106
WLM dynamic application environment operator commands 107

Chapter 5. Welcome to Variables . . . 109

Chapter 6. Welcome to Web servers 111

Chapter 7. Configuring Web server plug-ins 113

plugin-cfg.xml file 113
Web server plug-ins 121
Installing the WebSphere HTTP Plug-in for z/OS 122
- Private headers 124

Installing a distributed platform Web server plug-in 124

Supported distributed platform Web server plug-in configurations 126
Checking your IBM HTTP Server version 127
Manually editing the plug-in configuration . . . 127
Situations requiring manual editing of the plug-in configuration 128
Regenerating Web server plug-in configurations 130
GenPluginCfg command reference 131
Installing a Global Security Kit for a distributed platform Web server plug-in 132
Gskit install images files 135

Chapter 8. Welcome to Cell-wide settings 137

Chapter 9. Configuring the cell-wide environment 139
Virtual hosts 139
Why and when to use virtual hosting 139
The default virtual host (default_host) 140
How requests map to virtual host aliases . . . 140
Configuring virtual hosts 141
Virtual host collection 141
Variables 146
Configuring WebSphere variables 146
WebSphere variables collection 147
Cell-wide z/OS variables 149
Repository Service Custom Properties 149
Configuring Server custom properties 150
Shared library files 157
Managing shared libraries 158
Shared library collection 159
Library reference collection 160
Environment: Resources for learning 161

Notices 163

Trademarks and service marks 165

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 Application Servers

Overview

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

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

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

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

Asynchronous messaging

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

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

For Network Deployment and Enterprise Extensions, the JMS functions (of JMS providers) within the administration domain are served by one or more JMS servers. There can be at most one JMS server on each node in the administration domain, and any application server within the domain can access JMS resources served by any JMS server on any node in the domain.

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 process execution statements for starting or initializing a UNIX process, monitoring policies to track the performance of a process, process logs to which standard out and standard error streams write, and name-value pairs for properties.
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 70" for a sample procedure that you might follow in configuring the application server run-time and resources.

Related tasks

Tuning performance parameter index

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.

Related tasks

Using the Customization Dialog

Creating application servers

You can create a new application server using the wsadmin tool or the Create New Application Server page of the administrative console.On the Network Deployment product, you can also create a new application server when you add a cluster member to a server cluster.

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

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

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

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

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

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

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)	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	
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 Serversof the administrative console..

Network Deployment and z/OS WebSphere Application Server

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

Base WebSphere Application Server

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

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

1. Access the Application Servers page. Click **Servers > Application Servers** in the console navigation tree.
2. View information about application servers. The Application Servers page lists application servers in the cell and the nodes holding the application servers. 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 JMS servers in a deployment manager cell
Use this task to manage JMS servers on nodes in a WebSphere Application Server deployment manager cell.

Related reference

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

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

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

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

Object names

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. 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

JMS server collection
Each JMS server provides the functions of the JMS provider for a node in your administrative domain. Use this panel to list the JMS servers within the administration domain, or to select a JMS server to view or change its configuration properties.

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

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

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

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

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

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

Administrative console 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 54

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

IBM service log settings

Logging and tracing settings
Use this page to view and configure logging and trace settings for the server.

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.

Performance monitoring service settings
Use this page to specify settings for performance monitoring, including enabling performance monitoring, selecting the PMI module and setting monitoring levels.

"Process definition settings" on page 57

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 web container settings.

Name:

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

Data type	String
Default	server1

Initial State:

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

Data type	String
Default	Started

Application Class loader Policy:

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

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

Data type	String
Default	MULTIPLE

Application Classloading Mode:

Specifies whether the class loader should search in the parent class loader or in the application class loader first to load a class. The standard for Developer Kit class loaders and WebSphere class loaders is PARENT_FIRST. By specifying PARENT_LAST, your application can override classes contained in the parent class loader, but this action can potentially result in ClassCastException or LinkageErrors if you have mixed use of overridden classes and non-overridden classes.

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

Data type	String
Default	PARENT_FIRST

Short name:

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

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

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

Unique Id:

Specifies the unique ID of this server.

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

Process ID:

Specifies a string identifying the process.

Data type	String

Cell Name:

Specifies the name of the cell for the application server.

Data type	String
Default	*host_name*Network

Node Name:

Specifies the name of the node for the application server.

Data type	String

State:

Indicates whether the application server is started or stopped.

Data type	String
Default	Started

End point collection:

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

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

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

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* **> Security Port Endpoint**
- **Servers > JMS Servers >***server_name* **> End Points >***end_point_name*

Related reference

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

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

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

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

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

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

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

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

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

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

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

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

Administrative console 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 16

"Custom property collection" on page 18

"Custom service collection" on page 54

"Server component collection" on page 20

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

Administrative console 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 20, inclusive

Maximum Number of Instances:

Specifies the maximum number of servant process instances to activate

Data type	Integer
Range	Any value. If zero is specified, the number of instances is unlimited.

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 JMS servers on WebSphere Application Server.

Managing WebSphere internal JMS servers
Use this task to manage JMS servers on nodes in a WebSphere Application Server deployment manager cell.

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 for a node in a deployment manager cell, 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: If you created a new server definition using a Base WebSphere Application Server, you cannot start, stop, or manage the new server using the original base application server.

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

There are several options for starting an application server:

- Use the **startServer** command**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**startNode** command and then run the **startServer** command**startServer** command.
- Start an application server using the administrative console.
  1. 1. Click **Servers > Application Servers** from the administrative console navigation tree to go to the Application Server page.
  2. 2. If the node agent for the node on which the application server resides is not running, click **Restart** or **Restart all Servers on Node** on the Node Agents page to start the node agent.

     If the node agent does not start, run the **startNode** command**startNode** command and then run the **startServer** command**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**startNode** command to start the node agent on the node where it runs.
  3. 4. Place a checkmark in the check box beside the application server and click **Start**.

4. 5. View the **Status** value and any messages or logs to see whether the application server starts.

- Start an application server for tracing and debugging.

  To start the application server with standard Java debugging enabled:

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

- Start a Websphere for z/OS Application Server from the MVS console. A typical WebSphere for z/OS run time includes two nodes:
  – Deployment Manager node. This includes a location service daemon and a Deployment Manager with a controller and any number of servants
  – Application server node. This includes a location service daemon, a node agent with a controller, and an application server with a controller and any number of servants

  Before you start any of the application servers, you must verify that all resource managers that are required by your application (DB2, CICS, etc) are available. You must also start all prerequisite subsystems.

  1. 1. To start a server, issue the following command: S controlregionprocname, JOBNAME= server_shortname, ENV= cell_shortname.Node_shortname.Server_shortname where:

     controlregionprocname
     Is the JCL procedure name in the proclib that is used to start the server.

     server_shortname
     Is the short name of the server (or the step name used to start the proc). This allows you to identify the address space that is running when you view it in the SDSF panels.

     cell_shortname.Node_shortname.Server_shortname
     The ENV variable is a concatenation of the cell shortname, the node shortname, and the server shortname.

     Note: This command should be all upper case and should all be on one line (it is shown above on two lines because of space considerations only). For example,

     ```
     S BBO5ACR.BBOS001,ENV=SY1.SY1.BBOS001
     ```

     The following messages indicate that the control region is up:

     ```
     $HASP100 BBO5ACR  ON STCINRDR
     $HASP373 BBO5ACR  STARTED

     BBOO0001I WEBSPHERE FOR Z/OS CONTROL PROCESS BBODMNB/SY1/BBOC001/BBOS001
               IS STARTING.
     IRR812I PROFILE BBO*.* (G) IN THE STARTED CLASS WAS USED TO START BBOS001S
             WITH JOBNAME BBOS001S.
     $HASP100 BBOS001S ON STCINRDR
     $HASP373 BBOS001S STARTED
     ```

```
+BBOO0004I WEBSPHERE FOR Z/OS SERVANT PROCESS BBODMNB/SY1/BBOC001/BBOS001
           IS STARTING.
+BBOO0020I INITIALIZATION COMPLETE FOR WEBSPHERE FOR Z/OS SERVANT PROCESS
           BBOS001.
BBOO0019I INITIALIZATION COMPLETE FOR WEBSPHERE FOR Z/OS CONTROL PROCESS
          BBOS001.
```

2. 2. The Controller Region automatically starts the daemon by issuing a command that looks like:

```
S <dmn_proc>,JOBNAME=<dmn_jobname>,
       ENV=<cell_shortname.Node_shortname.daemon_instancename>
```

Note: This command should all be on one line (it is shown above on two lines because of space considerations only).

Following is an example of the messages that are displayed during daemon startup:

```
BBOO0001I WEBSPHERE FOR Z/OS CONTROL PROCESS BBODMNB/SY1/BBOC001/BBOS001
          IS STARTING.

IRR812I PROFILE BBO*.* (G) IN THE STARTED CLASS WAS USED TO START BBO5DMN
        WITH JOBNAME BBO5DMN.
$HASP100 BBO5DMN  ON STCINRDR
$HASP373 BBO5DMN  STARTED

BBOO0007I WEBSPHERE FOR Z/OS DAEMON BBODMNB/SY1/BBODMNB/SY1 IS STARTING.

IEC130I STEPLIB  DD STATEMENT MISSING
ITT102I CTRACE WRITER BBOWTR IS ALREADY ACTIVE.

BBOO0215I PRODUCT 'WAS FOR Z/OS' SUCCESSFULLY REGISTERED WITH IFAED SERVICE.
BBOO0015I INITIALIZATION COMPLETE FOR DAEMON SY1.
```

3. 3. WLM will start servant address spaces with a command that looks like:

```
S <Srv_Reg_Proc>,JOBNAME=<Server_shortname>,
           ENV=<Cell_shortname.Node_shortname.Server_shortname>
```

Note: This command should all be on one line (it is shown above on two lines because of space considerations only).

4. 4. Determine if the Daemon is up. See Determining if the location service daemon is up. If the Daemon is up, go to the next step to start the server.
5. 5. To start a server from the administrative console, click **Servers** > **Application Server** in the console navigation tree to access the Application Server page.
6. 6. Place a checkmark in the check box beside the application server that you want to start and click **Start**.
7. 7. View the **Status** value and any messages or logs to verify that the server starts.

Related reference

addNode command

removeNode command

serverStatus command

startNode command

START nodeagent_proc_name command

startServer command

START appserver_proc_name command

stopNode command

STOP nodeagent_proc_name command

stopServer command

STOP appserver_proc_name command

startManager command

START dmgr_proc_name command

stopManager command

STOP dmgr_proc_name command

Detecting and handling problems with run-time components

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

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

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.

There are two options for stopping an application server:

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

Related reference

stopServer command

STOP appserver_proc_name command

startServer 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.

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.

Related information

"Repository Service Custom Properties" on page 149

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
```

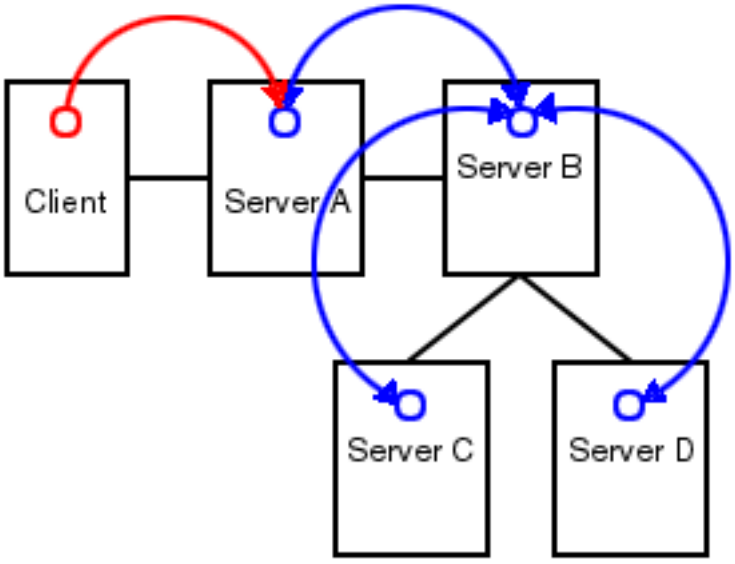
You do **not** want to see:

```
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 *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.

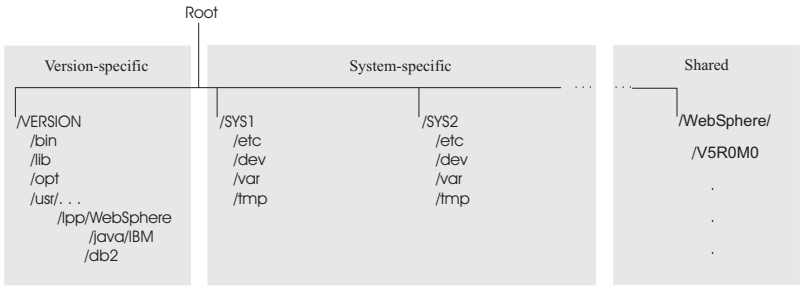
Table 4.

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 40
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 40
Making changes to TCP/IP	"Steps for making changes to TCP/IP" on page 42
Defining new Application Server clustered host instances in the sysplex	"Defining multiple WebSphere Application Server for z/OS systems in a sysplex" on page 43

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.

Table 6. Placing modules in LPA or link list (continued)

Modules	Notes

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:

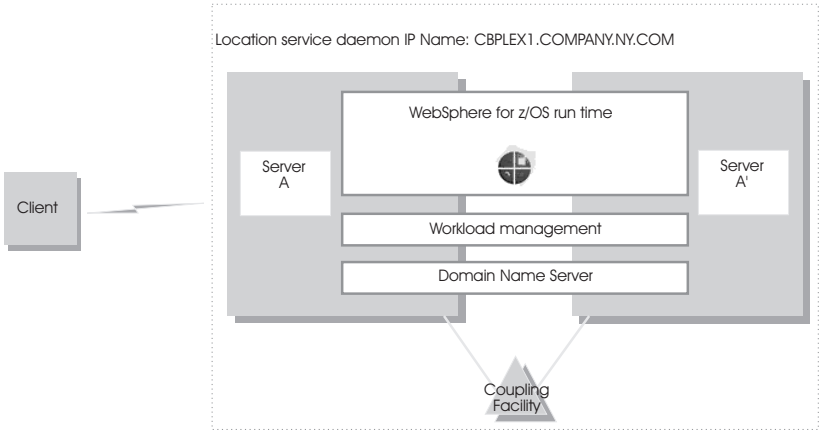
Use the Administrative Console to define additional systems in the sysplex with their servers. We assume you have already created the first WebSphere Application Server for z/OS system with an application server called BBOASR2, BBOC001, or both (the application servers used for the Installation Verification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

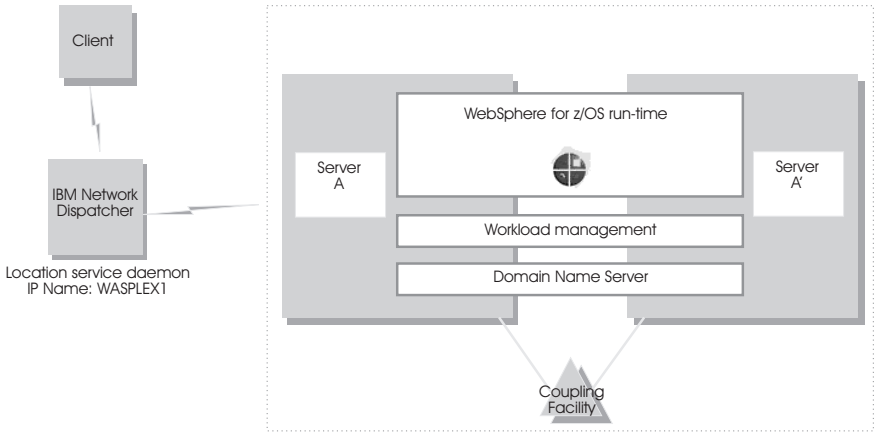
For additional details on setting up servers for connection optimization, see *z/OS Communications Server: IP Configuration Reference*.

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. The HTTP transport provides the means of accepting HTTP requests forwarded by an HTTP plug-in that is connected to a Web server.Prior to WebSphere Application Server Version 5.0.2, the HTTP transport functionality existed only as a means of accepting HTTP requests forwarded by an HTTP plug-in that was connected to a Web server. In Version 5.0.2, HTTP transport functionality is now a supported internal Web server. By default, the internal HTTP transport listens for HTTP requests on port 9080 and for HTTPS requests on port 9443.

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

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

Remember: You can only configure one HTTP port and one HTTPS port.

Related tasks

5.0.2 + Configuring access logging for internal Web server HTTP transport

5.0.2 + Configuring error logging for internal Web server HTTP transport

Configuring transports

A transport is the request queue between a WebSphere Application Server plug-in for Web servers and a Web container in which the Web modules of an application reside. To define the characteristics of the connections between that plug-in and the Web container, you must specify:

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

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**.
   d. Custom properties page, add and set any custom properties you want to use.
3. Regenerate the WebSphere plug-in for the Web server.
4. Stop the Application Server and start it again. You must stop the Application Server and start it again before the configuration changes you made take affect.

If the Web server is located on a machine remote from the Application Server, copy the `plugin-cfg.xml` file to the remote Web server and replace the file that is there. See ″Manually configuring supported Web servers″ in the InfoCenter for the base product for information about copying the `plugin-cfg.xml` and binary plug-in module to a remote Web server and configuring the Web server to use the files.

Related concepts

“Transports” on page 46

Related reference

z/OS port assignments

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 46

Related tasks

“Configuring transports” on page 46

Related reference

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

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

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

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

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

Administrative console 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 18

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

HTTP transport custom properties
Use this page to set custom properties for an HTTP transport.

You can set these properties on either the Web Container or HTTP Transport **Custom Properties** pages. When set on the Web container **Custom Properties** page, all transports inherit the properties. Setting the same properties on a transport overrides like settings defined for a Web container.

To specify values for these custom properties for a specific transport on the HTTP Transport **Custom Properties** page:

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

To specify a custom property:

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

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

Related tasks

Modifying the default Web container configuration

Tuning performance parameter index

Related reference

"HTTP transport settings" on page 48

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

Administrative console 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.

ConnectionIOTimeOut

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

Data type	Integer
Default	5120 seconds

ConnectionKeepAliveTimeout

Specifies the maximum number of seconds to wait for the next request on a keep alive connection. This value is input to the TCP/IP SOCK_TCP_KEEPALIVE option. The Application Server does not set this socket option by default, but will do so if a value is specified for this property.

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

MaxKeepAliveConnections

Specifies the maximum number of concurrent keep alive (persistent) connections across all HTTP transports. To make a particular transport close connections after a request, you can set MaxKeepAliveConnections to `0` (zero) or you can set KeepAliveEnabled to `false` on that transport.
The Web server plug-in keeps connections open to the application server as long as it can. However, if the value of this property is too small, performance is negatively impacted because the plug-in has to open a new connection for each request instead of sending multiple requests through one connection. The application server might not accept a new connection under a heavy load if there are too many sockets in TIME_WAIT state. If all client requests are going through the Web server plug-in and there are many TIME_WAIT state sockets for port 9080, the application server is closing connections prematurely, which decreases performance. The application server closes the connection from the plug-in, or from any client, for any of the following reasons:

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

- A time out occurred while waiting to read the next request or to read the remainder of the current request.

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

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

Data type	Integer
Default	10050 requests

KeepAliveEnabled
Specifies whether or not to keep connections alive

Data type	Boolean.
Default	true

MutualAuthCBindCheck
Specifies whether or not a client certificate should be resolved to a SAF principal. 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)
```

Data type	String
Default	false

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

Data type	Integer
Default	0 (zero)

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

Data type	String
Default	false

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.

For z/OS, custom services run in servants, not in controllers. For example, because there can be more than one servant started in the life of a server and these servants can be started long after the server (controller) is up, as needed by WLM, a custom service runs during the start of each servant.

Related tasks

"Developing custom services"

Related information

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

Developing custom services

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

The following restrictions apply to the WebSphere custom services implementation:

- The init and shutdown methods must return control to the runtime.
- No work is dispatched into the server instance until all custom service initialize methods return.
- The init and shutdown methods are called only once on each service, and once for each operating system process that makes up the server instance. File I/O is supported.
- Initialization of process level static data, without leaving the process, is supported.
- Only JDBC RMLT (resource manager local transaction) operations are supported. Every unit of work (OUW) must be completed before the methods return.
- Creation of threads is not supported.
- Creation of sockets and I/O, other than file I/O, is not supported. Execution of standard J2EE code (client code, servlets, enterprise beans) is not supported.
- The JTA interface is not available. This feature is available in J2EE server processes and distributed generic server processes only.
- While the runtime makes an effort to call shutdown, there is no guarantee that shutdown will be called prior to process termination.

Note that these restrictions apply to the shutdown and init methods equally. Some JNDI operations are available.

1. Develop a custom service class that implements the com.ibm.websphere.runtime.CustomService interface. The properties passed by the application server run-time to the initialize method can include one for an

external file containing configuration information for the service (retrieved with externalConfigURLKey). In addition, the properties can contain any name-value pairs that are stored for the service, along with the other system administration configuration data for the service. The properties are passed to the initialize method of the service as a Properties object.

There is a shutdown method for the interface as well. Both methods of the interface declare that they may throw an exception, although no specific exception subclass is defined. If an exception is thrown, the run-time logs it, disables the custom service, and proceeds with starting the server.

2. On the Custom Service page of the administrative console, click **New**. Then, on the settings page for a custom service instance, create a custom service configuration for an existing application server, supplying the name of the class implemented. If your custom services class requires a configuration file, specify the fully-qualified path name to that configuration file in the **externalConfigURL** field. This file name is passed into your custom service class.
3. Stop the application server and then restart the server.
4. Check the application server to ensure that the initialize method of the custom service ran as intended. Also ensure that the shutdown method performs as intended when the server stops.

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

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

    static String ConfigFileName="";

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

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

```
public void shutdown() throws Exception
{
    // Implement shutdown method
}
```

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

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

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

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

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

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

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 definition in the administrative console. Click **Servers > Application Servers** in the console navigation tree, click on an application server name and then **Process Definition**.
2. On the settings page for a process definition, specify the name of the executable to run, any arguments to pass when the process starts running, and the working directory in which the process will run. Then click **OK**.
3. Specify process execution statements for starting or initializing a UNIX process.
4. Specify monitoring policies to track the performance of a process.
5. Specify process logs to which standard out and standard error streams write. Complete this step if you do not want to use the default file names.
6. Specify name-value pairs for properties needed by the process definition.
7. Stop the application server and then restart the server.
8. Check the application server to ensure that the process definition runs and operates as intended.

Related concepts

"Process definition"

Related tasks

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

Running the deployment manager with a non-root user ID

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

Process definition settings

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

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

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" on page 56

Related tasks

"Defining application server processes" on page 56

Related reference

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

Administrative console 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 18

Start Command

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

Data type	String
Format	START *control JCL procedure name*

Example	START BBO5ACR

Servant process

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

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

Start Command Args

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

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

Servant process

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

Stop Command

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

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

Stop Command Args

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

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

Terminate Command

Specifies the platform-specific command to terminate the server process

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

Terminate Command Args

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

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

Executable Name

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

Data type	String

Executable Arguments

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

Data type	String
Units	Java command-line arguments

Working Directory

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

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

Data type	String

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

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

Administrative console 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. Establish the Application Server's stack affinity to the desired stack.
3. Next, do the following:
   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 9" 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 68" provides an example that involves defining a property for the JVM.

Related concepts

"Java virtual machines (JVMs)"

Related tasks

Deploying and managing using scripting

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)"

Related tasks

"Using the JVM" on page 63

Tuning performance parameter index

Related reference

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

Administrative console 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 18

Classpath

Specifies the standard class path in which the Java virtual machine code looks for classes.
Enter each classpath entry into a table row. You do not need to add the colon or semicolon at the end of each entry.

Data type	String
Units	Class path

Boot Classpath

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

Data type	String

Verbose Class Loading

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

Data type	Boolean
Default	false

Verbose Garbage Collection

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

Data type	Boolean
Default	false

Verbose JNI

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

Data type	Boolean
Default	false

Initial Heap Size

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

Increasing the minimum heap size can improve startup. The number of garbage collection occurrences are reduced and a 10% gain in performance is realized.

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

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

Maximum Heap Size

Specifies the maximum heap size available to the JVM code, in megabytes. Increasing the heap size can improve startup. By increasing heap size, you can reduce the number of garbage collection occurrences with a 10% gain in performance.

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

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

Run HProf

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

Data type	Boolean
Default	false

HProf Arguments

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

Data type	String

Debug Mode

Specifies whether to run the JVM in debug mode. The default is not to enable debug mode support.
If you set the Debug Mode property to true, then you must specify command-line debug arguments as values for the Debug Arguments property.

Data type	Boolean
Default	false

Debug Arguments

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

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

Data type	String
Units	Java command-line arguments

Generic JVM Arguments

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

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

Note: If the argument says it is for the IBM Developer Kit only, you cannot use the argument with another JVM, such as the Sun JDK or the HP JDK.

- **-Xquickstart:** You can use **-Xquickstart** for initial compilation at a lower optimization level than in default mode. Later, depending on sampling results, you can recompile to the level of the initial compile in default mode. Use **-Xquickstart** for applications where early moderate speed is more important than long run throughput. In some debug scenarios, test harnesses and short-running tools, you can improve startup time between 15-20%.

  The **-Xquickstart** option is not supported on OS/400.
- **-Xverify:none:** When using this value, the class verification stage is skipped during class loading . By using **-Xverify:none** with the just in time (JIT) compiler enabled, startup time is improved by 10-15%.

  The **-Xverify:none** option is not supported on OS/400.
- **-Xnoclassgc:** You can use this value to disable class garbage collection, which leads to more class reuse and slightly improved performance. The trade-off is that you won't be collecting the resources owned by these classes. You can monitor garbage collection using the verbose:gc configuration setting, which will output class garbage collection statistics. Examining these statistics will help you understand the trade-off between the reclaimed resources and the amount of garbage collection required to reclaim the resources. However, if the same set of classes are garbage collected repeatedly in your workload, you should disable garbage collection. Class garbage collection is enabled by default.
- **-Xgcthreads:** You can use several garbage collection threads at one time, also known as *parallel garbage collection*. When entering this value in the **Generic JVM Arguments** field, also enter the number of processors that your machine has, for example, `-Xgcthreads=`*number_of_processors*. On a node with *n* processors, the default number of threads is *n*. You should use parallel garbage collection if your machine has more than one processor. This argument is valid only for the IBM Developer Kit.

  The **-Xgcthreads** option is not supported on OS/400.
- **-Xnocompactgc:** This value disables heap compaction which is the most expensive garbage collection operation. Avoid compaction in the IBM Developer Kit. If you disable heap compaction, you eliminate all associated overhead.
- **-Xinitsh:** You can use this value to set the initial heap size where class objects are stored. The method definitions and static fields are also stored with the class objects. Although the system heap size has no upper bound, set the initial size so that you do not incur the cost of expanding the system heap size, which involves calls to the operating system memory manager. You can compute a good initial system heap size by knowing the number of classes loaded in the WebSphere product, which is about 8,000 classes, and their average size. Having

knowledge of the applications helps you include them in the calculation. You can use this argument only with the IBM Developer Kit.

- **-Xgpolicy:** You can use this value to set the garbage collection policy. If the garbage collection policy (gcpolicy) is set to `optavgpause`, concurrent marking is used to track application threads starting from the stack before the heap becomes full. The garbage collector pauses become uniform and long pauses are not apparent. The trade-off is reduced throughput because threads might have to do extra work. The default, recommended value is `optthruput`. Enter the value as `-Xgcpolicy:[optthruput|optavgpause]`. You can use this argument only with the IBM Developer Kit.
- **-XX:** The Sun-based Java Development Kit (JDK) Version 1.3 has generation garbage collection, which allows separate memory pools to contain objects with different ages. The garbage collection cycle collects the objects independently from one another depending on age. With additional parameters, you can set the size of the memory pools individually. To achieve better performance, set the size of the pool containing short lived objects so that objects in the pool do not live through more then one garbage collection cycle. The size of new generation pool is determined by the `NewSize` and `MaxNewSize` parameters. Objects that survive the first garbage collection cycle are transferred to another pool. The size of the survivor pool is determined by parameter `SurvivorRatio`. If garbage collection becomes a bottleneck, you can try customizing the generation pool settings. To monitor garbage collection statistics, use the object statistics in Tivoli Performance Viewer or the verbose:gc configuration setting. Enter the following values: `-XX:NewSize (lower bound)` , `-XX:MaxNewSize (upper bound)`, and `-XX:SurvivorRatio=NewRatioSize`. The default values are:`NewSize=2m MaxNewSize=32m SurvivorRatio=2` However, if you have a JVM with more than 1 GB heap size, you should use the values: `-XX:newSize=640m -XX:MaxNewSize=640m -XX:SurvivorRatio=16`, or set 50 to 60% of total heap size to a new generation pool.

  The **-XX** option is not supported on OS/400.
- **-server | -client:** Java HotSpot Technology in the Sun-based Java Development Kit (JDK) Version 1.3 introduces an adaptive JVM containing algorithms for optimizing byte code execution over time. The JVM runs in two modes, **-server** and **-client**. If you use the default **-client** mode, there will be a faster startup time and a smaller memory footprint, but lower extended performance. You can enhance performance by using **-server** mode if a sufficient amount of time is allowed for the HotSpot JVM to warm up by performing continuous execution of byte code. In most cases, use **-server** mode, which produces more efficient run-time execution over extended periods. You can monitor the process size and the server startup time to check the difference between **-client** and**-server**.

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

Data type	String
Units	Java command line arguments

Executable JAR File Name

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

Data type	String
Units	Path name

Disable JIT

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

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

Operating System Name

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

Data type	String

Example: Configuring JVM sendRedirect calls to use context root

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

1. Access the settings page for a property of the JVM.
   a. Click **Servers > Application Servers** in the console navigation tree.
   b. On the Application Server page, click on the name of the server whose JVM settings you want to configure.
   c. On the settings page for the selected application server, click **Process Definition**.
   d. On the Process Definition page, click **Java Virtual Machine**.
   e. On the Java Virtual Machine page, click **Custom Properties**.
   f. On the Properties page, click **New**.
2. On the settings page for a property, specify a name of `com.ibm.websphere.sendredirect.compatibility` and either `true` or `false` for the value, then click **OK**.
3. Click **Save** on the console taskbar.
4. Stop the application server and then restart the application server

Related tasks

"Using the JVM" on page 63

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

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

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

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

Steps for this task

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

Application server	**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.

Tuning Java virtual machines

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

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

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

    You can find more information about generational garbage collection at http://java.sun.com/docs/hotspot/gc/index.html.
  - Sun Java Development Kit 1.3 HotSpot JVM warm-up (-server)
  - Heap compaction (-Xnocompactgc)
  - Initial system heap size (-Xinitsh)
- **Set the initial heap size.**
- **Set the maximum heap size.**
- **Disable just in time (JIT) compiler.**

Preparing to host applications

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

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

Related tasks

Deploying and managing applications

Java memory tuning tips

Enterprise applications written in the Java language involve complex object relationships and utilize large numbers of objects. Although, the Java language automatically manages memory associated with object life cycles, understanding the application usage patterns for objects is important. In particular, verify the following:

- The application is not over-utilizing objects
- The application is not leaking objects
- The Java heap parameters are set properly to handle a given object usage pattern

Understanding the effect of garbage collection is necessary to apply these management techniques.

The garbage collection bottleneck

Examining Java garbage collection gives insight to how the application is utilizing memory. Garbage collection is a Java strength. By taking the burden of memory management away from the application writer, Java applications are more robust than applications written in languages that do not provide garbage collection. This robustness applies as long as the application is not abusing objects. Garbage collection normally consumes from 5% to 20% of total execution time of a properly functioning application. If not managed, garbage collection is one of the biggest bottlenecks for an application, especially when running on symmetric multiprocessing (SMP) server machines. The Java virtual machine (JVM) uses a parallel garbage collector to fully exploit an SMP during most garbage collection cycles where the Sun HotSpot 1.3.1 JVM has a single-threaded garbage collector. For more information about garbage collection in a Solaris operating environment see Performance: Resources for learning.

The garbage collection gauge

You can use garbage collection to evaluate application performance health. By monitoring garbage collection during the execution of a fixed workload, you gain insight as to whether the application is over-utilizing objects. Garbage collection can even detect the presence of memory leaks.

You can monitor garbage collection statistics using object statistics in the Tivoli Performance Viewer, or using the **verbose:gc**JVM configuration setting. The **verbose:gc** format is not standardized between different JVMs or release levels. For a description of the IBM verbose:gc output and more information about the IBM garbage collector, see Performance: Resources for learning.

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

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

Detecting over-utilization of objects

You can use the Tivoli Performance Viewer to check if the application is overusing objects, by observing the counters for the JVM runtime. You have to set the **-XrunpmiJvmpiProfiler** command line option, as well as the JVM module maximum level in order to enable the Java virtual machine profiler interface (JVMPI) counters. The best result for the average time between garbage collections is at least 5-6 times the average duration of a single garbage collection. If you do not achieve this number, the application is spending more than 15% of its time in garbage collection.

If the information indicates a garbage collection bottleneck, there are two ways to clear the bottleneck. The most cost-effective way to optimize the application is to implement object caches and pools. Use a Java profiler to determine which objects to target. If you can not optimize the application, adding memory, processors and clones might help. Additional memory allows each clone to maintain a reasonable heap size. Additional processors allow the clones to run in parallel.

Detecting memory leaks

Memory leaks in the Java language are a dangerous contributor to garbage collection bottlenecks. Memory leaks are more damaging than memory overuse, because a memory leak ultimately leads to system instability. Over time, garbage collection occurs more frequently until the heap is exhausted and the Java code fails with a fatal `Out of Memory` exception. Memory leaks occur when an unused object has references that are never freed. Memory leaks most commonly occur in collection classes, such as Hashtable because the table always has a reference to the object, even after real references are deleted.

High workload often causes applications to crash immediately after deployment in the production environment. This is especially true for leaking applications where the high workload accelerates the magnification of the leakage and a memory allocation failure occurs.

The goal of memory leak testing is to magnify numbers. Memory leaks are measured in terms of the amount of bytes or kilobytes that cannot be garbage collected. The delicate task is to differentiate these amounts between expected sizes of useful and unusable memory. This task is achieved more easily if the numbers are magnified, resulting in larger gaps and easier identification of inconsistencies. The following list contains important conclusions about memory leaks:

- **Long-running test**

  Memory leak problems can manifest only after a period of time, therefore, memory leaks are found easily during long-running tests. Short running tests can lead to false alarms. It is sometimes difficult to know when a memory leak is occurring in the Java language, especially when memory usage has seemingly increased either abruptly or monotonically in a given period of time. The reason it is hard to detect a memory leak is that these kinds of increases can be valid or might be the intention of the developer. You can learn how to differentiate the delayed use of objects from completely unused objects by running applications for a longer period of time. Long-running application testing gives you higher confidence for whether the delayed use of objects is actually occurring.
- **Repetitive test**

  In many cases, memory leak problems occur by successive repetitions of the same test case. The goal of memory leak testing is to establish a big gap between unusable memory and used memory in terms of their relative sizes. By repeating the same scenario over and over again, the gap is multiplied in a very progressive way. This testing helps if the number of leaks caused by the execution of a test case is so minimal that it is hardly noticeable in one run.

  You can use repetitive tests at the system level or module level. The advantage with modular testing is better control. When a module is designed to keep the private module without creating external side effects such as memory usage, testing for memory leaks is easier. First, the memory usage before running the module is recorded. Then, a fixed set of test cases are run repeatedly. At the end of the test run, the current memory usage is recorded and checked for significant changes. Remember, garbage collection must be suggested when recording the actual memory usage by inserting System.gc() in the module where you want garbage collection to occur, or using a profiling tool, to force the event to occur.
- **Concurrency test**

  Some memory leak problems can occur only when there are several threads running in the application. Unfortunately, synchronization points are very susceptible to memory leaks because of the added complication in the program logic. Careless programming can lead to kept or unreleased references. The incident of memory leaks is often facilitated or accelerated by increased concurrency in the system. The most common way to increase concurrency is to increase the number of clients in the test driver.

Consider the following points when choosing which test cases to use for memory leak testing:
  - A good test case exercises areas of the application where objects are created. Most of the time, knowledge of the application is required. A description of the scenario can suggest creation of data spaces, such as adding a new record, creating an HTTP session, performing a transaction and searching a record.
  - Look at areas where collections of objects are used. Typically, memory leaks are composed of objects within the same class. Also, collection classes such as Vector and Hashtable are common places where references to objects are implicitly stored by calling corresponding insertion methods. For example, the get method of a Hashtable object does not remove its reference to the retrieved object.

Tivoli Performance Viewer can help find memory leaks. For best results, repeat experiments with increasing duration, like 1000, 2000, and 4000-page requests. The Tivoli Performance Viewer graph of used memory should have a sawtooth shape. Each drop on the graph corresponds to a garbage collection. There is a memory leak if one of the following occurs:
- The amount of memory used immediately after each garbage collection increases significantly. The sawtooth pattern looks more like a staircase.
- The sawtooth pattern has an irregular shape.

Also, look at the difference between the number of objects allocated and the number of objects freed. If the gap between the two increases over time, there is a memory leak.

Heap consumption indicating a possible leak during a heavy workload (the application server is consistently near 100% CPU utilization), yet appearing to recover during a subsequent lighter or near-idle workload, is an indication of heap fragmentation. Heap fragmentation can occur when the JVM can free sufficient objects to satisfy memory allocation requests during garbage collection cycles, but the JVM does not have the time to compact small free memory areas in the heap to larger contiguous spaces.

Another form of heap fragmentation occurs when small objects (less than 512 bytes) are freed. The objects are freed, but the storage is not recovered, resulting in memory fragmentation until a heap compaction has been run.

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

Java heap parameters

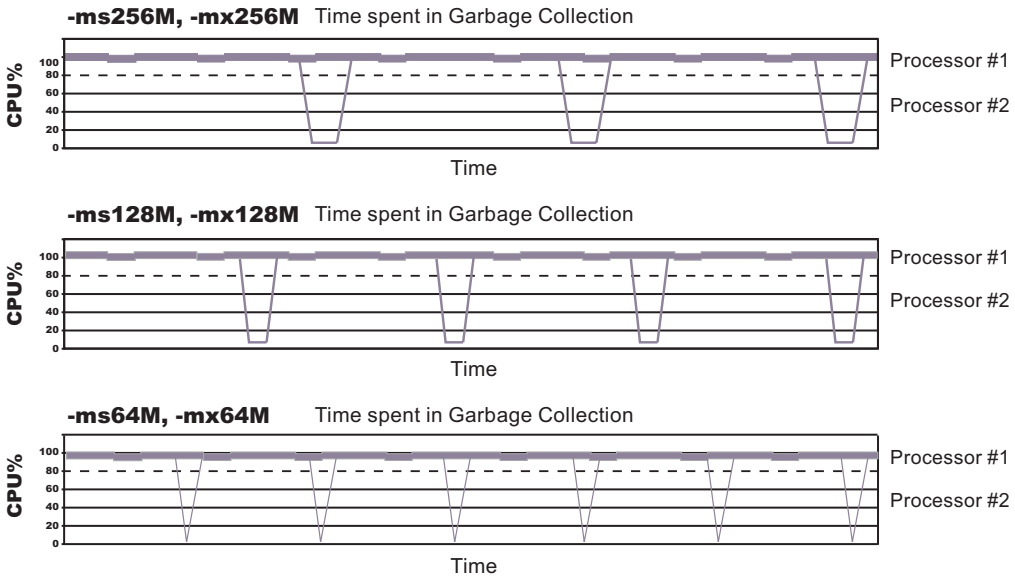
The Java heap parameters also influence the behavior of garbage collection. Increasing the heap size supports more object creation. Because a large heap takes longer to fill, the application runs longer before a garbage collection occurs. However, a larger heap also takes longer to compact and causes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

Note that Java Heap information is contained in SMF records and can be viewed dynamically using the console command DISPLAY,JVMHEAP.

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.

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

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

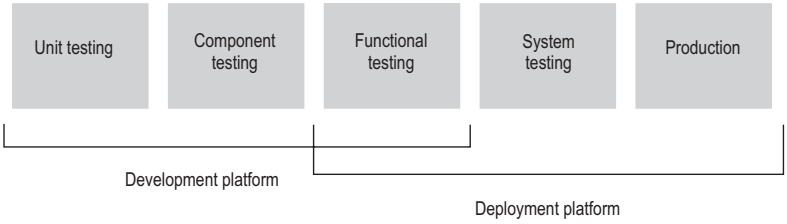
IBM Support has documents that can save you time gathering information needed to resolve this problem. Before opening a PMR, see the IBM Support page.

Related tasks

Tuning performance parameter index

Testing and production phases

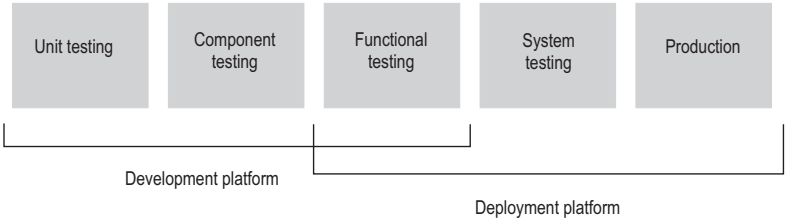
Before explaining the test and production configurations for WebSphere Application Server for z/OS, you must understand which test phase should be done on the z/OS platform and which should be done on other platforms. The graphic below shows the test and production phases. The sections that follow explain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

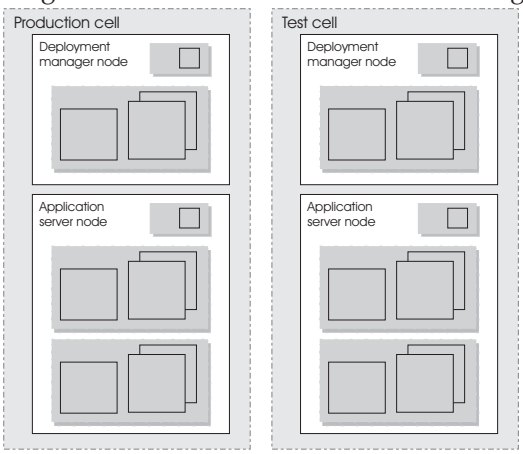
Before you put an application into production on z/OS, you should deploy the code into a WebSphere Application Server for z/OS server for testing. You may bring up the application and simulate a real load on the application. The important point here is that the code needs to run on z/OS before it goes 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.



Tuning application servers

The WebSphere Application Server contains interrelated components that must be harmoniously tuned to support the custom needs of your end-to-end e-business application. This group of interrelated components is known as the queuing network. The queuing network helps the system achieve maximum throughput while maintaining the overall stability of the system.

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

- **Tune the object request broker.** An Object Request Broker (ORB) manages the interaction between clients and servers, using the Internet InterORB Protocol (IIOP). It supports client requests and responses received from servers in a network-distributed environment. You can tune the ORB with the following parameters:
  - **Pass by reference (com.ibm.CORBA.iiop.noLocalCopies)**
  - Thread pool **Maximum size**
  - **com.ibm.CORBA.FragmentSize**

  The Object Request Broker tuning guidelines offer tips on using these parameters to tune the ORB.
- **Tune the XML parser definitions.**
  - **Description:** Facilitates server startup by adding XML parser definitions to the `jaxp.properties` and `xerxes.properties` files in the ${*install_root*}/jre/lib directory. The XMLParserConfiguration value might change as new versions of Xerces are provided.
  - **How to view or set:** Insert the following lines in both files:

    ```
    javax.xml.parsers.SAXParserFactory=org.apache.xerces.jaxp.SAXParserFactoryImpl
    javax.xml.parsers.DocumentBuildFactory=org.apache.xerces.jaxp.DocumentBuilderFactoryImpl
    org.apache.xerces.xni.parser.XMLParserConfiguration=org.apache.xerces.parsers.StandardParser
    ```
  - **Default value:** None
  - **Recommended value:** None

- **Tune the dynamic cache service.** Using the dynamic cache service can improve performance. See Improving performance through the dynamic cache service for information about using the dynamic cache service and how it can affect your application server performance.
- **Tune the Web container.** One of the parts of each WebSphere Application Server is a Web container. To route servlet requests from the Web server to the Web containers, the product establishes a transport queue between the Web server plug-in and each Web container. The Web container is initially created with default property values suitable for simple Web applications. However, these values might not be appropriate for more complex Web applications. Using the following parameters, you can tune the Web container to fit the specific needs of your application server.
  - Set the thread pool **Maximum size**.
  - Specify a **growable thread pool**.
  - Set **MaxKeepAliveConnections** and **MaxKeepAliveRequests** on the HTTP transport custom settings page.
- **Tune the EJB container.** An EJB container is automatically created when you create an application server. After the EJB container is deployed, you can use the following parameters to make adjustments that improve performance.
  - Set the **Cleanup interval**.
  - Set the **Cache size**.
  - **Break CMP enterprise beans into several enterprise bean modules** while Assembling EJB modules.

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

Chapter 3. Welcome to 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 by enabling 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. With z/OS, you can set up workload management to balance the tasks between different clusters.

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.

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

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.
7. Regenerate the plug-in configuration. After changing configurations to plug-ins, transports or virtual hosts, you must regenerate your Web server plug-in for the changes to take effect.
8. **5.0.2 +** If you did not do so when creating a cluster, configure a weight advisor to assist the cluster in balancing the work loads of cluster members.
9. **5.0.2 +** Configure a backup cluster that handles requests if the primary cluster fails.
10. Start all of the application servers by starting the cluster. Workload management automatically begins when you start the cluster members of the application server.
11. Stop the cluster.
12. Upgrade applications on clusters.
13. Detect and handle problems with server clusters and their workloads.
14. Tune the behavior of the workload management run time. If your application is experiencing problems with timeouts or your network experiences extreme latency, change the timeout interval for the com.ibm.CORBA.RequestTimeout property. Or,if the workload management state of the client is refreshing too soon or too late, change the interval for the com.ibm.websphere.wlm.unusable.interval property.

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

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

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

Related tasks

"Regenerating Web server plug-in configurations" on page 130

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 81

Techniques for managing state

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

requests that depend on the results received from the earlier requests. This sequence of operations on behalf of a client falls into two categories:

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

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

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

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

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

  Related concepts

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

  "Workload management (WLM)" on page 82

  "Clusters" on page 90

  Sessions
  A session is a series of requests to a servlet, originating from the same user at the same browser.

  Session management support
  WebSphere Application Server provides facilities, grouped under the heading

Session Management, that support the javax.servlet.http.HttpSession interface described in the Servlet API specification.

Related tasks

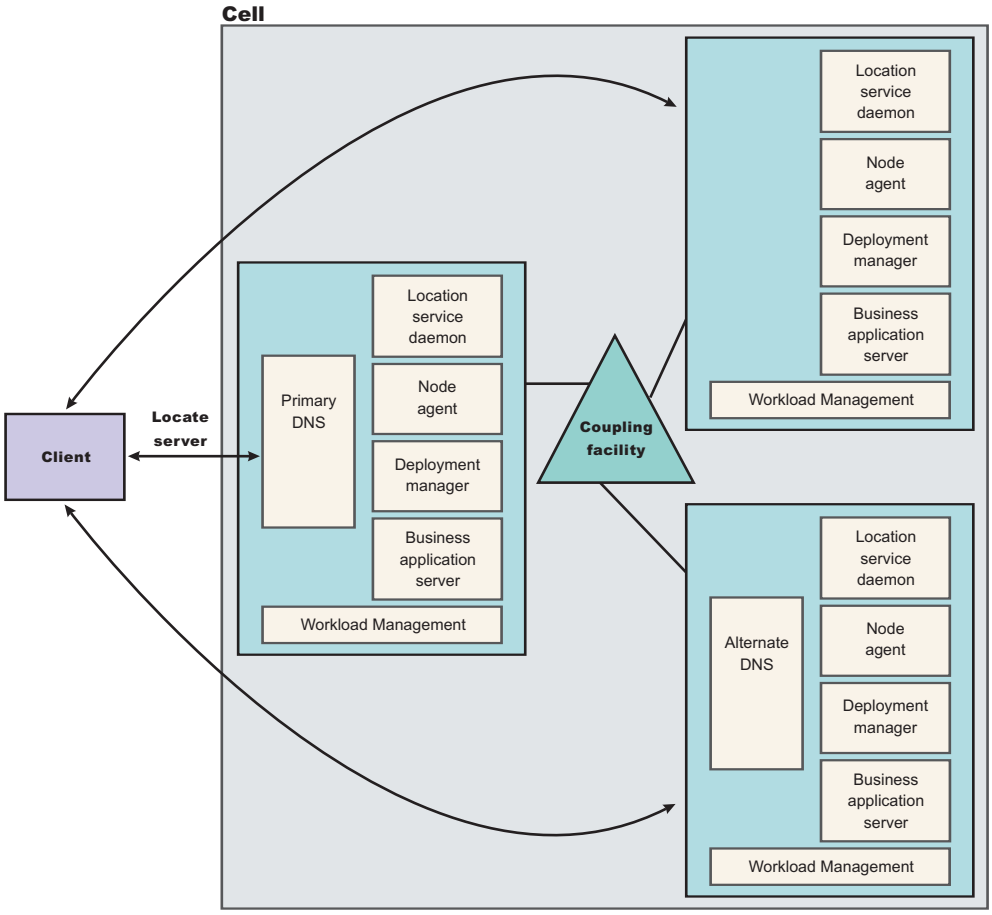
Chapter 4, "Balancing workloads with clusters," on page 81

Managing HTTP sessions

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.

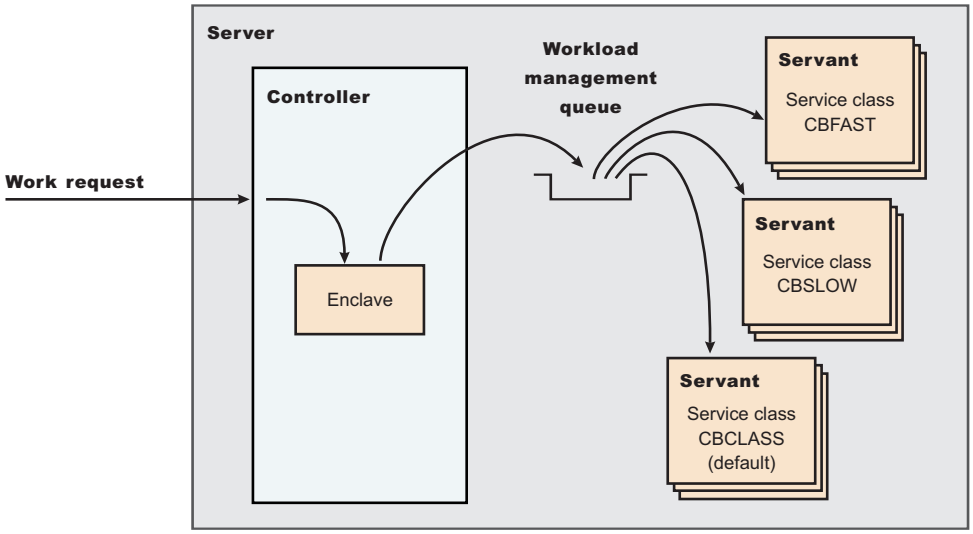
The location service daemon uses the object key to locate a server that supports the object represented by the object key, then hands the server name to workload management. Workload management chooses the optimal server in the cell to handle the request. The location service daemon merges specific IOR 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.
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.

Managing the number of servant regions

With Workload Management (WLM), you can manage the performance and number of application servant regions in WebSphere Application Server for z/OS. WLM manages the response time and throughput of WebSphere transactions according to their assigned service class, the associated performance objectives, and the availability of system resources. To meet these goals, WLM sometimes needs to control or override the number of servant regions that are active.

WebSphere applications are deployed within a WebSphere generic *server*. One or more server instances must be defined on one or more systems within the WebSphere *node*. Each server instance consists of a *controller region* and one or more *servant regions*. The controller regions are started by MVS as *started tasks*, and servant regions are started by WLM, as they are needed.

If you have WLM dynamic application environments enabled on your system (z/OS Release 2 plus APAR OW54622 or later), WLM honors the specifications for the number of servant regions. If you are using static application environments (specified through the WLM ISPF panels), then you must also enable multiple servant regions by indicating **No limit** in the WLM ISPF panels.

Note: Be aware of the following regarding multiple servant regions:

- Some applications, such as the Admin Console application itself and the deployment manager, have serialization requirements that only work in a single JVM. These cannot be run in a server with multiple servant regions. Do not enable multiple instances for any of these servers.
- If you specify a maximum number of instances, WLM is restricted from starting more than this number of servant regions for this server instance.
- **The maximum number of servant regions should be at least as large as the number of different service classes that might be used by transactions that are run in the server**. Remember to account for the *default CB-type service class* and enclaves that may originate outside WebSphere servers and are classified by other classification rules such as the IBM HTTP Server (IHS).

Related tasks

Classifying WebSphere transaction workload for WLM
This topic describes how to use transaction classes to classify client workload for workload management. The workload is different WebSphere transactions targeted to separate servant regions, each with goals defined by appropriate service classes. Each transaction is dispatched in its own WLM enclave in a servant region process, and is managed according to the goals of its service class.

Enabling multiple servant regions

1. Start the Administrative Console.
2. Click **Servers > Application Servers >** *server name* **> Server Instance**.
3. Put a check mark in the box beside **Multiple Instances Enabled**.

4. Click **Apply** to finish the Server Instance changes.
5. Click **OK**.

Workload Management (WLM) may start additional servant regions to meet performance goals, based on the prioritization (importance) of its work compared to other work in the system, the availability of system resources that are needed to satisfy those objectives, and a determination by WLM of whether starting more address spaces will help achieve the objectives. It is also important to make the goals reasonable.

Related concepts

"Managing the number of servant regions" on page 89

Related tasks

"Controlling the number of servant regions"

Controlling the number of servant regions

You can control the minimum or maximum number of servant regions for a server using the Administrative Console page. The minimum value is useful for starting up a basic number of servant regions before your day's work arrives. This can reduce delays while waiting for WLM to start up additional regions. The maximum value is useful for *capping* the number of address spaces that are started by WLM for each *server instance*, if you determine that excessive servant regions are contributing to service degradation.

1. Start the Administrative Console.
2. Click **Servers > Application Servers >** *server name* **> Server Instance**.
3. Type a value into the **Multiple Instances Enabled** and **Maximum Number of Instances** field, or leave these fields blank to allow WLM to determine the numbers.
4. Click **Apply** to finish the Server Instance changes.
5. Click **OK**.

Related concepts

"Managing the number of servant regions" on page 89

Related tasks

"Enabling multiple servant regions" on page 89

Related reference

"Servant Instance Settings" on page 21

Clusters

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

A cell can have no clusters, one cluster, or multiple clusters.

Servers that belong to a cluster are *members* of that cluster set and must all have identical application components deployed on them. Other than the applications configured to run on them, cluster members do not have to share any other configuration data. One cluster member might be running on a huge multi-processor enterprise server system while another member of that same cluster might be running on a smaller system. The server configuration settings for

each of these two cluster members are very different, except in the area of application components assigned to them. In that area of configuration, they are identical.

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

A WebSphere Web server plug-in routes application access among cluster members by server-weighting, to provide better distribution control.

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

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 81

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 90

Related tasks

"Creating clusters" on page 91

"Replicating data" on page 99

Related reference

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

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

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

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

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 90

Related tasks

"Creating clusters" on page 91

Related reference

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

Administrative console 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 95

5.0.2 + Weight advisor settings
Use this page to provide information about the weights assigned to cluster members. A server cluster uses this information to balance the work loads of its cluster members.

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

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 81

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 90

Related tasks

Chapter 4, "Balancing workloads with clusters," on page 81

"Creating cluster members" on page 95

Related reference

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

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

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

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

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 90

Related tasks

"Creating cluster members" on page 95

Related reference

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

Administrative console 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 99

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 99

Replication domain

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

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

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

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

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

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

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

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

Related tasks

"Replicating data" on page 99

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 97

   Related tasks

   Configuring cache replication

   Related reference

   Internal messaging configuration settings
   Use this page to set advanced configurations for Memory to Memory session replication.

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 98

Related tasks

"Replicating data" on page 99

Related reference

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

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

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

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

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 98

Related tasks

"Replicating data" on page 99

Related reference

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

Administrative console 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 100

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 97

Related tasks

"Replicating data" on page 99

Related reference

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

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

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

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

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 97

Related tasks

"Replicating data" on page 99

Related reference

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

Administrative console 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.

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

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

Related tasks

Chapter 4, “Balancing workloads with clusters,” on page 81

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 81

WLM dynamic application environment operator commands

The dynamic application environments are displayed and controlled separately from static application environments. In order to control the dynamic environments, you must set the Resource Access Facility (RACF) server class profiles to give you the proper permission to issue the operator commands.

You can issue the following commands from the MVS console:

Display a specific dynamic application environment

`D WLM,DYNAPPL=appl_env_name`(appl_env_name is the short cluster name)

Display all dynamic application environments

```
D WLM,DYNAPPL=*
```

Restart a specific dynamic application environment

```
V WLM,DYNAPPL=appl_env_name,RESUME
```

Quiesce a specific dynamic application environment

```
V WLM,DYNAPPL=appl_env_name,QUIESCE
```

Related concepts

crun_wlmzos

crun_srvgrp

Related reference

RACF server class profiles

The Resource Access Control Facility (RACF) server class profiles are used to control dynamic application environments. Dynamic application environments are displayed and controlled separately from static application environments.

Chapter 5. Welcome to Variables

Variables in the WebSphere environment come in a variety of flavors. Variables are used to control settings and properties relating to the server environment. There are three main variable options that a WebSphere Application Server user should know and understand: custom properties, environment variables, and WebSphere-specific variables.

Environment variables

Environment variables, also called native environment variables, are not specific to the WebSphere Application Server and are defined by other elements, such as UNIX, LE, or third party vendors among others. Some of the UNIX specific native variables are LIBPATH and STEPLIB. These tend to be operating system specific.

Environment variables are specified in the administrative console by following the path **Application Server >***server_name***> Process Definition>Servant Process> Environment Entries**.

This path is also used to set environment variables that control the collection of application server and Web container information in z/OS SMF records.

WebSphere variables

WebSphere variables are used for three purposes:

1. Configuring WebSphere Application Server path names, such as JAVA_HOME, and APP_INSTALL_ROOT.
2. Configuring certain cell-wide customization values.
3. Configuring the WebSphere Application Server for z/OS Location Service.

WebSphere variables are specified in the administrative console by following the path **Environment > Manage WebSphere variables**. How the WebSphere variable is set determines its scope; whether it applies to a cell, a node, or a server. If the variable is set:

- At the server level, it applies to the entire server.
- At the node level, it applies to all servers in the node unless you set the same variable at the server level. In that case, for that server, the setting specified at the server level overrides the setting specified at the node level.
- At the cell level, it applies to all nodes in that cell, unless you set the same variable at the node or server level.
  - If you set the same variable at the server level, for that server, the setting specified at the server level overrides the setting specified at the cell level.
  - If you set the same variable at the node level, for all servers in that node, the setting specified at the node level overrides the setting specified at the cell level.

Custom properties

Custom properties are property settings meant for a specific functional component. Any configuration element can have a custom property. Common configuration

elements are cell, node, server, Web container, and transaction service. There are a limited number of supported custom properties and these can be set in the administrative console using the custom properties link associated with the functional component.

For example, to set HTTP transport custom properties, follow one of the following paths:

- **Servers > Application Servers >** *server_name* **> Web Container > HTTP Transport >Additional Properties>Custom Properties**
- **Servers > Application Servers >** *server_name* **> Web Container >Additional Properties>Custom Properties**

Custom properties set from the Web container Custom Properties page apply to all transports associated with that Web container; custom properties set from the HTTP Transport's Custom Properties page apply only to that specific transport. If the same properties are set on both pages, the settings on the Transport page overrides the settings defined on the Web container page for that specific transport.

Related tasks

"Configuring WebSphere variables" on page 146

Using the WebSphere Application Server administrative console to enable properties for specific SMF record types

Related reference

Session management custom properties

Web container custom properties
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.

Setting trace controls
Controlling behavior through timeout values

Setting trace controls for IBM service

Setting dump controls for IBM service

Related information

"Cell-wide z/OS variables" on page 149

"Repository Service Custom Properties" on page 149

"Configuring Server custom properties" on page 150

"HTTP transport custom properties" on page 49

Chapter 6. Welcome to Web servers

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

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

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

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

Chapter 7. Configuring Web server plug-ins

A WebSphere Web server plug-in can be used to forward requests for Web applications from a supported Web server to a WebSphere Application Server for z/OS Web container. A WebSphere Application Server internal HTTP transport provides the connection between the Web server plug-in and the Web container.

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

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

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

   Ensure that Web servers are configured to perform operations required by Web applications, such as GET and POST. Typically, this involves setting a directive in the Web server configuration file (such as httpd.conf for IBM HTTP Server). Refer to the Web server documentation for instructions. If an operation is not enabled when a servlet or JSP file requiring the operation is accessed, an error message displays, such as this one from IBM HTTP Server:

   IMW0093E Method POST is disabled on this server
2. If you encounter problems starting your Web server, check the plug-in log file in the WebSphere logs directory for information on what portion of the plug-in configuration file contained the error. The log file states the line number on which the error occurred along with other details that might help you diagnose why the Web server did not start. A frequent reason for a Web server not starting is an improper entry in a plug-in configuration file.
3. Install either the WebSphere HTTP Plug-in for z/OS, or a WebSphere plug-in for a Web server that is running on a distributed platform..
4. Check the version of your IBM HTTP Server installation.
5. Regenerate the plug-in configuration or manually edit the plug-in configuration file.. After changing configurations to plug-ins, transports or virtual hosts, you must regenerate your Web server plug-in for the changes to take effect.

   Related tasks

   "Configuring transports" on page 46

   Changing the HTTP plug-in configuration

plugin-cfg.xml file

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

Config (required)

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

IgnoreDNSFailures
Specifies whether the plug-in ignores DNS failures within a configuration when starting. When set to `true`, the plug-in ignores DNS failures within a configuration and starts successfully if at least one server in each ServerCluster is able to resolve the host name. Any server for which the host name can not be resolved is marked *unavailable* for the life of the configuration. No attempts to resolve the host name are made later on during the routing of requests. If a DNS failure occurs, a log message is written to the plug-in log file and the plug-in initialization continues rather than causing the Web server not to start. The default value is `false`, meaning DNS failures cause the Web server not to start.

RefreshInterval
The time interval (in seconds) at which the plug-in should check the configuration file to see if updates or changes have occurred. The plug-in checks the file for any modifications that have occurred since the last time the plug-in configuration was loaded.

In a development environment in which changes are frequent, a lower setting than the default setting of 60 is preferable. In production, a higher value than the default is preferable because updates to the configuration will not occur so often. If the plug-in reload fails for some reason, a message is written to the plug-in log file and the previous configuration is used until the plug-in configuration file successfully reloads. If you are not seeing the changes you made to your plug-in configuration, check the plug-in log file for indications of the problem.

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

The value can be `true` or `false`.

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

The value can be `true` or `false`.

ResponseChunkSize
The plug-in reads the response body in 64k chunks until all of the response data is read. This approach causes a performance problem for requests whose response body contains large amounts of data.

The ResponseChunkSize attribute allows the users to specify the maximum chunk size to use when reading the response body. For example, `<Config ResponseChunkSize="N">`, where N equals the chunk size in kilobytes.

If the content length of the response body is unknown, a buffer size of N kilobytes is allocated and the body is read in N kilobyte size chunks, until the entire body is read. If the content length is known, then a buffer size of either content length or N (whichever is less) is used to read the response body.

The default chunk size is 64k.

Log The log describes the location and level of log messages that are written by the plug-in. If a log is not specified within the configuration file, then, in some cases, log messages are written to the Web server error log.

For example, you might specify the following:

```
<Log LogLevel="Error" Name="/log_directory/filename"/>
```

Name (exactly one attribute for each Log)

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

Note: The time the information was written to the log file and the process ID will be appended to the file name specified on this element.

LogLevel (zero or one attribute for each Log)

The level of detail of the log messages that the plug-in should write to the log. You can specify one of the following values for this attribute:

- Trace. All of the steps in the request process are logged in detail.
- Warn. All warning and error messages resulting from abnormal request processing are logged.
- Error. Only error messages resulting from abnormal request processing are logged.

If a LogLevel is not specified for the Log element, the default value Error is used.

Be careful when setting the level to Trace. A lot of messages are logged at this level which can cause the file system to fill up very quickly. A Trace setting should never be used in a normally functioning environment as it adversely affects performance.

ServerCluster (one or more elements for each Config)

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

In the simplest case, the cluster contains only one server definition. In the case in which more than one server is defined, the plug-in will load balance across the defined servers using either a Round Robin or a Random algorithm. The default is Round Robin.

Following is an example of a ServerCluster element

```
<ServerCluster CloneSeparatorChange="false"
        LoadBalance="Round Robin" Name="Cluster1"
        PostSizeLimit="10000000" RemoveSpecialHeaders="true" RetryInterval="60">
<Server
CloneID="BA36BEC1EB243D8B000000E4000000030926301B"
            ConnectTimeout="0" ExtendedHandshake="false"
            LoadBalanceWeight="2" MaxConnections="0"
            Name="SY1_ClusterMember1" WaitForContinue="false">
<Transport Hostname="BOSSXXXX.PLEX1.L2.IBM.COM" Port="9084" Protocol="http"/>
<Transport Hostname="BOSSXXXX.PLEX1.L2.IBM.COM" Port="0" Protocol="https">
<Property Name="Keyring" value="/WebSphere/V5R0M0/DeploymentManager/etc/plugin-key.kdb"/>
<Property Name="Stashfile" value=""/WebSphere/V5R0M0/DeploymentManager/etc/plugin-key.sth"
<Property Name="certLabel" Value="selfsigned"/>
</Transport>
</Server>
<Server CloneID="BA36BED017FDF40E000000E4000000030926301B"
            ConnectTimeout="0" ExtendedHandshake="false"
            LoadBalanceWeight="2" MaxConnections="0"
            Name="SY1_ClusterMember2" WaitForContinue="false">
<Transport Hostname="BOSSXXXX.PLEX1.L2.IBM.COM" Port="9085" Protocol="http"/>
<Transport Hostname="BOSSXXXX.PLEX1.L2.IBM.COM" Port="0" Protocol="https">
<Property Name="Keyring" value="/WebSphere/V5R0M0/DeploymentManager/etc/plugin-key.kdb"/
<Property Name="Stashfile" value="/WebSphere/V5R0M0/DeploymentManager/etc/plugin-key.sth"/
<Property Name="certLabel" Value="selfsigned"/>
</Transport>
```

```
</Server>
<PrimaryServers>
<Server Name="Server Name="SY1_ClusterMember1"/>
<Server Name="Server Name="SY1_ClusterMember2"/>
</PrimaryServers>
</ServerCluster>
```

Note: If you are using the WebSphere HTTP Plug-in for z/OS, the `Property Name=keyring` and the `Property Name=stashfile` elements included here will be ignored if they are included in the plugin-cfg.xml file for that plug-in. The WebSphere HTTP Plug-in for z/OS uses the SSL setup specified in the hosting HTTP Server's httpd.conf file and does not look for these elements in the plugin-cfg.xml file.

Name (exactly one attribute for each ServerCluster)
The logical or administrative name to be used for this group of servers.

LoadBalance (zero or one attribute for each ServerCluster)
The default load balancing type is Round Robin.

The Round Robin implementation has a random starting point. The first server will be picked randomly. Round Robin will be used to pick servers from that point forward. This implementation ensures that in multiple process based Web servers, all of the processes don't start up by sending the first request to the same Application Server.

RetryInterval (zero or one attribute for each ServerCluster)
An integer specifying the length of time that should elapse from the time that a server is marked down to the time that the plug-in will retry a connection. The default is 60 seconds.

RemoveSpecialHeaders (zero or one attribute for each ServerCluster)
The plug-in adds special headers to the request before it is forwarded to the application server. These headers store information about the request that will need to be used by the application. By default the plug-in will remove these headers from incoming requests before adding the headers it is supposed to add.

The value can be `true` or `false`. Setting the attribute to false introduces a potential security exposure by not removing headers from incoming requests.

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

The value can be `true` or `false`.

PostSizeLimit (zero or one attribute for each ServerCluster)
The maximum number of bytes of request content allowed in order for the plug-in to attempt to send the request to an application server. If a request is received that is greater than this size, the plug-in fails the request. The default value is 10 million bytes.

Server (one or more elements for each ServerCluster)
A WebSphere Application Server instance that is configured to handle requests routed to it given the routing rules of the plug-in

configuration. The Server should correspond to an application server running on either the local machine or a remote machine.

Name (exactly one attribute for each Server)
The administrative or logical name for the server.

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

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

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

WaitForContinue (zero or one attribute for each Server)
Specifies whether to use the HTTP 1.1 100 Continue support before sending the request content to the application server. Possible attribute values are `true` or `false`. The default value is false; the plug-in does not wait for the 100 Continue response from the application server before sending the request content because it is a performance hit.

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

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

LoadBalanceWeight (zero or one attribute for each Server)
The weight associated with this server when the plug-in does weighted Round Robin load balancing. The algorithm for this attribute decrements all weights within the server cluster until all weights reach zero. Once a particular server's weight reaches zero, no more requests are routed to that server until all servers in the cluster have a weight of zero. After all servers reach zero, the weights for all servers in the cluster are reset and the algorithm starts over.

ConnectTimeout (zero or one attribute for each Server)
The ConnectTimeout attribute of a Server element enables the plug-in to perform non-blocking connections with the application server. Non-blocking connections are beneficial when the plug-in is unable to contact the destination to determine if the port is available or unavailable.

If no ConnectTimeout value is specified, the plug-in performs a blocking connect in which the plug-in sits until an operating system times out (as long as 2 minutes depending on the platform) and allows the plug-in to mark the server *unavailable*. A value of 0 causes the plug-in to perform a blocking connect. A value greater than 0 specifies the number of seconds you want the plug-in to wait for a successful connection. If a connection does not occur after that time interval, the plug-in marks the server *unavailable* and fails over to one of the other servers defined in the cluster.

ExtendedHandshake (zero or one attribute for each Server)
The ExtendedHandshake attribute is used when a proxy firewall is between the plug-in and the application server. In such a case, the plug-in is not failing over, as expected.

The plug-in marks a server as down when the connect() fails. However, when a proxy firewall is in between the plug-in and the application server, the connect() will succeed, even though the back end application server is down. This causes the plug-in to not failover correctly to other application servers.

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

The value can be `true` or `false`.

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

It is possible to configure the Server to have one non-secure transport and one that uses SSL. In this configuration, a match of the incoming request protocol will be performed to determine the appropriate transport to use to send the request to the application server.

Hostname (exactly one attribute for each Transport)
The host name or IP address of the machine on which the WebSphere application server instance is running.

Port (exactly one attribute for each Transport)
The port on which the WebSphere application server instance is listening.

Protocol (exactly one attribute for each Transport)
The protocol to use when communicating over this transport -- either HTTP or HTTPS.

Property (zero, one, or more elements for each Transport)
When the Protocol of the Transport is set to HTTPS, use this element to supply the various initialization parameters, such as password, keyring and stashfile.

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

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

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

ClusterAddress (zero or one element for each ServerCluster)
A ClusterAddress is like a Server element in that you can specify the same attributes and elements as for a Server element. The difference is that you can only define one of them within a ServerCluster. Use a ClusterAddress when you do not want the plug-in to perform any type of load balancing because you already have some type of load balancer in between the plug-in and the application server.

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

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

BackupServers (zero or one element for each ServerCluster)
Lists servers to which requests should be sent to if all servers specified in the PrimaryServers list are unavailable. The plug-in does not load balance across the BackupServers list but traverses the list in order until no servers are left in the list or until a request is successfully sent and a response received from an application server.

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

Following is an example of a VirtualHost Group element and associated elements and attributes

```
<VirtualHostGroup Name="Hosts">
<VirtualHost Name="www.x.com"/>
<VirtualHost Name="www.x.com:443"/>
<VirtualHost Name="*:8080"/>
<VirtualHost Name="www.x.com:*"/>
<VirtualHost Name="*:*"/>
</VirtualHostGroup>
```

Name (exactly one attribute for each VirtualHostGroup)
The logical or administrative name to be used for this group of virtual hosts.

VirtualHost (one or more elements for each VirtualHostGroup)
The name used for a virtual or real machine used to determine if incoming requests should be handled by WebSphere Application Server or not. Use this element to specify host names that will be

in the HTTP Host header which should be seen for requests that need to be handled by the application server. You can specify specific host names and ports that incoming requests will have or specify an * for either the host name, port, or both.

Name (exactly one attribute for each VirtualHost)
The actual name that should be specified in the HTTP Host header in order to match successfully with this VirtualHost.

The value is a host name or IP address and port combination, separated by a colon.

You can configure the plug-in to route requests to the application server based on the incoming HTTP Host header and port for the request. The Name attribute specifies what those combinations are.

You can use a wildcard for this attribute. The only acceptable solutions are either an * for the host name, a * for the port, or a * for both. A * for both means that any request will match this rule. If no port is specified in the definition the default HTTP port of 80 is assumed.

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

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

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

Name (exactly one attribute for each UriGroup)
The logical or administrative name for this group of URIs.

Uri (one or more elements for each UriGroup)
The virtual path to the resource that will be serviced by WebSphere Application Server. Each URI specifies the incoming URLs that need to be handled by the application server. You can use a wildcard in these definitions.

Name (exactly one attribute for each Uri)
The actual string that should be specified in the HTTP request line in order to match successfully with this URI. You can use a wildcard within the URI definition. You can specify rules such as *.jsp or /servlet/* to be handled by WebSphere Application Server. When you assemble your application, if you specify **File Serving Enabled** then only a wildcard URI is generated for the Web application, regardless of any explicit servlet mappings. If you specify **Serve servlets by classname** then a URI having `<Uri Name="Web_application_URI/servlet/*">` is generated.

AffinityCookie (zero or one attribute for each Uri)
The name of the cookie the plug-in should use when trying to determine if the inbound request has session affinity. The default value is **JSESSIONID**. (See the description of the CloneID attribute for additional session affinity information.)

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

The route definition is the central element of the plug-in configuration. It specifies how the plug-in will handle requests based on certain characteristics of the request. The route definition contains the other main elements: a required ServerCluster, and either a VirtualHostGroup, UriGroup, or both.

Using the information that is defined in the VirtualHostGroup and the UriGroup for the route, the plug-in determines if the incoming request to the Web server should be sent on to the ServerCluster defined in this route.

Following is an example of this element:

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

VirtualHostGroup (zero or one attribute for each Route)
The group of virtual hosts that should be used in route determination. The incoming host header and server port are matched to determine if this request should be handled by the application server.

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

UriGroup (zero or one attribute for each Route)
The group of URIs to use for determining the route. The incoming URI for the request is matched to the defined URIs in this group to determine if this request should be handled by the application server.

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

ServerCluster (exactly one attribute for each Route)
The cluster to which to send request that successfully match the route.

The cluster that should be used to handle this request. If both the URI and the virtual host matching is successful for this route then the request is sent to one of the servers defined within this cluster.

Related tasks

"Manually editing the plug-in configuration" on page 127

"Installing a distributed platform Web server plug-in" on page 124

"Installing the WebSphere HTTP Plug-in for z/OS" on page 122

Web server plug-ins

Web server plug-ins enable the Web server to communicate requests for dynamic content, such as servlets, to the application server.

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

You can install WebSphere Application Server plug-ins for your Web servers after you finish installing the application server.

Installing the WebSphere HTTP Plug-in for z/OS

The WebSphere HTTP Plug-in for z/OS ships as part of the WebSphere Application Server for z/OS product. To use this plug-in, you must have an IBM HTTP Server for z/OS or OS/390 Version 5.3 configured as part of a z/OS system.

After the application server, the HTTP Server and the plug-in are properly configured:

- WebSphere Application Server for z/OS can use this plug-in to perform regular plug-in functions.
- Requests can be routed from a browser, through the HTTP Server and plug-in, to one of the application server J2EE server instances defined in the ServerGroup element in the `plugin-cfg.xml` file. (New requests are sent to randomly selected server instances, but once a session is established, requests get routed back to the server instance assigned to the original request.)
- Private headers can be used as a mechanism for forwarding proxy information from this plug-in to a WebSphere Application Server for z/OS Web container. (This information is not otherwise included with HTTP requests.)

1. Make sure a Version 5.3 IBM HTTP Server for z/OS and OS/390 is installed on a z/OS or OS/390 system.
2. Use FTP or another file transfer mechanism to download, in binary format, the WebSphere HTTP Plug-in for z/OS from your WebSphere Application Server for z/OS system to that HTTP Server and place it in the *webserver_install_root*/bin/ directory. The ihs390WAS50Plugin_http.so DLL is located in the application server's /*install_root*/bin/ directory.
3. Using an authorized z/OS user ID, issue the following commands from an OMVS command line prompt to turn on the "p" bit in the HFS where the WebSphere HTTP Plug-in for z/OS is now located:

   ```
   chmod 777 ihs390WASPlugin_http.so
   extattr +p ihs390WASPlugin_http.so
   ```

4. Add ServerInit, Service, and ServerTerm directives to the `httpd.conf` configuration file of the HTTP Server:
   - The following ServerInit and ServerTerm directives indicate the entry points to the plug-in's initialization and exit routines. These routines exist as entry points init_exit, and term_exit, respectively, within the `ihs390WAS50Plugin_http.so` DLL file.

     ```
     ServerInit /webserver_install_root/bin/
         ihs390WAS50Plugin_http.so:init_exit fully_qualified_path
     ServerTerm /webserver_install_root/bin/ihs390WAS50Plugin_http.so:term_exit
     ```

     webserver_install_root is the name of the root directory where the Web server is installed, and *fully_qualified_path* is the fully qualified path to the plugin-cfg.xml file.
   - The following Service directive for each application that will be using the Web server plug-in. This directive indicates the entry point to the plug-in's request routine. The request routine exists as the entry point service_exit within the `ihs390WAS50Plugin_http.so` Dynamic Link Library (DLL) file.

     ```
     Service /webapp_contextroot/*  /webserver_install_root/bin/
        ihs390WAS50Plugin_http.so:service_exit
     ```

     webapp_contextroot is the context root of the Web application, and *webserver_install_root* is the name of the root directory where the Web server is installed.

Notes:

a. In this discussion, the ServerInit and Service directives are split for printing purposes. In the actual `httpd.conf` file, enter each of these directives on a single line.
b. The HTTP Server interprets a blank in a directive specification as a delimiter and a number sign (#) as the beginning of a comment that should be ignored. Therefore, if you need to use a blank or number sign in a directive, you must include a backslash (\) before the blank or number sign to enable the HTTP Server to correctly process the directive.
c. If a servlet sets an HTTP response code by any means, such as using methods lastModified() or setStatus(), and the client does not receive the expected response code, add the following directive to the HTTP Server configuration file:

   ```
   ServiceSync On
   ```
d. If you want to use Secure-socket layer (SSL), make sure that the HTTP Server is configured for SSL. The SSL connection of the Web server plug-in uses the SSL session established by the HTTP Server. (See *z/OS HTTP Server Planning, Installing, and Using, Version 5.3*, SC34-4826, for a description of how to configure the HTTP Server for SSL.)

5. Configure the plug-in. Use either use the **Update Web Server Plug-in Configuration** page in the administrative console, or issue the GenPluginCfg.sh script to create your `plugin-cfg.xml` file.

   Both methods will create the plug-in configuration file, `plugin-cfg.xml`, in EBCDIC format, which is the proper format for execution in a z/OS environment.

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

   a. Go to the Update Web Server Plug-in page. Click **Environment > Update Web Server Plugin** in the console navigation tree.
   b. Click **OK**.
   c. You might need to stop the application server and then start the application server again to enable the Web server to locate the `plugin-cfg.xml` file.
6. Using the administrative console, make sure the virtual host is configured with an alias for the port number used by the z/OS V5.3 HTTP Server. Specify the same port on a <VirtualHost Name=> element in the plugin-cfg.xml file.
7. If you intend to use private headers, using the administration console, set the TrustedProxy property for the transport to true. This setting enables the transport to trust private headers received from any supported Web server plug-in.

   You can set the TrustedProxy property to true from either the Web container **Custom Properties** page or the HTTP Transport **Custom Properties** page. If you set it on the Web container **Custom Properties** page, all transports will support private headers.

   Note: If you try to use private headers without adding the TrustedProxy property, they will be ignored. If the private headers are ignored, the application server might not locate the requested application.

   After you add this setting, the transport(s) trusts all private headers it receives. Therefore, you must ensure that all paths to the HTTP or HTTPS transport are trusted.
8. Stop the application server and the HTTP Server and start them again.

The configuration is complete. To activate the configuration, stop and restart both the application server and the HTTP Server. If the WebSphere HTTP Plug-in for z/OS comes up when the HTTP Server starts again, you receive the following messages:

```
WebSphere HTTP Plug-in for z/OS Version 5.00 Service Level 0 is starting
 WebSphere HTTP Plug-in for z/OS initializing with configuration file :
         fully_qualified_path_to_the_plugin-cfg.xml_file
WebSphere HTTP Plug-in for z/OS initialization went OK :-)
```

Related tasks

Modifying the default Web container configuration

"Configuring virtual hosts" on page 141

"Regenerating Web server plug-in configurations" on page 130

Related reference

"HTTP transport custom properties" on page 49

"plugin-cfg.xml file" on page 113

Private headers

Web server plug-ins enable the Web server to communicate requests for dynamic content, such as servlets, to the application server.

Once you configure a WebSphere plug-in for Web servers, in addition to regular plug-in functions, you can use private headers as a mechanism for forwarding proxy information from these plug-ins to the application server on a z/OS platform. This information is not otherwise included with the HTTP requests.

Private headers are implemented as a set of HTTP request header name and value pairs that the plug-ins add to the HTTP request header as it is forwarded by the Web server. The application server Web container removes this information from the header and processes it.

Private headers can include such information as the remote (client) user, the remote (client) host name, or an SSL client certificate. They conform to a naming standard so that there is no namespace collision with the architected HTTP header fields (hence the name "private").

For example, authentication information, such as a client certificate, is normally requested by the Web server once during the establishment of an HTTP session. It is not required again for individual requests within that session. However, a client certificate must accompany each request forwarded to the application server on a z/OS platform so that the application server can use it as needed.

Similarly, the Web server examines the TCP/IP socket connection for information about the host address of the client. The application server cannot do this examination because its socket connection is with the plug-in, not the actual client. Therefore, one of the private headers is the host address of the actual client.

Related tasks

"Installing a distributed platform Web server plug-in"

Installing a distributed platform Web server plug-in

Web server plug-ins for distributed platform Web servers are no longer shipped with the WebSphere Application Server for z/OS product. Use the installation CD for WebSphere Application Server or WebSphere Application Server Network Deployment V5 for distributed platforms to install these Web server plug-ins.

Note: If you do not have a copy of Version 5 of the WebSphere Application Server or WebSphere Application Server Network Deployment for distributed platforms product, you must request a not-for-resale copy from IBM.

Once you have the application server, the Web server and the plug-in properly configured, requests can be routed from a browser, through that Web server and plug-in, to one of the application server J2EE server instances defined in the ServerGroup element in the `plugin-cfg.xml` file. New requests are sent to randomly selected server instances, but once a session is established, requests will get routed back to the correct HTTP or HTTPS internal transport the J2EE server Web container assigned to the original request.

WebSphere plug-ins for Web servers, in addition to regular plug-in functions, enable you to use private headers as a mechanism for forwarding proxy information from these plug-ins to the WebSphere Application Server for z/OS Web container. This information is not otherwise included with the HTTP and HTTPS requests.

1. Make sure a supported distributed platform Web server is installed on your workstation. If you need to install a Web server, follow the instructions provided with that Web server.

   Note: If you have WebSphere Application Server Network Deployment V5 or later for distributed platforms installed on your workstation, you can use the IBM HTTP Server that is provided with that product, and the Web server plug-in that is appropriate for that Web server.
2. Install the appropriate Web server plug-in. Using the installation CD for WebSphere Application Server or WebSphere Application Server Network Deployment V5 for distributed platforms, select to perform a custom installation, and then under ″Web Server Plugins″ select the appropriate Web server plug-in for your system.
3. Re-configure the Web server.

   Once the plug-in files are installed on the Web server, update the Web server configuration file on the workstation with the location of the plug-in and plug-in configuration file. For Windows systems, also add the location of the common file, `plugin_common.dll`, to the PATH statement.

   Specific instructions for configuring plug-ins within a Web server configuration file are contained in the documentation for each Web server.
4. Go to the WebSphere Application Server for z/OS administrative console and make sure the virtual host contains an alias for the port number used by the Web server. Specify this same port on a <Virtual Hostname> element in the plug-in `plugin-cfg.xml` file.
5. Configure the plug-in. Use either use the Update Web Server Plug-in Configuration page in the administrative console, or issue the GenPluginCfg.sh script to create your `plugin-cfg.xml` file.

   Both methods will create the plug-in configuration file, `plugin-cfg.xml`, in EBCDIC format, which is the proper format for execution in a z/OS environment.

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

   a. Go to the Update Web Server Plug-in page. Click **Environment > Update Web Server Plugin** in the console navigation tree.
   b. Click **OK**.

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

6. If you intend to use private headers, using the WebSphere Application Server for z/OS administrative console, set the TrustedProxy custom property for the internal tranport to true. This setting enables the HTTP internal transport to trust private headers received from any WebSphere plug-in for Web servers.

   You can set the TrustedProxy property to true from either the Web container **Custom Properties** page or the HTTP Transport **Custom Properties** page. If you set it on the Web container **Custom Properties** page, all transports will support private headers.

   Note: If you try to use private headers without adding the TrustedProxy property, they will be ignored. If the private headers are ignored, the application server might not locate the requested application.

   After you add this setting, the HTTP or HTTPS internal transport trusts all private headers it receives. Therefore, you must ensure that all paths to the HTTP or HTTPS transport are trusted.

7. If you want to use Secure-socket layer (SSL) with this configuration, install the appropriate GSKIT installation image file on your workstation using the native install process for that platform.

The configuration is complete. To activate the configuration, stop and restart both the WebSphere Application Server for z/OS and the Web server that is running on your workstation.

Related concepts

"Private headers" on page 124

Related tasks

Modifying the default Web container configuration

"Configuring virtual hosts" on page 141

"Installing a Global Security Kit for a distributed platform Web server plug-in" on page 132

"Manually editing the plug-in configuration" on page 127

Related reference

"HTTP transport custom properties" on page 49

Supported distributed platform Web server plug-in configurations

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

Operating system	Web server	Plug-in executable file
Windows 2000 and Windows NT	IBM HTTP Server for distributed platforms	mod_ibm_ app_server_http.dll
	Lotus Domino	libdomino5_http.dll
	Apache	app_server_http.dll
	iPlanet (Netscape)	libns41_http.dll
	Microsoft Internet Information Server (IIS)	iisWASPlugin _http.dll

IBM AIX	IBM HTTP Server for distributed platforms	mod_ibm_app_ server_http.so
	Lotus Domino	libdomino5_ http.a
	Apache	mod_app_ server_http.so
	iPlanet (Netscape)	libns41_http.so
HP-UX	IBM HTTP Server for distributed platforms	mod_ibm_app_ server_http.sl
	Lotus Domino	libdomino5 _ http.sl
	Apache	mod_app_ server_http.sl
	iPlanet (Netscape)	libns41_http.sl
Solaris Operating Environment	IBM HTTP Server for distributed platforms	mod_ibm_app_ server_http.so
	Lotus Domino	libdomino5 _ http.a
	Apache	/mod_app_ server_http.so
	iPlanet (Netscape)	libns41_ http.so
Linux	IBM HTTP Server for distributed platforms	mod_ibm_app_ server_http.so
	Apache	mod_app_ server_http.so

Related tasks

Configuring high-speed external caching through the Web server

Checking your IBM HTTP Server version

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

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

   ```
   apache.exe -V
   ```

4. On a UNIX-based platforms issue:

   ```
   ./apachectl -V 4
   ```

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

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

Manually editing the plug-in configuration

Regenerating the plug-in configuration file does not guarantee a correct configuration for advanced configuration scenarios. If your use of the WebSphere Application Server product requires a plug-in configuration more advanced than what is provided with the default plug-in configuration, or if the plug-in does not

behave as properly after the configuration file is regenerated, you can manually edit the configuration file to obtain the proper plug-in behavior for your environment.

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

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

   CAUTION: If the plug-in configuration is regenerated, your manual edits to the plug-in configuration file will be overwritten. Therefore, you should maintain a record of any manual changes you make for future reference.
3. If you edited the plugin-cfg.xml files on federated nodes, turn off automatic synchronization of nodes on the File Synchronization Service page to prevent the edited files from being overwritten. Because a plugin-cfg.xml file on a Network Deployment machine is stored in the configuration repository, the plugin-cfg.xml file is overwritten on all federated nodes in your network whenever the Network Deployment machine updates the configuration repository on these nodes. If you need to regenerate the plugin-cfg.xml file on the Network Deployment machine:
   a. Save your edited plugin-cfg.xml files on the federated nodes.
   b. Force node synchronization of each federated node.
   c. Replace the updated plugin-cfg.xml file with the saved copy or merge the customized pieces of the saved copy into the new plugin-cfg.xml file.

Related reference

"plugin-cfg.xml file" on page 113

Situations requiring manual editing of the plug-in configuration

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

IBM WebSphere Application Server for z/OS integrates the WebSphere Application Server product and the Network Deployment product into a single package. When you see WebSphere Application Server product or Network Deployment product in reference to WebSphere Application Server for z/OS, the term product equates to function.

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

- If the Web server and `plugin-cfg.xml` file are installed on a separate remote system, you must change the paths in `plugin-cfg.xml` file if:
  - The plug-in was generated on a Windows 32 system platform. Copy it to a remote Linux or UNIX system with an HTTP Server and a WebSphere Application Server Version 5 plug-in.
  - The plug-in was generated on a Linux or UNIX system. Copy it to a remote Windows platform with an HTTP Server and a WebSphere Application Server Version 5 plug-in.

  - The plug-in was generated on a Linux or UNIX system and needs to be copied to another, remote Linux or UNIX system that has a different configuration. For example, the plug-in was generated on a system having a single-server (Base) or Network Deployment installation on AIX in the default path, and the remote HTTP Server and plug-in are installed on a Solaris or Linux system with the plug-in installed in the default location.

- In a Network Deployment environment, if the single-server (base) product, Network Deployment product, plug-in, and HTTP Server are all installed on the same node, change entries in the `plugin-cfg.xml` file that refer to the `/config/cells` directory in the installation root of the deployment manager to instead, refer to the `/config/cells` directory in the installation root of the base Application Server. References exist to the `/config/cells` directory of the installation root of the deployment manager because the deployment manager runs the plugin-update procedure.

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

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

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

Related reference

"plugin-cfg.xml file" on page 113

Related information

Regenerating Web server plug-in configurations

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

CAUTION: Regenerating the plug-in configuration can overwrite manual configuration changes that you might want to preserve. Before performing this task, understand its implications as described in Chapter 7, "Configuring Web server plug-ins," on page 113.

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

```
WAS_HOME/AppServer/GenPluginCfg.sh
```

WAS_HOME is the root directory for your installation of IBM WebSphere Application Server.

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

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

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

Regenerating the configuration might take a while to complete. After it finishes, all objects in the administrative cell use their newest settings, which the Web server can access. Whether triggered manually or occurring automatically, plug-in regeneration requires about 30 to 60 seconds to complete when the Application Server is on the same physical machine (node) as the Web server. In other cases, it takes more time.

The delay is important because it determines how soon the new plug-in configuration takes effect. Suppose you add a new served path for a servlet, then regenerate the plug-in configurations. The regeneration requires 40 seconds, after which a user should be able to access the servlet by the new served path.

For an HTTP plug-in, the length of the delay is determined by the Refresh Interval attribute of the Config element in the `plugin-cfg.xml` file. The plug-in polls the file system at this interval to see whether the configuration has changed. The default interval is 60 seconds. To regenerate the plug-in configuration requires twice the refresh interval.

In a development environment in which you are frequently changing settings in the administrative console, it is recommended that you set the refresh interval to 3 to 5 seconds.

In a production environment, set a longer refresh interval, perhaps as long as 30 minutes, depending on the frequency of changes.

When the deployment manager is installed on a machine that is remote from the base WebSphere Application Server installation, one of the following solutions must be implemented in order for the `plugin-cfg.xml` file to retain the *WAS_HOME*`/AppServer` directory structures, and not assume those of the *WAS_HOME*`/DeploymentManager`, after a regeneration of the plug-in and full synchronization. The `plugin-cfg.xml` file is located in the *WAS_HOME*`/AppServer/config/cells` directory.

- **Command line**:

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

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

  ```
  GenPluginCfg -destination.root "/WebSphere/V5R0M0/AppServer"
  ```

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

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

  Perform a full synchronization so the `plugin-cfg.xml` file is replicated in all the WebSphere Application Server nodes. You can use scripting to perform a node synchronization or you can use the administrative console. (See *WebSphere Application Server for z/OS V5.0.2: System Administration* or the WebSphere Application Server V5 InfoCenter for more information on how to perform this task.

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

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

Related tasks

Changing the WebSphere Application Server configuration using wsadmin

Managing nodes

GenPluginCfg command reference

Purpose

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

CAUTION: Regenerating the plug-in configuration can overwrite manual configuration changes that you might want to preserve. Before performing this task, understand its implications as described in Chapter 7, “Configuring Web server plug-ins,” on page 113.

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

```
WAS_HOME/AppServer/GenPluginCfg.sh
```

WAS_HOME is the root directory for your installation of IBM WebSphere Application Server.

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

Related tasks

Chapter 7, “Configuring Web server plug-ins,” on page 113

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

In addition to the plug-in files, there is another software package provided with WebSphere Application Server for z/OS that must be installed on the Web server that is running on your workstation. This package is called the Global Security Kit (GSKit). It helps connect this Web server to the WebSphere Application Server for z/OS product. This package is required if the Secure Sockets Layer (SSL) Transport (also known as HTTPS) is used.

There is one GSKit installation image per platform. GSKit is the same for all Web servers running on that platform.

1. On your workstation, add the GSKit installation directory to the Web server’s PATH statement.
2. Using FTP or another file transfer mechanism, download, in binary format, the appropriate GSKit installation image file from the WebSphere Application Server for z/OS */install_root*`/DownloadPlugins` directory.

   When downloading these files, remove the `"_bin"` suffix from the file name. It is included as a reminder that these files are in binary format. For example, when downloading the `gskkm.rte_bin` file for the AIX GSKit installation image residing on AIX, use `gskkm.rte` as the file name.
3. Using the native install process for the operating system you are running on your workstation, install this file onto the Web server. For example, DSMIT should be run on AIX, or the gsk5bas.exe, which invokes InstallShield, should be run on a Windows system.
4. If you intend to use Secure-socket layer SSL:
   - Configure the Web server for SSL support. (See your Web server documentation for a description of how to configure SSL for your specific Web server.)

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

     ```
     LoadModule ibm_ssl_module modules/IBMModuleSSL128.dll
     Listen port_number
     Keyfile C:\ssl\http_session\plug-inKeys.kdb
     ```

```
<VirtualHost virtual_host_name:port_number>
      ServerName virtual_host_name
      SSLEnable
      SSLClientAuth none
</VirtualHost>
```

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

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

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

- Configure an HTTPS internal transport to listen on the port the Web server plug-in is using to redirect requests to the WebSphere Application Server for z/OS Web container. Specify this same port on a <Transport Hostname> element in the plug-in `plugin-cfg.xml` file. Use the administrative console to determine the port on which the internal transport is listening.
- Create an SSL key file for the Web server plug-in.

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

  a. Create an SSL key file without the default signer certificates.
     1) Start IKeyMan. (See *WebSphere Application Server for z/OS V5.0.2: Security* or the WebSphere Application Server V5 InfoCenter for a description of how to perform this task.)
     2) Create a new key database file. Click **Key Database File** > **New**. Then specify settings:
        – Key database type: **JKS**
        – File Name: **appServerKeys.jks**
        – Location: *your myKeys directory*, such as */install_root*/myKeys

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

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

          Click **OK**.

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

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

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

   The plug-inKeys.kdb file is the key database file, created using the z/OS System SSL gskkyman utility, that contains the public keys, private keys, trusted CAs, and certificates for the Web server plug-ins.
4) Enter the associated password and click **OK**.
5) Click **Personal Certificates** > **Signer Certificates**.
6) Click **Add**. Then specify settings:
   - Data Type: **Base64-encoded ASCII data**
   - Certificate File Name: **appServer.arm**
   - Location: *path_to_myKeys_directory*
7) Click **OK**.
8) Click **Key Database File** > **Exit**.

c. Reference the key file in the administrative console.

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

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

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

```
<ServerCluster CloneSeparatorChange="false"
        LoadBalance="Round Robin" Name="Cluster1"
        PostSizeLimit="10000000" RemoveSpecialHeaders="true"
             RetryInterval="60">
<Server
CloneID="BA36BEC1EB243D8B000000E4000000030926301B"
            ConnectTimeout="0" ExtendedHandshake="false"
            LoadBalanceWeight="2" MaxConnections="0"
            Name="SY1_ClusterMember1" WaitForContinue="false">
<Transport Hostname="BOSSXXXX.PLEX1.L2.IBM.COM" Port="9084" Protocol="http"/>
<Transport Hostname="BOSSXXXX.PLEX1.L2.IBM.COM" Port="0" Protocol="https">
<Property Name="Keyring" value="/WebSphere/V5R0M0/DeploymentManager/etc/
            plugin-key.kdb"/>
<Property Name="Stashfile" value=""/WebSphere/V5R0M0/DeploymentManager/etc/
```

```
            plugin-key.sth"/>
<Property Name="certLabel" Value="selfsigned"/>
</Transport>
</Server>
<Server CloneID="BA36BED017FDF40E000000E4000000030926301B"
            ConnectTimeout="0" ExtendedHandshake="false"
            LoadBalanceWeight="2" MaxConnections="0"
            Name="SY1_ClusterMember2" WaitForContinue="false">
<Transport Hostname="BOSSXXXX.PLEX1.L2.IBM.COM" Port="9085" Protocol="http"/>
<Transport Hostname="BOSSXXXX.PLEX1.L2.IBM.COM" Port="0" Protocol="https">
<Property Name="Keyring" value="/WebSphere/V5R0M0/DeploymentManager/etc/
             plugin-key.kdb"/
<Property Name="Stashfile" value="/WebSphere/V5R0M0/DeploymentManager/etc/
             plugin-key.sth"/>
<Property Name="certLabel" Value="selfsigned"/>
</Transport>
</Server>
<PrimaryServers>
<Server Name="Server Name="SY1_ClusterMember1"/>
<Server Name="Server Name="SY1_ClusterMember2"/>
</PrimaryServers>
</ServerCluster>
```

where:

- `plug-inKeys.kdb` is the key database file created using the z/OS System SSL gskkyman utility, that contains the public keys, private keys, trusted CAs, and certificates for the Web server plug-ins.file containing the keys for the plug-ins.
- `plug-inpw.sth` is the stash file in which the z/OS System SSL gskkyman utility stored the encrypted database password for these certificates.

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

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

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

   Related tasks

   "Installing a distributed platform Web server plug-in" on page 124

Gskit install images files

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

Operating system	GSKit Installation image file
Windows 2000 and Windows NT	*install_root*/ Download Plugins/ Win32gsk5bas.exe_bin
AIX	*install_root*/ DownloadPlugins/AIX/ gsk/gskkm.rte_bin
HP-UX	*install_root*/ DownloadPlugins/HPUX/ gsk/gsk5bas.tar.Z_bin
Solaris Operating Environment	*install_root*/ DownloadPlugins/Solaris/ gsk/gsk5bas.tar.Z_bin
Linux	*install_root*/ DownloadPlugins/LINUX/ gsk/gsk5bas-5.0-4.79. i386.rpm_bin

Related tasks

"Installing a distributed platform Web server plug-in" on page 124

Chapter 8. Welcome to Cell-wide settings

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

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

Chapter 9. Configuring the cell-wide environment

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

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

Virtual hosts

A virtual host is a configuration enabling a single host machine to resemble multiple host machines. Resources associated with one virtual host cannot share data with resources associated with another virtual host, even if the virtual hosts share the same physical machine.

Each virtual host has a logical name and a list of one or more DNS aliases by which it is known. A DNS alias is the TCP/IP hostname and port number used to request the servlet, for example yourHostName:80. When no port number is specified, 80 is assumed.

When a servlet request is made, the server name and port number entered into the browser are compared to a list of all known aliases in an effort to locate the correct virtual host and serve the servlet. If no match is found, an error is returned to the browser.

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

A virtual host is not associated with a particular node (machine). It is a configuration, rather than a "live object," explaining why you can create it, but cannot start or stop it. For many users, creating virtual hosts is unnecessary because the default_host is provided.

Adding a localhost to the virtual hosts adds the host name and IP address of the localhost machine to the alias table. This allows a remote user to access the administrative console.

Related tasks

"Configuring virtual hosts" on page 141

Why and when to use virtual hosting

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

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

Now suppose both company's sites offer the same servlet. Each site has its own instances of the servlet, which are unaware of the other site's instances. If the company whose site is organized on VirtualHost2 is past due in paying its account with the ISP, the ISP can refuse all servlet requests that are routed to VirtualHost2. Even though the same servlet is available on VirtualHost1, the requests directed at VirtualHost2 will not be routed there.

The servlets on one virtual host do not share their context with the servlets on the other virtual host. Requests for the servlet on VirtualHost1 can continue as usual, even though VirtualHost2 is refusing to fill requests for the same servlet.

Related tasks

"Configuring virtual hosts" on page 141

The default virtual host (default_host)

The product provides a default virtual host (named default_host).

The virtual host configuration uses wildcard entries with the ports for its virtual host entries.

- The default alias is *:80, using an internal port that is not secure.
- Aliases of the form *:9080 use the secure internal port.
- Aliases of the form *:9443 use the external port that is not secure.
- Aliases of the form *:443 use the secure external port.

Unless you specifically want to isolate resources from one another on the same node (physical machine), you probably do not need any virtual hosts in addition to the default host.

Related tasks

"Configuring virtual hosts" on page 141

How requests map to virtual host aliases

When you request a resource, WebSphere Application Server tries to map the request to an alias of a defined virtual host.

Mappings are both case sensitive and insensitive. For example, the portion "http://host:port/" is case insensitive, but the URL that follows is case sensitive. The match must be alphanumerically exact. Also, different port numbers are treated as different aliases.

For example, the request `http://www.myhost.com/myservlet` maps successfully to `http://WWW.MYHOST.COM/myservlet` but not to `http://WWW.MYHOST.COM/MYSERVLET` or `Www.Myhost.Com/Myservlet`. In the latter two cases, these mappings fail due to case sensitivity. The request `http://www.myhost.com/myservlet` does not map successfully to `http://myhost/myservlet` or to `http://myhost:9876/myservlet`. These mappings fail because they are not alphanumerically correct.

You can use wildcard entries for aliases by port and specify that all valid hostname and address combinations on a particular port map to a particular virtual host.

If you request a resource using an alias that cannot be mapped to an alias of a defined virtual host, you receive a 404 error in the browser used to issue the request. A message states that the virtual host could not be found.

Related tasks

"Configuring virtual hosts" on page 141

Configuring virtual hosts

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

1. Create a virtual host using the Virtual Hosts page of the administrative console. Click **Environment > Virtual Hosts** from the navigation tree of the console, click **New** and, on the settings page for a virtual host that displays, specify an administrative name for the virtual host. When you create a virtual host, a default set of 90 MIME entries is created for the virtual host.
2. Determine whether you need a virtual host alias for each HTTP transport port. There must be a virtual host alias corresponding to each port used by an HTTP transport. There is one HTTP transport in each Web container, with one Web container in each application server.

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

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

Related concepts

"Virtual hosts" on page 139

"Transports" on page 46

Related tasks

Exporting applications

Virtual host collection

Use this page to manage virtual hosts.

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

Related concepts

"Virtual hosts" on page 139

Related tasks

"Configuring virtual hosts" on page 141

Related reference

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

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

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

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

Name

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

Virtual host settings

Use this page to configure a virtual host instance.
To view this administrative console page, click **Environment > Virtual Hosts >***virtual_host_name*.

Related concepts

"Virtual hosts" on page 139

Related tasks

"Configuring virtual hosts" on page 141

Related reference

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

Administrative console 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.

"Host alias collection" on page 143

"MIME type collection" on page 144

Name:

Specifies a logical name for configuring Web applications to a particular host name. The default virtual host is suitable for most simple configurations.

Data type	String
Default	default_host

Host alias collection

Use this page to manage host name aliases defined for a virtual host. An alias is the DNS host name and port number that a client uses to form the URL request for a Web application resource.

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

Related concepts

"Virtual hosts" on page 139

"How requests map to virtual host aliases" on page 140

Related tasks

"Configuring virtual hosts" on page 141

Related reference

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

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

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

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

Host Name:

Specifies the IP address, DNS host name with domain name suffix, or just the DNS host name, used by a client to request a Web application resource (such as a servlet, JSP, or HTML page). For example, the host alias name is `myhost` in a DNS name of `myhost:8080`.

Port:

Specifies the port for which the Web server has been configured to accept client requests. For example, the port assignment is *8080* in a DNS name of `myhost:8080`. A URL refers to this DNS as: http://*myhost:8080*/servlet/snoop.

Host alias settings:

Use this page to view and configure a host alias.

To view this administrative console page, click **Environment > Virtual Hosts >***virtual_host_name* **> Host Aliases >***host_alias_name*.

Related concepts

"How requests map to virtual host aliases" on page 140

Related tasks

"Configuring virtual hosts" on page 141

Related reference

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

Administrative console 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.

Host Name:

Specifies the IP address, DNS host name with domain name suffix, or the DNS host name that clients use to request a Web application resource, such as a servlet, JSP file, or HTML page.

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

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

Data type	String
Default	*

Port:

Specifies the port where the Web server accepts client requests. Specify a port value in conjunction with the host name.

The default reflects the value specified at product setup. The default might be `80`, `81`, `9080` or a similar value.

Data type	Integer
Default	80

MIME type collection

Use this page to view and configure Multi-Purpose Internet Mail Extensions (MIME) object types and their file name extensions.
The list shows a collection of MIME type extension mappings defined for a virtual host. Virtual host MIME entries apply when you do not specify MIME entries at the Web module level.

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

Related concepts

"Virtual hosts" on page 139

Related tasks

"Configuring virtual hosts" on page 141

Related reference

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

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

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

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

MIME Type:

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

Extensions:

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

MIME type settings:

Use this page to configure a Multi-Purpose Internet Mail Extensions (MIME) object type.

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

Related concepts

"Virtual hosts" on page 139

Related tasks

"Configuring virtual hosts" on page 141

Related reference

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

Administrative console 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.

MIME Type:

Specifies a MIME type, which can be text, image, or application.

An example value for MIME type is `text/html`. A default value appears only if you are viewing the configuration for an existing instance.

Data type	String

Extensions:

Lists file extensions of files that map the MIME type.

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

instance.

Data type String

Variables

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

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

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

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

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

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

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

Related tasks

"Configuring WebSphere variables"

IBM Toolbox for Java JDBC driver

Configure and use the jt400.jar file

Configuring WebSphere variables

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

1. Click **Environment > Manage WebSphere Variables** in the console navigation tree. On the WebSphere Variables page, click **New**.
2. Specify the scope of the new variable. Indicate if new variable should be for the **Cell**, **Node**, or **Server** and click **Apply**. The scope control enables you to choose the variable level in which you wish to work. Choosing to apply to a cell, node or server allows you to put the variable on all servers at that particular level. If

you specify the same variable on a cell or node as on a server, the server level variable overrides the cell or node variable. Similarly, a variable at the node level overrides the instance specified at the cell level.

3. On the Variable page, specify a name and value for the variable. So other people can understand what the variable is used for, also specify a description for the variable. Then click **OK**.
   - Use cell-wide customization values.
   - You can use WebSphere variables to modify the daemon configuration. By appending a server custom property onto a daemon tag, you can designate that variable specifically for that daemon. Enter `DAEMON_<server custom property>` in the **Name** field. For example, if you enter `DAEMON_ras_trace_outputlocation` in the Name field and `SYSOUT` in the Value field, you can direct that particular daemon's trace output to SYSPRINT.
   - WebSphere variables support substitution. The name of a variable can be formed by substituting the value of another variable. If you enter `${<variable name>}` in the **Name** field, the value of `<variable name>` will be the name of your new WebSphere variable. For example `${JAVA_HOME}` will create a WebSphere variable with a name that is equal to the Java home directory.
   - There are also WebSphere internal variables. The application server uses these variables for its own purposes. The prefixes that indicate that a variable is WebSphere internal are `WAS_DAEMON_<server custom property>`, `WAS_DAEMON_ONLY_<server custom property>`, and `WAS_SERVER_ONLY_<server custom property>`. Any variables with these tags are not intended for your use. They are reserved exclusively for use by the server runtime. Modifying these variables may cause unexpected errors.
4. Verify that the variable is shown in the list of variables.
5. Save your configuration.
6. To have the configuration take effect, stop the server and then start the server again.

Related concepts

"Variables" on page 146

Related reference

Session management custom properties

Related information

"HTTP transport custom properties" on page 49

WebSphere variables collection

Use this page to view and change a list of substitution variables with their values and scope.

To view this administrative console page, click **Environment > Manage WebSphere Variables**.

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

Related concepts

"Variables" on page 146

Related tasks

"Configuring WebSphere variables" on page 146

IBM Toolbox for Java JDBC driver

Configure and use the jt400.jar file

Related reference

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

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

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

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

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

Variable settings
Use this page to define the name and value of a WebSphere Application Server substitution variable.
To view this administrative console page, click **Environment > Manage WebSphere Variables >***WebSphere_variable_name*.

Related reference

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

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

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

For example, ″CONFIG_ROOT″ is the symbolic name representing the configuration directory path ″C:\WebSphere\AppServer\Config″ for the base WebSphere Application Server product on a Windows system.

Data type String

Value:

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

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

Data type String

Description:

Documents the purpose of a variable.

Data type String

Cell-wide z/OS variables

Several predefined WebSphere Application Server for z/OS cell-wide variables can be added using the administrative console.

Related reference

“HTTP transport custom properties” on page 49

protocol_http_defaultIdentity

Specifies the user identity associated with unauthenticated HTTP requests. This variable is used only if unauthenticated users have been permitted through the Global Security settings. The value is initially configured trough the WebSphere Application Server for z/OS ISPF Customization Dialog.

Data type String
Default WSADMIN

protocol_https_defaultIdentity

Specifies the user identity associated with unauthenticated HTTPS requests. Use this variable is used only if unauthenticated users have been permitted through the Global Security settings. The value is initially configured trough the WebSphere Application Server for z/OS ISPF Customization Dialog.

Data type String
Default WSADMIN

Repository Service Custom Properties

Use this page to add custom properties for the Repository Service.

Repository Service custom properties can be specified in the administrative console using the path **System Administration>Node Agents**>*Nodeagent*>**administrative services > repository service>Custom Properties** to get to this panel.

Support for the following custom property is provided with the WebSphere Application Server for z/OS product.

Related concepts

"Peer restart and recovery" on page 30

Related reference

"HTTP transport custom properties" on page 49

recoveryNode

Used to indicate that a node is a recovery node. Set this value to `true` if you want a node in a Network Deployment cell to act as a peer restart and recovery node for another node in the same cell. The recovery node shadows the complete configuration of its recovery peer. Use this property only if you need to support peer restart and recovery and are not using a shared file system.

Data type	Boolean

Configuring Server custom properties

Use this page to configure server custom properties on a WebSphere Application Server for z/OS system.

Server custom properties can be specified in the administrative console using the path **Application Servers** > *server*>**Custom Properties**. The support for the following server custom properties is provided with the WebSphere Application Server for z/OS product.

Related tasks

Using the WebSphere Application Server administrative console to enable properties for specific SMF record types

Related reference

"HTTP transport custom properties" on page 49

protocol_accept_http_work_after_min_srs

Used to indicate whether or not the application server is to wait for a minimum number of server regions (specified on the `wlm_minimumSRCount` variable) to be up before starting the HTTP transports. If this property is set to `true`, when the minimum number of server regions is ready for work, the HTTP transport will start accepting work. If this property is set to false, the HTTP transports are started when the control region comes up.

Data Type	Boolean
Default	false
Used by Daemon	No

protocol_bboc_log_response_failure

Used to indicate that if message BBOO0168W is issued, the failure detected when attempting to send a response to a client is to be recorded. The message is sent to the error log. The message text contains the request method name, the reply status, and routing information identifying the client.

Data Type	Boolean
Default	false
Used by Daemon	Yes

protocol_bboc_log_return_exception

Used to indicate that if message BBOO0169W is issued, the response that contains the SystemException is to recorded. The message is sent to the error log. The message text will contain the exception identifier and minor code, the request method name, and routing information identifying the client.

Data Type	Boolean
Default	false
Used by Daemon	Yes

protocol_giop_level_highest

Used to indicate the CORBA GIOP protocol version level used by the Application Server's ORB. Valid values are 1.1 and 1.2. Interoperable object references (IORs) exported from this server will use the GIOP level indicated.
You may need to change this property from the default if you are using a non-WebSphere Application Server client ORB that supports a lower version of the CORBA standard. For example, you may need to change from the default protocol version level of 1.2 to 1.1 to support an older, non-WebSphere Application Server client ORB.

Data Type	String
Default	1.2
Used by Daemon	Yes

protocol_http_backlog

Used to specify the maximum length for the queue of pending connections using HTTP. The value used may be limited by the specification of the `SOMAXCONN` statement in the TCP/IP Profile.

Data Type	Integer
Default	10
Used by Daemon	No

protocol_http_large_data_response_buffer

Used to specify, in bytes, the maximum length of the response buffer used for HTTP requests. Responses larger than this value are rejected. A value of 0 indicates not to allocate a large response buffer. An HTTP large response buffer is not required if all HTTP responses are less than 10 MB.

Data Type	Integer
Default	104857600
Used by Daemon	No

protocol_http_large_data_inbound_buffer

Used to specify, in bytes, the length of a serially reusable inbound buffer used for HTTP requests larger than 10 MB. Inbound HTTP requests larger than this value are rejected. The default is zero which indicates that no buffer is needed and requests larger than 10 MB are rejected.

Data Type	Integer
Default	0
Used by Daemon	No

protocol_https_backlog

Used to specify the maximum length for the queue of pending connections using HTTPS. The value used may be limited by the specification of the `SOMAXCONN` statement in the TCP/IP Profile.

Data Type	Integer
Default	10

Used by Daemon	No

protocol_iiop_backlog

Used to specify the maximum length for the queue of pending connections using IIOP. The value used may be limited by the specification of the `SOMAXCONN` statement in the TCP/IP Profile.

Data Type	Integer
Default	10
Used by Daemon	Yes

protocol_iiop_backlog_ssl

Used to specify the maximum length for the queue of pending connections using IIOP SSL. The value used may be limited by the specification of the `SOMAXCONN` statement in the TCP/IP Profile.

Data Type	Integer
Default	10
Used by Daemon	Yes

ras_debugEnabled

When set to `true`, this property enables an external debugger tool to be used with the Application Server for tracing and/or debugging client and server application components such as servlets, JSPs, and Enterprise beans.

Data Type	Boolean
Default	false
Used by Daemon	Yes

ras_default_msg_dd

Used to redirect write-to-operator (WTO) messages that use the default routing to hardcopy. These messages are redirected to the location identified through the DD card on the server's JCL start procedure. These WTO messages are primarily messages that WebSphere Application Server for z/OS issues during initialization.

Data Type	String
Default	empty string
Used by Daemon	Yes

ras_dumpoptions_dumptype

Used to specify the default dump used by the signal handler. You should not change this property unless directed by IBM service personnel.

0	No dump is generated.
1	A ctrace dump is taken.
2	A cdump dump is taken.
3	A csnap dump is taken.
4	A CEE3DMP dump is taken.

Data Type	Integer
Default	3
Used by Daemon	Yes

ras_dumpoptions_ledumpoptions
Used to specify dump options to be used with a CEE3DMP. If you want more than one option, separate each option with a blank. You should not change this property unless directed by IBM service personnel.

Data Type	String
Default	THREAD(ALL) BLOCKS
Used by Daemon	Yes

ras_hardcopy_msg_dd
Used to redirect write-to-operator (WTO) messages that WebSphere Application Server for z/OS routes to hardcopy. These messages are redirected to the location identified through the DD card on the server's JCL start procedure. These WTO messages are primarily audit messages issued from Java code during initialization.

Data Type	String
Default	empty string
Used by Daemon	Yes

ras_log_logstreamName
Used to specify the log stream for WebSphere Application Server for z/OS to use for error information. If the specified log stream is not found or not accessible, a message is issued and errors are written to the server's job log. If this variable is not specified, WebSphere Application Server for z/OS uses STDERR.

Data Type	String
Default	empty string
Used by Daemon	Yes

ras_minorcode_action
Determines the default behavior for gathering documentation about system exception minor codes.

Data Type	String
Default	NODIAGNOSTICDATA
Used by Daemon	Yes

The following values can also be specified:

- `CEEDUMP` - Captures callback and offsets. It takes time for the system to take CEEDUMPs. Transaction time-outs could occur.
- `TRACEBACK` - Captures Language Environment and z/OS UNIX traceback data.
- `SVCDUMP` - Captures an MVS dump (but will not produce a dump in the client).

ras_time_local
Used to control whether timestamps in the error log appear in local time or Greenwich Mean Time (GMT), which is the default.

Data Type	Boolean
Default	false
Used by Daemon	Yes

ras_trace_basic
Used to specify tracing overrides for particular WebSphere Application Server for z/OS subcomponents. Subcomponents, specified by numbers, receive basic and

exception traces. If you specify more than one subcomponent, use parentheses and separate the numbers with commas. Contact IBM service for the subcomponent numbers and their meanings. Do not change this property unless directed by IBM service personnel.

Data Type	String
Default	null (empty string)
Used by Daemon	Yes

ras_trace_BufferCount

Used to specify the number of trace buffers to allocate.

Data Type	Integer
Valid values	4 through 8
Default	4
Used by Daemon	Yes

ras_trace_BufferSize

Used to specify the size of a single trace buffer in bytes. You can use the letters ″K″ (for kilobytes) or ″M″ (for megabytes).

Data Type	String
Valid values	128K through 4M
Default	1M
Used by Daemon	Yes

ras_trace_ctraceParms

Identifies the `CTRACE PARMLIB` member. The value can be either a two-character suffix, which is added to the string `CTIBBO` to form the name of the PARMLIB member, or the fully-specified name of the PARMLIB member. For example, you could use the suffix ″01″, which the system resolves to `CTIBBO01`. A fully-specified name must conform to the naming requirements for a CTRACE PARMLIB member. For details, see z/OS MVS Diagnosis: Tools and Service Aids, GA22-7589.
If this property is specified and the PARMLIB member is not found, the default PARMLIB member, `CTIBBO00`, is used. If neither the specified nor the default PARMLIB member is found, tracing is defined to CTRACE, but there is no connection to a CTRACE external writer.

Data Type	String
Default	null (empty string)
Used by Daemon	Yes

ras_trace_defaultTracingLevel

Used to specify the default tracing level for WebSphere Application Server for z/OS. Use this variable in conjunction with the `ras_trace_basic` and `ras_trace_detail` variables to set tracing levels for Application Server for z/OS subcomponents. You should not change this property unless directed by IBM service personnel.

0	No tracing
1	Exception tracing
2	Basic and exception tracing
3	Detailed tracing, including basic and exception tracing

Data Type	Integer
Default	1
Used by Daemon	Yes

ras_trace_detail

Used to specify tracing overrides for particular WebSphere Application Server for z/OS subcomponents. Subcomponents, specified by numbers, receive detailed traces. If you specify more than one subcomponent, use parentheses and separate the numbers with commas. Contact IBM service for the subcomponent numbers and their meanings. Do not change this property unless directed by IBM service personnel.

Data Type	String
Default	null (empty string)
Used by Daemon	Yes

ras_trace_exclude_specific

Specifies WebSphere Application Server for z/OS trace points to exclude from tracing activity.
Trace points are specified by 8-digit, hexadecimal numbers. Do not use this property unless directed by IBM service personnel.

If IBM service personnel directs you to specify more than one trace point, use parentheses and separate the numbers with commas. You also can specify a WebSphere variable name by enclosing the name in single quotes.

Data Type	String
Default	empty string
Used by Daemon	Yes

Note: Sometimes results depend on the setting of another environment variable, ras_trace_minorCodeDefault. If you code `ras_trace_minorCodeTraceBacks=ALL` and `ras_minorcode_action=NODIAGNOSTICDATA`, you get a traceback. But, if you code `ras_trace_minorCodeTraceBacks=(null value)` and `ras_minorcode_action=TRACEBACK`, you also get a traceback. So, specifying `ras_trace_minorCodeTraceBacks=(null value)` does not cancel TRACEBACK; it simply does not cause a TRACEBACK data to be collected.

ras_trace_outputLocation

Specifies where you want trace records to be sent:

- To SYSPRINT
- To a memory buffer (BUFFER), the contents of which are later written to a CTRACE data set
- To a trace data set (TRCFILE) specified on the TRCFILE DD statement in the server's start procedure.

For servers, you may specify one or more values, separated by a space.

Data Type	String
Default	SYSPRINT BUFFER
Used by Daemon	Yes

ras_trace_specific
Used to indicate tracing overrides for specific WebSphere Application Server for z/OS trace points. Trace points are specified by 8-digit, hexadecimal numbers. To specify more than one trace point, use parentheses and separate the numbers with commas. You can also specify tracing on a specific environment variable by using the name enclosed in single quotes. Do not use this property unless directed by IBM service personnel.

Data Type	String
Default	null (empty string)
Used by Daemon	Yes

server_region_jvm_localrefs
Should only be used under the direction of IBM support personnel.

Data Type	Integer
Default	128
Used by Daemon	No

server_region_jvm_logfile
Used to specify the HFS file in which JNI and class debug messages from the JVM will be logged. Use this variable only in a single-server environment. If you use this property in a multiple-server environment, all the servers write to the same file, so you might have difficulty using the file for diagnostic purposes.

Data Type	String (file name)
Default	empty string (no file name)
Used by Daemon	No

server_region_recycle_count
Used to specify the number of transactions processed by a servant process after which the servant process will be recycled. WLM will end the server region once all affinity requirements have been met. Specify a non-zero value to enable recycling.
You might want to enable recycling if, after running for an extended period of time, your application is experiencing out-of-memory exceptions . (Out-of-memory exceptions can be the result of memory leakage by your application.)

Data Type	Integer
Default	0
Used by Daemon	No

server_start_wait_for_initialization_Timeout
Used to specify how long startServer.sh command processing waits for WebSphere Application Server initialization to complete. By default, it waits indefinitely until initialization is complete.
You might want to use this property if you want to:

- Control how long the application server will wait for other dependent servers to start.
- Limit the amount of wait time when trying to debug problems with application initialization. (For example, you might not want to continue waiting if auto-started Web applications unexpectedly enter a long wait.)

Data Type	Integer

Default	0
Used by Daemon	No

transaction_recoveryTimeout

Used to specify the time, in minutes, that this controller (region) uses to attempt to complete all restart transactions before issuing a write-to-operator-with-reply (WTOR) message to the console, requesting whether it should:

- Stop trying to resolve all restart transactions,
- Write transaction-related information to the job log or hard copy log, and
- Terminate.

If the operator replies that recovery should continue, the controller (region) will attempt recovery for the specified amount of time before reissuing the WTOR message. Once all the transactions are resolved, the control region terminates. This variable applies only to controllers in peer restart and recovery mode.

Data Type	Integer
Default	15
Used by Daemon	No

Shared library files

Shared library files in WebSphere Application Server consist of a symbolic name, a Java classpath, and a native path for loading Java Native Interface (JNI) libraries.

You can define a shared library at the cell, node, or server level. Defining a library at one of the three levels does not cause the library to be placed into the application server's class loader. You must associate the library to an application or server in order for the classes represented by the shared library to be loaded in either a server-wide or application-specific class loader.

A separate class loader is used for shared libraries that are associated with an application server. This class loader is the parent of the application class loader, and the WebSphere Application Server extensions class loader is its parent. Shared libraries that are associated with an application are loaded by the application class loader.

Related concepts

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

"Managing shared libraries" on page 158

Related reference

"Shared library collection" on page 159

"Shared library settings" on page 159

Managing shared libraries

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

1. Identify library files and their classpaths.
   a. Click **Environment > Shared Libraries** in the console navigation tree to access the Shared Libraries page.
   b. Change the scope of the collection table to see what shared libraries are in a cell, node, or server. Select the cell, a node, or a server and click **Apply**.
   c. Click **New**.
   d. On the settings page for a shared library, specify the name, classpath, and any other variables for the library file that are needed.
   e. Click **Apply**.

   Repeat this step until you define a shared library instance for each library file that your applications need.
2. Associate shared library files with an application that must use one or more shared libraries.

   Use this step to associate a file with an application or perform the next step to associate a file with an Application Server, which associates the file with every application on the server.
   a. Click **Applications > Enterprise Applications** in the console navigation tree.
   b. Click on the installed application that uses the shared libraries.
   c. Click **Libraries** to access the Library Ref page.
   d. Click **Add**.
   e. On the settings page for a library reference, specify variables for the library reference as needed.
   f. Click **Apply**.

   Repeat this step until you define a library reference instance for each library file that your application requires.
3. Click **Servers > Application Servers >***server_name* to associate a shared library with an Application Server for the run-time environment.

   Use this step to associate a file with an Application Server, which associates the file with every application on the server, or perform the previous step to associate a file with an application.
   a. Set the application class-loader (sometimes referred to as classloader) policy and application class-loader mode on the settings page for an application server as described in Class loading.
   b. Click **Libraries** on the settings page for a class loader.
   c. Click **Add** from the Library Ref page.
   d. Specify variables for the library reference as needed on the settings page for a library reference and click **OK**.

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

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

Related concepts

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.

"Shared library files" on page 157

Shared library collection

Use this page to define a list of shared library files that deployed applications can use.

To view this administrative console page, click **Environment > Shared Libraries**.

By default, a shared library is accessible to applications deployed (or installed) on the same node as the shared library file. Use the **Scope** field to change the scope to a different node or to a specific server.

Related tasks

"Managing shared libraries" on page 158

Related reference

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

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

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 shared library.

Description

Describes the shared library file.

Shared library settings

Use this page to make a library file available to deployed applications.

To view this administrative console page, click **Environment > Shared Libraries >***shared_library_name*.

Related tasks

"Managing shared libraries" on page 158

Related reference

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

Administrative console 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 shared library.

Data type	String

Description:

Describes the shared library file.

Data type	String

Classpath:

Specifies the class path used to locate the JAR files for the shared library support.

Data type	String
Units	Class path

Native Library Path:

Specifies the class path for locating platform-specific library files for shared library support; for example, .dll, .so, or *SRVPGM objects.

Data type	String
Units	Class path

Library reference collection

Use this page to view and manage library references that define how to use global libraries. For example, you can use this page to associate shared library files with a deployed application.

To view this administrative console page, click **Applications > Enterprise Applications >***application_name* **> Libraries**.

Related tasks

"Managing shared libraries" on page 158

Related reference

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

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

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

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

Library Name
Specifies a name for the library reference.

Library reference settings
Use this page to define library references, which specify how to use global libraries.
To view this administrative console page, click **Applications > Enterprise Applications >***application_name* **> Libraries >***library_reference_name*. A shared library must be defined to view this page.

Related tasks

"Managing shared libraries" on page 158

Class loading

Related reference

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

Administrative console 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.

Library Name:

Specifies the name of the shared library to use for the library reference.

Data type String

Environment: Resources for learning

Use the following links to find relevant supplemental information about configuring the WebSphere Application Server cell-wide environment. The information resides on IBM and non-IBM Internet sites, whose sponsors control the technical accuracy of the information.

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

View links to additional information about:
- "Environment: Resources for learning"
- "Environment: Resources for learning"

Programming instructions and examples
- WebSphere Application Server education

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

Administration
- Listing of all IBM WebSphere Application Server Redbooks

Related tasks

Chapter 9, “Configuring the cell-wide environment,” on page 139

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.